Kommerzielle Applikationen für Open Source Software und deutsches Urheberrecht

Till Jaeger

I. Einführung

II. Funktionsweise des Open Source-Lizenzmodells

III. Kommerzielle Nutzung von Freier Software

IV. Lizenzkompatibilität bei Copyleft-Lizenzen
1. Lizenzkompatibilität bei Verwendung der General Public License (GPL), Version 2
2. Lizenzkompatibilität bei Verwendung der General Public License (GPL), Version 3
3. Lizenzkompatibilität bei Verwendung der Mozilla Public License
4. Lizenzkompatibilität bei Verwendung der Lesser General Public License (LGPL), Version 2.1

V. Bearbeitung und Verbindung von Werken im Softwareurheberrecht
1. Beurteilung von Programmen in der Computerprogrammrichtlinie
2. Schnittstellen
3. Zur Reichweite des Begriffs der Umarbeitung in § 69 c Nr. 2 UrhG
4. Werkverbindungen bei Computerprogrammen

VI. Berücksichtigung urheberrechtlicher Gesichtspunkte bei der Auslegung von Copyleft-Klauseln

VII. Zusammenfassung

I. Einführung

Noch vor einigen Jahren führte Freie Software, auch Open Source Software genannt, ein Nischendasein und spielte in der Softwareindustrie nur eine untergeordnete Rolle. Einsatz und Weiterentwicklung von Freier Software fanden im Wesentlichen an Universitäten statt und mit den neuen Austauschmöglichkeiten des Internets dann auch in einer sich herausbildenden Entwickler-Community. Diese Entwickler-Community, bei der das Eigeninteresse an der Softwareprogrammierung im Vordergrund steht, konnte mit zunehmender Größe und verbesserter Vernetzung qualitativ

wettbewerbsfähige Software erstellen, die auch für professionelle Anwender und herkömmliche Softwareunternehmen von Interesse ist.

Wesentlich für diesen Erfolg ist auch die von Richard Stallman, den Gründer der Free Software Foundation (FSF), formulierte Intention, ein vollständiges System Freier Software parallel neben der herkömmlichen „proprietären" Softwarewelt ins Leben zu rufen.[1] Damit wurde die Entwicklung von Basiskomponenten und Systemsoftware gefördert, die oftmals modular aufgebaut ist. Dies wiederum erlaubt technisch den Rückgriff auf diese Softwarekomponenten – zumeist Programmbibliotheken, Datenbanken und Standardelemente – im Rahmen von Neuentwicklungen der Softwareindustrie.

So sind es nicht nur die bekannten Open Source-Programme wie Linux, Apache und MySQL, die ein breites Einsatzfeld gefunden haben, sondern auch zahlreiche Softwaretools und Standardkomponenten wie etwa uClibc, Qt und Busybox oder ganze Frameworks wie Eclipse. Proprietäre[2] und Freie Software wurden nicht mehr nur in ihren eigenen, getrennten Systemen eingesetzt und in vielen Bereichen bauten die Anbieter von Softwarelösungen mehr oder weniger auf vorbestehenden Open Source-Komponenten auf. Dieser Effekt beschleunigt sich noch dadurch, dass diejenigen Softwareunternehmen, die die Eigenentwicklung oder kostenpflichtige Lizenzierung von Standardelementen reduzieren können, im Wettbewerb einen Kostenvorteil besitzen.

Die Entwicklungsabteilungen waren oftmals Vorreiter dieser Hinwendung zum Einsatz von Freier Software und suchten gezielt auf den bekannten Entwicklerplattformen wie Sourceforge nach passenden Open Source-Programmen, während die Berücksichtigung der lizenzrechtlichen Vorgaben nicht selten in den Hintergrund trat. Inzwischen haben viele Softwareunternehmen die Voraussetzungen für eine Überprüfung der lizenzrechtlichen Pflichten geschaffen.

Hier stellen sich vor allem Fragen der lizenzrechtlichen Zulässigkeit der Kombination von Softwarebestandteilen unter Open Source-Lizenzen mit proprietären Programmen und Programmen unter abweichenden Open Source-Lizenzbedingungen. Diese Fragen sind – wie zu zeigen sein wird – eng mit den urheberrechtlichen Begriffen der Umarbeitung und Werkverbindung im Softwareurheberrecht verbunden.

---

1 Siehe den Aufruf vom 27. September 2003, abrufbar unter http://www.gnu.org/gnu/initial-announcement.html.

2 Der Begriff „proprietäre Software" soll sämtliche Lizenzierungsformen bezeichnen, die nicht den Open Source Definitionen entsprechen, insbesondere also die lizenzgebührenpflichtige Nutzung.

II. Funktionsweise des Open Source-Lizenzmodells

Um die Rechtsprobleme des Einsatzes von Freier Software in der Softwareindustrie beurteilen zu können, ist ein Blick auf die Funktionsweise der Open Source-Lizenzierung hilfreich. Sämtliche Open Source-Lizenzen haben Gemeinsamkeiten, die in den beiden maßgeblichen und im Verkehr anerkannten Definitionen, der Open Source Definition[3] und der Free Software Definition[4], niedergelegt sind. Da die Anforderungen, die die beiden Definitionen aufstellen, inhaltlich sehr weitgehend übereinstimmen, können die beiden Begriffe „Open Source Software" und „Freie Software" insoweit synonym verwendet werden.

Wesentliche Kernbedingung für alle Open Source-Lizenzen ist die Möglichkeit, an der Software umfassende Nutzungsrechte erwerben zu können. Der Lizenzgeber bietet jedermann ein einfaches Nutzungsrecht zur Vervielfältigung, Bearbeitung und zum Weitervertrieb an.[5] Wesentliche Vorbedingungen für eine freie Nutzbarkeit sind zudem das Gebot der Lizenzgebührenfreiheit und die Offenlegung des Quellcodes. Nur so ist sichergestellt, dass der Lizenznehmer die Software auch ungehindert weiterentwickeln und vertreiben kann. Der deutsche Gesetzgeber knüpft an die genannten Aspekte eine Reihe von Erleichterungen, um dieses Lizenzmodell abzusichern. So ermöglicht die sog. „Linux-Klausel" des § 32 Abs. 3 S. 3 UrhG einen Verzicht auf den Anspruch des Urhebers auf angemessene Vergütung gegen die Lizenznehmer.[6]

Angesichts der – nicht verbindlichen, aber im Verkehr akzeptierten – Vorgaben der Open Source Definitionen gleichen sich viele Open Source-Lizenzen sehr weitgehend. Hauptunterscheidungsmerkmal ist die Frage, wie Weiterentwicklungen lizenzrechtlich zu behandeln sind. Die sog. „Non-Co-

---

3 In der Version 1.9 unter http://www.opensource.org/osd.html abrufbar.

4 Abrufbar im Internet unter http://www.gnu.org/philosophy/free-sw.html.

5 Anbieter und Nutzer von Freier Software gehen wie selbstverständlich davon aus, dass der in den meisten Open Source-Lizenzen verwendete Begriff „distribution" nicht nur die Weitergabe von Vervielfältigungsstücken umfasst, sondern auch die öffentliche Zugänglichmachung der Software, vor allem das Angebot zum Download im Internet. Dies wurde nach einigen Literaturmeinungen angezweifelt, vgl. dazu *Jaeger/Metzger*, Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, 2. Aufl., 2006, Rn. 29. Die Version 3 der GNU General Public License (GPL) verwendet daher zur Klarstellung den Begriff „convey", der explizit auch das Online-Angebot mitumfasst.

6 Die Formulierung „Der Urheber kann aber unentgeltlich ein einfaches Nutzungsrecht für jedermann einräumen." findet sich seit der Novellierung des Urheberrechtsgesetzes durch den „2. Korb" auch in § 32 a Abs. 3 S. 2 UrhG und in § 32 c Abs. 3 S. 2 UrhG sowie in ähnlicher Form in § 31 a Abs. 1 S. 2 UrhG.

pyleft-Lizenzen“ machen dazu keine Vorgaben, so dass die Lizenztexte oftmals nur wenige Zeilen lang sind.[7] Dem Bearbeiter einer auf diese Weise lizenzierten Software steht es damit frei, ob er seine Weiterentwicklungen ebenfalls wieder unter der Ursprungslizenz als Open Source Software freigibt oder unter abweichenden Bedingungen lizenziert. So ist es auch zulässig, dass Weiterentwicklungen nur „proprietär“ angeboten werden. So verwenden auch herkömmliche Softwareunternehmen Programmbestandteile, die unter Non-Copyleft-Lizenzen frei erhältlich sind, im Rahmen ihrer Produkte.

Um die freie Nutzbarkeit von Software nachhaltig zu sichern, hat der Softwareentwickler und Gründer der FSF, Richard Stallman, ein neuartiges Lizenzmodell konzipiert. Dieses als „Copyleft“ bezeichnete Modell sieht ebenfalls eine lizenzgebührenfreie Nutzung der Software vor, verpflichtet den Lizenznehmer aber zur Freigabe seiner Weiterentwicklungen unter der Ursprungslizenz, wenn dieser die bearbeitete Software weitervertreibt.[8] Damit perpetuiert sich das Lizenzmodell und es wird sichergestellt, dass auch bei einer dezentralen Programmentwicklung die gesamte Software frei verfügbar bleibt. Bekanntester und in der Praxis wichtigster Vertreter dieses Lizenztyps ist die GNU General Public License (GPL)[9]. Unscharf wird der Copyleft-Effekt auch als „viral“ bezeichnet,[10] da bei der Kombination von Softwarebestandteilen das Erfordernis bestehen kann, zuvor proprietär lizenzierte Teile unter der entsprechenden Copyleft-Lizenz freizugeben. Die Charakterisierung als „viral“ ist jedoch missverständlich, da sie den Eindruck erweckt, die Lizenzierung als Freie Software würde automatisch erfolgen. Richtig ist jedoch, dass eine Lizenzierung immer nur mit dem entsprechenden Willen des Rechtsinhabers erfolgen kann und stets die Wahlfreiheit verbleibt, Weiterentwicklungen unter der Copyleft-Lizenz zu vertreiben oder überhaupt nicht an Dritte weiterzugeben. Hier zeigen sich bereits einige der für die kommerzielle Nutzung wesentlichen Lizenzentscheidungen, mit denen sich die Softwarewirtschaft beim Einsatz Freier Software auseinandersetzen muss.

---

7 Vgl. den Prototyp einer Non-Copyleft-Lizenz, die „BSD License“, http://opensource.org/licenses/bsd-license.php.

8 Dazu ausführlich *Jaeger/Metzger*, Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, 2. Aufl., 2006, Rn. 45 ff; What is Copyleft?, http://www.gnu.org/copyleft/copyleft.html.

9 Abrufbar in der Version 2 unter http://www.gnu.org/licenses/old-licenses/gpl-2.0.html sowie in der neuen Version 3 unter http://www.fsf.org/licensing/licenses/gpl.html.

10 Vgl. etwa *Funk/Zeifang*, CR 2007, 617, 618.

III. Kommerzielle Nutzung von Freier Software

„Freie Software“ darf nicht mit „kostenloser Software“ verwechselt werden. Nach einem geflügelten Wort ist „frei“ wie in „Freiheit“ und nicht wie in „Freibier“ zu verstehen.[11] In der Free Software Definition heißt es dementsprechend: *“Free software does not mean non-commercial. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important.”* Ausgeschlossen sind lediglich Lizenzgebühren, d. h. ein Entgelt für die Einräumung von Nutzungsrechten.[12] So kann die Software selbst durchaus verkauft oder vermietet werden, d. h. ein Entgelt für die dauerhafte oder vorübergehende Übertragung einer Programmkopie verlangt werden. Ebenso können für begleitende Dienstleistungen oder die Einräumung von Gewährleistungsrechten eine Vergütung verlangt werden.

Der Unterschied zwischen unzulässigen Lizenzgebühren und dem erlaubten Verkaufspreis wird vor allem im Dreipersonenverhältnis deutlich: Der Verkäufer kann für einen Datenträger mit der Software einen beliebigen Preis verlangen,[13] jedoch nicht verhindern, dass der Käufer selbst unentgeltlich weitere Vervielfältigungsstücke erstellt und an Dritte entgeltlich oder kostenlos weitergibt.[14] „Royalties“ sind somit ausgeschlossen, was im Regelfall wirtschaftlich einen Einfluss auf die Verkaufspreise von Freier Software hat. Ist der Betrag zu hoch, wird sich zumeist jemand finden, der eine kostenlose Kopie im Internet zum Download anbietet, so dass sich der Verkaufspreis entsprechend reguliert und einem akzeptierten Gegenwert für die entgeltlichen Leistungen entspricht.

Im Zweipersonenverhältnis wird die Frage vor allem relevant, wenn der Erwerber die Software auf mehreren Rechnern installieren möchte. Der Verkäufer darf seinen Verkaufspreis nicht von der Zahl der Installationen abhängig machen, wohl aber Supportgebühren an diesen Parameter knüpfen. In der Praxis wird durchaus versucht, diese Beschränkung der Preisgestaltung zu umgehen, indem durch Markenrechte oder Kombinationen mit proprietärer Software ein neuer Anknüpfungspunkt für Lizenzgebüh-

---

11 Vgl. http://www.gnu.org/philosophy/free-sw.html.

12 Zu einem fehlerhaften Verständnis gelangt insoweit *Koch*, ITRB 2007, 261 (262 f.), bei der Auslegung der Version 3 der GPL.

13 So etwa ausdrücklich Ziffer 4 GPLv3: *“You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.”*

14 Explizit in Ziffer 10 GPLv3: *“For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License …”*

ren geschaffen wird. Die Grenzen der Zulässigkeit sind eine Frage der Gestaltung im Einzelfall, jedoch wird dem Erwerber zumeist die Möglichkeit verbleiben, die Freie Software von solchen proprietären Zusätzen zu befreien und dann unentgeltlich zu vervielfältigen. Wirtschaftlich am bedeutsamsten ist der gemeinsame Vertrieb von Open Source Software und proprietärer Software in einem einheitlichen Produkt. Hier fragt sich dann bei Copyleft-Lizenzen, ob die proprietären Bestandteile beliebig lizenziert werden können oder ebenfalls als Open Source Software freigegeben werden müssen.

IV. Lizenzkompatibilität bei Copyleft-Lizenzen

Das Problem der Lizenzkompatibilität ist eine neuere Entwicklung, die auf dem herkömmlichen Softwaremarkt nahezu unbekannt war. In der proprietären Softwarewelt werden zwar ebenfalls Programmbibliotheken und andere Softwarekomponenten hinzugekauft, um sie in einem eigenen Produkt zu integrieren, aber da bei den verwendeten Lizenzbedingungen keine „Copyleft"-Klauseln enthalten sind, ergeben sich auch keine besonderen Kompatibilitätsprobleme: Der Erwerber hat darauf zu achten, dass ihm die erforderlichen Nutzungsrechte für die Integration und den Vertrieb in seinem Produkt eingeräumt werden, im Rahmen der Weiterveräußerung spielen die Lizenzbedingungen der inkorporierten Teile aber regelmäßig keine Rolle mehr.

Anders ist die Lage bei Copyleft-Lizenzen wie der GPL. Hier stellt sich die Frage nach der Reichweite des Copyleft. Wenn etwa Softwarekomponente A und Softwarekomponente B in einem Programm kombiniert werden und Softwarekomponente A einer Copyleft-Lizenz untersteht, können sich Kompatibilitätsprobleme ergeben, wenn die Komponenten A und B so zu einem neuen Werk verbunden werden, dass dieses insgesamt nur unter den Lizenzbedingungen der Softwarekomponente A vertrieben werden darf. Denn nur wenn die Lizenzbedingungen der Softwarekomponente B auch die Lizenzierung unter der Copyleft-Lizenz von Komponente A erlauben, ist der gemeinsame Vertrieb zulässig. Kurz gesagt: Die Lizenzen müssen kompatibel sein.

In der Praxis ergeben sich eine Reihe von unterschiedlichen Konstellationen, abhängig von der verwendeten Copyleft-Lizenz.

1. Lizenzkompatibilität bei Verwendung der General Public License (GPL), Version 2

Die GPLv2 enthält die am weitestreichende Copyleft-Klausel unter den gebräuchlichen Open Source-Lizenzen. Alles, was als „derivative work" („abgeleitetes Werk") im Sinne der Lizenz anzusehen ist, muss unter der GPLv2 lizenziert werden, wenn die veränderte Software weiterverbreitet wird. Dabei ist der Inhalt dessen, was unter den Begriff „derivative work" fällt, eine Frage des Einzelfalls und dürfte im Wesentlichen alle Bearbeitungen der Ursprungssoftware umfassen.[15]

Die GPLv2 betont in Ziffer 6 ausdrücklich, dass dem Enderwerber vom Lizenznehmer keine Beschränkungen aufgebürdet werden dürfen, die die GPLv2 selbst nicht kennt: *"You may not impose any further restrictions on the recipients' exercise of the rights granted herein."* Somit ist eine Kombination von GPLv2-Code mit anders lizenzierter Software im Rahmen eines „derivative work" nur in wenigen Fällen möglich:

- Die hinzugefügte Software darf unter einer Lizenz genutzt werden, die explizit auch die Nutzung unter den Lizenzbedingungen der GPLv2 gestattet. Solche „Kompatibilitätsklauseln" finden sich etwa in Ziffer 3 LGPL, § 3 (3) der Deutschen Freien Software Lizenz (d-fsl)[16] und Ziffer 5 der European Public License (EUPL)[17]. Durch diese Form des „Dual Licensing"[18] akzeptieren Lizenzbedingungen in anderen Copyleft-Lizenzen, dass in einem Konfliktfall die Regelungen der GPLv2 vorgehen. Da diese Lizenzen Pflichten vorsehen, die die GPLv2 nicht kennt, wäre ansonsten ein Vertrieb als „derivative work" wegen Verstoß gegen Ziffer 6 GPLv2 nicht zulässig.
- Die hinzugefügte Software darf unter einer Lizenz genutzt werden, die keine Verpflichtungen enthält, die über die durch die GPLv2 vorgesehenen Lizenzpflichten hinausgehen. Dies ist nur bei einigen Non-Copyleft-Lizenzen der Fall, die nahezu keine Lizenzpflichten enthalten, etwa der MIT License.[19]

---

15 Detaillierte Ausführungen zur Auslegung des Begriffs „derivative work" in Ziffer 2 GPL finden sich bei *Jaeger*, in: ifrOSS (Hrsg.), Die GPL kommentiert und erklärt, 2004, Ziffer 2, Rn. 10 ff.

16 Abgedruckt bei *Jaeger/Metzger*, Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, 2. Aufl., 2006, S. 257 ff.

17 Abrufbar unter http://ec.europa.eu/idabc/en/document/6523.

18 Siehe zu diesem Begriff *Jaeger/Metzger*, Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, 2. Aufl., 2006, Rn. 114 ff.

19 Siehe unter http://www.opensource.org/licenses/mit-license.php.

– Der Anbieter besitzt die ausschließlichen Nutzungsrechte an der hinzugefügten Software und kann diese unter der GPLv2 lizenzieren. Zunehmend bieten Softwareunternehmen Programme sowohl unter einer proprietären Lizenz und unter der GPLv2 an, um einerseits die GPLv2-Version als Standard zu etablieren oder von Entwicklungsbeiträgen Dritter zu profitieren, und andererseits Lizenznehmern, die auf eine herkömmliche Lizenzierung angewiesen sind, dennoch eine proprietäre Lizenz anbieten zu können.

2. Lizenzkompatibilität bei Verwendung der General Public License (GPL), Version 3

Die Verbesserung des Problems der Lizenzkompatibilität war ein wesentliches Anliegen der Autoren der Version 3 der GPL.[20] Das in den ersten Entwürfen noch recht komplexe Regelwerk schrumpfte schließlich auf eine Liste zulässiger Pflichten, die zusätzlich zu den in der GPL ohnehin vorgesehenen Pflichten vorgesehen werden dürfen, um so die Kompatibilität mit anderen Lizenzen zu ermöglichen. Praktisch ist dieser Katalog an der Apache License 2.0 ausgerichtet, da hier die Kompatibilität für besonders wichtig erachtet wurde. So sind nach Ziffer 7 e) GPLv3 auch gewisse Freistellungsklauseln gestattet, die die Apache License in ihrer Ziffer 9 ebenfalls vorsieht. Somit ist die GPLv3 im Gegensatz zur GPLv2 mit der Apache License 2.0 kompatibel. GPLv2 und GPLv3 sind wiederum nicht kompatibel. Dieses Problem wird allerdings dadurch abgeschwächt, dass die GPLv2 in Ziffer 9 zugleich die Nutzung unter neueren Lizenzversionen zulässt.[21]

Eine spezielle Klausel zur Lizenzkompatibilität findet sich in Ziffer 13 GPLv3, die die Nutzung von Softwarekombinationen unter der GPL und der Affero General Public License, Version 3 (AGPL) unter der AGPL gestattet. Die AGPL geht insoweit über die GPL hinaus, als sie auch bei ASP-Angeboten, bei denen keine Programmkopie auf den Rechner des Nutzers übertragen wird, die Bereitstellung des Sourcecodes verlangt. Dies ist unter der GPLv3 nicht erforderlich. Programme, die sowohl Bestandteile unter der GPLv3 und der AGPLv3 beinhalten, können insgesamt unter der AGPLv3 genutzt werden.

---

20 Dazu näher die folgenden Erläuterungen der FSF: GPLv3 Final Discussion Draft Rationale, http://gplv3.fsf.org/gpl3-dd4-rationale.pdf, und GPLv3 Third Discussion Draft Rationale, http://gplv3.fsf.org/gpl3-dd3-rationale.pdf.

21 Eine Ausnahme stellt Linux dar, hier ist die Lizenz auf „GPLv2 only" beschränkt, vgl. *Jaeger*, in: ifrOSS (Hrsg.), Die GPL kommentiert und erklärt, 2004, Ziffer 2, Rn. 17.

3. Lizenzkompatibilität bei Verwendung der Mozilla Public License

Relativ einfach ist die Beurteilung der Lizenzkompatibilität bei der Mozilla Public License, die für die verschiedenen Nachfolgeprogramme des Netscape Communicators Anwendung findet und Vorbild für eine Reihe von weiteren Lizenzen ist.[22] Die MPL sieht nur ein „beschränktes Copyleft" vor, d. h. es müssen nicht alle Bearbeitungen unter der MPL lizenziert werden, sondern nur solche, die sich in derselben Datei mit Quellcode unter der MPL befindet. Dies ergibt sich aus der Copyleft-Klausel in Ziffer 3.1 MPL und der dort relevanten Definition von „Modification" in Ziffer 1.9 MPL.

Wenn sich also die Programmkomponenten, die mit Quellcode unter der MPL kombiniert werden sollen, in eigenen Quellcodedateien befinden, bestehen aus Sicht der MPL keine Kompatibilitätsprobleme. Damit ist die Kombination mit Software unter einer proprietären Lizenz oder einer Non-Copyleft-Lizenz einfach möglich, wenn diese formale Trennung beachtet wird. Erfolgt diese Trennung in eigene Dateien nicht, ist ein Vertrieb der veränderten Software nur möglich, wenn diese insgesamt der MPL unterstellt wird. Keine Kompatibilität besteht jedoch mit anderen Copyleft-Lizenzen wie der GPL, die ebenfalls verlangen würden, dass gemeinsame Codekombinationen der Ursprungslizenz unterstellt werden und zwar unabhängig davon, ob sich diese in getrennten Dateien befinden.

4. Lizenzkompatibilität bei Verwendung der Lesser General Public License (LGPL), Version 2.1

Die LGPL ist eine Open Source-Lizenz, die von der FSF für Programmbibliotheken entwickelt wurde.[23] Ziel war es, die Verlinkung sowohl mit GPL-Programmen als auch mit proprietärer Software zu ermöglichen, um solche Bibliotheken als Standard durchzusetzen. Dies wird dadurch erreicht, dass die Lizenz zwischen Änderungen an der Programmbibliothek selbst und der Verlinkung mit einem anderen Programm unterscheidet. Während Änderungen an der Bibliothek selbst mit Ausnahme der in Ziffer 7 LGPL geregelten Fälle bei der Weitergabe unter der LGPL lizenziert werden müssen, darf das auf die Bibliothek zugreifende Programm nach freier

22 Der Lizenztext der MPL ist abrufbar unter http://www.mozilla.org/MPL/MPL-1.1.html. Weitere Lizenzen mit nahezu gleichem Wortlaut finden sich unter http://www.ifross.de/ifross_html/lizenzcenter.html#MPL.

23 Der Lizenztext ist unter http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html abrufbar.

Wahl des Lizenznehmers proprietären Lizenzbedingungen unterstehen oder auch einer Open Source-Lizenz. Dies gilt unabhängig davon, ob das zugreifende Programm und die Bibliothek ein einheitliches neues Programm bilden oder nicht.

Da bei der Kombination einer LGPL-Bibliothek mit einem GPL-Programm die Lizenzbedingungen von LGPL und GPL eingehalten werden müssen und somit die Möglichkeit besteht, dass die gesamte Programmkombination der GPL unterstellt werden muss, sieht Ziffer 3 LGPL eine besondere Kompatibilitätsklausel vor, die die Nutzung der Bibliothek auch unter der GPL gestattet. Somit wird die unproblematische Verwendung von LGPL-Bibliotheken mit GPL-Programmen sichergestellt.

V. Bearbeitung und Verbindung von Werken im Softwareurheberrecht

Wie oben gezeigt wurde, sind gerade Copyleft-Lizenzen nicht kompatibel, wenn sie keine spezielle Kompatibilitätsklausel enthalten, da wegen abweichender Lizenzbedingungen nicht zugleich alle betroffenen Copyleft-Lizenzen erfüllt werden können. Damit kommt der weiteren Frage Bedeutung zu, in welchen Fällen die Frage der Lizenzkompatibilität überhaupt relevant ist. Wenn eigenständige Programme zusammen vertrieben werden, können diese unter verschiedenen Lizenzbedingungen angeboten werden, die Frage der Lizenzkompatibilität stellt sich nicht. Die Beurteilung, wann urheberrechtlich eigenständige Werke vorliegen und wann Softwareprogramme zu einem neuen einheitlichen Werk verbunden werden, ist im Softwareurheberrecht noch kaum analysiert worden. Die besondere Schwierigkeit besteht darin, dass Software im Regelfall dafür vorgesehen ist, mit anderen Programmen zusammen abzulaufen, Daten auszutauschen oder zu kommunizieren. Die selbständige Ablauffähigkeit ist damit als Kriterium unzureichend. So sind etwa Anwendungsprogramme ganz überwiegend nicht ohne ein Betriebssystem ablauffähig. Dennoch besteht insoweit Konsens, dass Anwendungsprogramme unabhängige Werke darstellen und keine urheberrechtliche Gestattung von den Rechtsinhabern der Betriebssysteme benötigt wird, auf denen sie ablauffähig sind.

1. Beurteilung von Programmen in der Computerprogrammrichtlinie

Ein wesentlicher Hinweis für die Beurteilung der Eigenständigkeit von Computerprogrammen lässt sich der Richtlinie 91/250/EWG über den Rechtsschutz von Computerprogrammen von 1991 entnehmen. Dort heißt es in den Erwägungsgründen:

„Die Funktion von Computerprogrammen besteht darin, mit den anderen Komponenten eines Computersystems und den Benutzern in Verbindung zu treten und zu operieren. Zu diesem Zweck ist eine logische und, wenn zweckmäßig, physische Verbindung und Interaktion notwendig, um zu gewährleisten, dass Software und Hardware mit anderer Software und Hardware und Benutzern wie beabsichtigt funktionieren können.
Die Teile des Programms, die eine solche Verbindung und Interaktion zwischen den Elementen von Software und Hardware ermöglichen sollen, sind allgemein als ‚Schnittstellen' bekannt.
Diese funktionale Verbindung und Interaktion ist allgemein als ‚Interoperabilität' bekannt. Diese Interoperabilität kann definiert werden als die Fähigkeit zum Austausch von Informationen und zur wechselseitigen Verwendung der ausgetauschten Informationen."

Hier wird deutlich, dass der Austausch von Daten über Schnittstellen als normale Funktion eines Computerprogramms angesehen wird. Damit lässt sich sagen, dass zumindest Programme, die lediglich über Schnittstellen kommunizieren, als eigenständig anerkannt werden können. In den Erwägungsgründen wird in diesem Zusammenhang weiterhin explizit auf die Unabhängigkeit der Entwicklung abgestellt: *„. . . die für die Interoperabilität eines unabhängig geschaffenen Programms mit anderen Programmen notwendig sind:".* Dieser Aspekt dürfte gerade bei der Entwicklung von Open Source-Programmen Bedeutung haben, wie nachfolgend noch erläutert wird.

2. Schnittstellen

Die Informatik fasst unter den Begriff der Schnittstelle bzw. des „Interfaces" verschiedene technische Gestaltungen. So wird zwischen Kommunikationsschnittstellen (*inter-process communication*, IPC), Programmierschnittstellen (*application programming interface*, API) und Komponentenschnittstellen für modular gestaltete Programme unterschieden. Hier fragt sich, ob

alle diese Schnittstellen für die Einordnung von Programmkomponenten als eigenständige Programme herangezogen werden können.

a) Programmierschnittstellen und Systemaufrufe

Bei den meisten Betriebssystemen wie Linux und Windows besitzt der Betriebssystemkern den alleinigen Zugriff auf die Hardwarefunktionen der Hauptrecheneinheit eines Computers. Sämtliche Anwendungsprogramme müssen für den Zugriff auf die Hardware über sog. Systemaufrufe (*system calls*) auf den Betriebssystemkern zugreifen. Diese Trennung dient der besseren Stabilität des Gesamtsystems. Der Zugriff der Systemaufrufe auf den Betriebssystemkern erfolgt über ein ABI (*application binary interface*), die Binärschnittstelle. Für diesen Standardfall der Kommunikation zwischen Anwendungsprogramm und Betriebssystemkern wird man von einer klassischen Interoperabilität eigenständiger Programme ausgehen können.

Während ein ABI eine Schnittstelle auf Binärebene darstellt, enthalten APIs auf der Quellcodeebene eine Definition, wie eine Anbindung anderer Programme erfolgen kann.[24] Sie dienen damit zur Herstellung von Interoperabilität zwischen selbständigen Programmen. Neben APIs für den Austausch von Informationen zwischen Anwendungsprogrammen kommt der Programmierschnittstelle zum Ablauf eines Anwendungsprogramms auf einem Betriebssystem besondere Bedeutung zu. Dabei ist es durchaus üblich, dass gleichsam zwischen der Anwendung (die in verschiedensten Programmiersprachen implementiert sein kann) und dem Betriebssystemkern eine Basisbibliothek die Datenkonvertierung übernimmt und dann an die ABI weiterreicht. Somit können Anwendungen auf verschiedenen Betriebssystemen ablauffähig sein.

b) Kommunikationsschnittstellen

Bei der *inter-process communication* dienen die Schnittstellen der Kommunikation in verteilten Systemen, d. h. zwischen selbständigen Prozessen, die miteinander interagieren. Dazu gehört insbesondere die Netzwerkkommunikation in Client-Server-Systemen, aber auch das Ablaufen selbständiger Prozesse in einem Rechner. Klassische Techniken für die

24 Bei den Programmiersprachen C und C++ erfolgt diese Definition üblicherweise in sog. Headerfiles. Bei der Kompilierung wird dann überprüft, ob die ausgetauschten Daten der Definition entsprechen, sodann wird eine Objectcode-Datei erzeugt.

inter-process communication sind „Pipes“[25] und „Sockets“[26]. Traditionell werden Kommunikationsschnittstellen für die Interaktion von unabhängigen Programmen verwendet, insbesondere zur Datenkommunikation, so dass diese Technik als ein Indiz für Eigenständigkeit anzusehen ist.

c) Komponentenschnittstellen

In der modernen Softwareentwicklung sind Komponentenmodelle beliebt. Dadurch wird die Software modular gestaltet, so dass Softwarekomponenten in einem anderem Kontext wiederverwendet werden können. Die Verbindung von Softwarekomponenten erfolgt wiederum durch Schnittstellen. Diese definieren den Umfang der Abhängigkeiten von anderen Teilen der Software. Beispiele für solche Softwarekomponenten sind etwa Plugins, und Kernelmodule, die zur funktionalen Erweiterung eines Programms dienen.[27] Auch die objektorientierte Programmierung mit Programmklassen führt zu einem modularen Programmaufbau, dort dienen die Schnittstellen der Definition von Methoden einer Klasse.

Bei der Frage, ob Komponentenschnittstellen eigenständige Werke trennen, zeigt sich, dass das vorherrschende Programmverständnis aus der Zeit der Computerprogramm-Richtlinie in Auflösung begriffen ist. Während Programmier- und Kommunikationsschnittstellen im Regelfall eigenständige Programme bzw. Betriebssystem und Anwendungsprogramme trennen, lässt sich dies von Komponentenschnittstellen nicht in dieser Allgemeinheit sagen. Ein Programm kann aus mehreren Komponenten bestehen, die zudem in unterschiedlich starkem Umfang voneinander abhängig sind. Dies kann im Extremfall dazu führen, dass eine Komponente in unveränderter Form für eine Vielzahl von Programmen verwendbar ist oder eben so eng mit einem einzigen Programm verwoben ist, dass sie mehr oder weniger in diesem Programm aufgeht. So hat beispielsweise die

---

25 Unter einer Pipe wird in der Informatik ein Datenstrom zwischen zwei Prozessen verstanden, so dass die Ausgabe eines Programms als Eingabe für ein anderes Programm verwendet werden kann.

26 Sockets können der Netzwerkkommunikation über APIs sowie der lokalen Interprozesskommunikation dienen. Insbesondere bei der plattformunabhängigen Programmierung wie bei JAVA werden Sockets verwendet.

27 Daneben bestehen durchaus noch weitere Techniken, die zu einer weitergehenden Unabhängigkeit führen. So werden eigene Scriptsprachen verwendet, um Erweiterungen (*extensions*) für ein Programm entwickeln zu können. Die Scripte werden dann durch einen eigenen Interpreter ausgeführt. Der Entwickler benötigt dann wenige Kenntnisse des Programms, für das die Erweiterung entwickelt wird. Ein Beispiel sind die Erweiterungen für den Browser Firefox.

Entwicklung von Kernelmodulen für Linux gezeigt, dass die Verknüpfung mit dem Kernel im Laufe der Zeit zunehmend intensiver erfolgt, so dass bei der Analyse von Fehlern oftmals nicht ohne Weiteres festgestellt werden kann, ob dieser aus einem Kernelmodul stammt.[28]

3. Zur Reichweite des Begriffs der Umarbeitung in § 69 c Nr. 2 UrhG

Von urheberrechtlicher Bedeutung ist zunächst die Frage, wann eine Umarbeitung i. S. d. § 69 c Nr. 2 UrhG vorliegt, da hierfür regelmäßig die Zustimmung der Rechtsinhaber des Ursprungsprogramms erforderlich ist.[29] Rechtsprechung und Literaturmeinungen finden sich zu dieser Frage nur spärlich. In der Entscheidung „Programmfehlerbeseitigung" hat der BGH offengelassen, ob das Hinzufügen von zusätzlichen Modulen eine Umarbeitung darstellt.[30] Nimmt man die Verwendung von Schnittstellen zum entscheidenden Anknüpfungskriterium, so wäre keine urheberrechtliche Zustimmung für die Hinzufügung solcher Module erforderlich, die an eine bestehende Schnittstelle eines anderen Programms oder Programmteils angebunden werden. Diese Auslegung ist vor dem Hintergrund des § 69 e UrhG vorzugswürdig, da die Schranke zugunsten der Dekompilierung zur Herstellung von Interoperabilität nur dann eine sinnvolle Bedeutung hat, wenn die unabhängig geschaffene und mit dem dekompilierten Programm interoperable Software auch ohne weitere Zustimmung genutzt werden darf.

Dies dürfte auch dann gelten, wenn ein Programm modular aufgebaut ist und über eine Komponentenschnittstelle nur ein Modul angebunden wird. Denn die Frage, wann ein „Programm" und wann nur eine „Programmkomponente" vorliegt, lässt sich praktisch nicht sinnvoll abgrenzen. Zumeist werden mehrere Funktionalitäten in einem einheitlichen Programm zusammengefasst (z. B. Webbrowser). Es ist aber jederzeit möglich, einzelne Funktionalitäten auszugliedern und in einer eigenen Einheit programmiertechnisch umzusetzen. Die Grenzen sind daher fließend und die bloß formale Betrachtung letztlich nur eine Frage der Programmiertechnik, umgekehrt

28 Dazu näher *Jaeger*, in: ifrOSS (Hrsg.), Die GPL kommentiert und erklärt, 2004, Ziffer 2, Rn. 29 ff.

29 Keiner Zustimmung bedarf es im Falle einer notwendigen Fehlerbehebung, die durch die gesetzliche Lizenz des § 69 d UrhG gedeckt ist, vgl. *BGH*, GRUR 2000, 866 (868) – *Programmfehlerbeseitigung*.

30 *BGH*, GRUR 2000, 866 (868) – *Programmfehlerbeseitigung*.

aber die Beurteilung, was „ein Programm“ ausmacht, eine Wertung ohne nachvollziehbare Kriterien.

Stellt man jedoch alleine darauf ab, ob Schnittstellen vorliegen und beschränkt man die Zustimmungsbedürftigkeit auf Umarbeitungen innerhalb von Programmteilen, die nur nach außen mit Schnittstellen abgetrennt sind, bedeutet dies, dass das Tatbestandsmerkmal „unabhängig geschaffenes Programm“ in § 69 e UrhG nicht aufgrund einer wertenden Betrachtung im Einzelfall auszulegen ist, sondern anhand eines formalen Kriteriums, nämlich der Frage, ob eine Programmierung über eine Schnittstelle mit einem anderen Programm kommuniziert oder dieses Programm anderweitig verändert wurde.

Jedoch wird auch diese Betrachtung nicht ohne inhaltliche Beschränkungen auskommen. So wird man zunächst verlangen müssen, dass der Entwickler des Ursprungswerkes eine Schnittstelle zur Verfügung gestellt hat, die die Anbindung von anderen Programmen oder Programmmodulen erlaubt. Die Schaffung einer eigenen Schnittstelle im fremden Ursprungswerk – wie dies bei Open Source Software wegen des zugänglichen Quellcodes einfach möglich ist – kann im Regelfall nicht dazu führen, das Zustimmungserfordernis zu umgehen. Denn dann hat nicht der Urheber selbst die Voraussetzungen für die Interoperabilität seines Programms geschaffen, sondern der Bearbeiter. Würde man dies genügen lassen, wäre es nur eine Frage des technischen Aufwands, das, was eigentlich eine Bearbeitung eines Programms darstellt, über eine hinzugefügte Schnittstelle „herauszuprogrammieren“.

Weiterhin wird man eine Zustimmung für erforderlich halten müssen, wenn der Entwickler eines Softwaremoduls nicht nur dieses programmiert, sondern auch das Ursprungsprogramm so abändert, damit dieses überhaupt in der Lage ist, mit dem hinzugefügten Modul ablauffähig zu sein. Dann wird die Abhängigkeit zwischen den Programmbestandteilen unter Umständen so groß sein, dass man auch im Hinblick auf das über die Schnittstelle angebundene Modul nicht mehr von einem *unabhängig* geschaffenen Programm ausgehen kann. Eine Gegenausnahme kann jedoch dann vorliegen, wenn die Änderungen in dem Ursprungsprogramm keine Relevanz für das hinzugefügte Modul haben. Auch hier sind angesichts der Programmiervielfalt die Grenzen fließend.[31]

31 Gerade bei neuen Programmiertechniken, wie z. B. der aspektorientierten Programmierung, ist zu prüfen, inwieweit sich das klassische Verständnis von Schnittstellen übertragen lässt.

4. Werkverbindungen bei Computerprogrammen

Über § 69 a Abs. 4 UrhG ist auch die Regelung des § 9 UrhG zu Werkverbindungen anwendbar.[32] Sollen mehrere Softwarekomponenten „zur gemeinsamen Verwertung" verbunden werden, so bedarf dies einer rechtsgeschäftlichen Einigung.[33] Dies ist jedoch dann nicht der Fall, wenn lediglich ein Softwaremodul für eine offengelegte Schnittstelle entwickelt wird oder diese ohne Zustimmung des Rechtsinhabers dekompiliert wurde. Es fehlt dann an dem Merkmal einer Verbindung „zur gemeinsamen Verwertung".

Open Source-Lizenzen erlauben jedoch stets die gemeinsame Verwertung einer Open Source Software mit hinzugefügten Bestandteilen. Betrachtet man jedes über eine Schnittstelle abgrenzbare Programmmodul, das die erforderliche Individualität aufweist, als eigenständiges Werk, so liegt konsequenterweise ein verbundenes Werk gem. § 9 UrhG vor, wenn die Programmbestandteile so zusammen vertrieben werden, dass sie gemeinsam ablaufen können.

Es steht jedem Urheber frei, ob er der Verbindung seines Werkes mit Werken anderer Urheber zustimmt. Daher kann er seine Zustimmung durchaus davon abhängig machen, wie die Art und Weise der Verwertung erfolgen soll. Dies zeigt, dass urheberrechtlich unterschiedliche Anforderungen an die Erstellung und den Vertrieb von eigenständigen Softwarekomponenten sowie den Vertrieb von verbundenen Softwarekomponenten gestellt werden können. Während in ersterem Fall keine lizenzrechtliche Abhängigkeit besteht, bedarf es für die Nutzung von zur gemeinsamen Verwertung verbundener Programme einer vorherigen Zustimmung. Diese Unterscheidung ist gerade für Open Source-Lizenzen beachtlich.

VI. Berücksichtigung urheberrechtlicher Gesichtspunkte bei der Auslegung von Copyleft-Klauseln

Wie oben gezeigt wurde, können die Copyleft-Klauseln in Open Source-Lizenzen eine ganz unterschiedliche Reichweite besitzen. Dies führt zu der Frage, ob und in welchem Umfang Copyleft-Klauseln die Lizenzierung von Programmbestandteilen Dritter unter den eigenen Lizenzbedingungen verlangen dürfen. Dies gilt insbesondere für den Fall, wenn sich das Copyleft

32 *Dreier*, in: Dreier/Schulze, UrhG, 2. Aufl., 2006, § 69a, Rn. 34.

33 *Loewenheim*, in: Schricker (Hrsg.), Urheberrecht, 3. Aufl., 2006, § 9, Rn. 7.

auf solche Softwarekomponenten erstrecken würde, die ohne eine urheberrechtliche Zustimmung eines anderen Rechtsinhabers genutzt werden dürfen.

Unproblematisch ist auch hier ein beschränktes Copyleft, wie es bei der Mozilla Public License und ähnlichen Lizenzen anzutreffen ist. Da hier das Copyleft auf Änderungen innerhalb einer Datei beschränkt ist, wird nahezu immer auch eine zustimmungsbedürftige Bearbeitung vorliegen. Das Copyleft reicht damit nicht über das urheberrechtliche Zustimmungserfordernis hinaus.

Einige Open Source-Lizenzen stellen für die Reichweite des Copylefts auf das Vorliegen einer Bearbeitung ab. So definiert die Common Public License[34] eine „Contribution", die unter das Copyleft fällt, über den Begriff „derivative work". Dies gilt ebenso für die Open Software License.[35] Hier wird auf den urheberrechtlichen Umarbeitungsbegriff in der oben entwickelten Form abzustellen sein.[36] Dies bedeutet, dass eigenständige Softwarekomponenten regelmäßig nicht unter das Copyleft fallen, wenn sie über eine vorgegebene Schnittstelle angebunden werden und keine spezielle Anpassung des Ursprungsprogramms erfolgt ist. Anderweitige Änderungen und Hinzufügungen müssen hingegen der Ursprungslizenz unterstellt werden.

Die GPL enthält in Version 2 eine eigene Auslegung dessen, was als „derivative work" anzusehen ist und kann damit über den Umarbeitungsbegriff des § 69 c Nr. 2 UrhG hinausgehen. So findet nach Ziffer 2 GPLv2 das Copyleft auch Anwendung, wenn an sich eigenständige Komponenten zusammen mit einem GPL-Bestandteil als Teile eines Ganzen vertrieben werden.[37] Allerdings wird dann eine zustimmungspflichtige Werkverbindung vorliegen, für die hinsichtlich des gemeinsamen Vertriebs auch die Lizenzierung unter der GPL verlangt werden kann. Wer eigenständige Komponenten, die keinen GPL-Code enthalten, aber über eine Schnittstelle angebunden werden können, alleine verwertet, wird dies machen können, ohne diese Komponente unter der GPL lizenzieren zu müssen. Dabei wird man aber nicht auf den alleinigen Vertrieb, sondern auf die alleinige Verwertung abzustel-

---

34 Abrufbar unter http://www.eclipse.org/legal/cpl-v10.html.

35 Abrufbar unter http://opensource.org/licenses/osl-3.0.php.

36 Zu berücksichtigen sind allerdings noch Fragen des IPR bei der Auslegung, vgl. *Jaeger/Metzger*, Open Source Software – Rechtliche Rahmenbedingungen der Freien Software, 2. Aufl., 2006, Rn. 366.

37 *"But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it."*

len haben, so dass der Umgehung des Copylefts Grenzen gesetzt sind. Der Vertrieb von verbundenen Softwarekomponenten auf zwei Datenträgern an einen Empfänger dürfte dann noch als gemeinsame Verwertung anzusehen sein, die vom Copyleft erfasst wird.

Damit kann für die genannten Open Source-Lizenzen konstatiert werden, dass die Copyleft-Klauseln nur insoweit eingreifen als auch eine urheberrechtliche Erlaubnis für die gemeinsame Verwertung erforderlich ist.

VII. Zusammenfassung

Die kommerzielle Nutzung von Freier Software ist zulässig und Realität in der modernen Softwarewirtschaft. Das Lizenzgebührenverbot hat eine Reihe von Geschäftsmodellen gefördert, die auf Support oder den Verkauf der Software ausgelegt sind. Großer Beliebtheit erfreut sich insbesondere die Kombination von Softwarebestandteilen unter verschiedenen Lizenzbedingungen.

Hier stellt sich bei Copyleft-Lizenzen die Frage nach der Kompatibilität der betroffenen Lizenzbedingungen und in diesem Zusammenhang auch die Frage nach den Grenzen des Umarbeitungsrechts und der Werkverbindung. Die Untersuchung hat gezeigt, dass entsprechend der Computerprogramm-Richtlinie auf verwendete Schnittstellen als relevantes Abgrenzungskriterium abzustellen ist. Damit bedarf es auch dann keiner urheberrechtlichen Zustimmung, wenn für ein Programm ein Softwaremodul entwickelt wird, das über eine Komponentenschnittstelle angebunden wird, ohne dass zu diesem Zweck das Programm selbst angepasst werden muss. Bei der Vielzahl der Programmiertechniken werden sich zwar eine Reihe von Abgrenzungsfragen stellen, dennoch ist die Schnittstelle als Bezugspunkt das einzige hinreichend objektivierbare Kriterium, das auch mit der Computerprogramm-Richtlinie vereinbar ist.

Werden Open Source-Programme mit hinzugefügten Softwarekomponenten zur gemeinsamen Verwertung verbunden, hängt es von der einschlägigen Open Source-Lizenz ab, inwieweit durch eine Copyleft-Klausel Lizenzierungsvorgaben für die gesamte Software entstehen. Im Rahmen der dabei erforderlichen vertraglichen Vereinbarung kann die gemeinsame Verwertung von der Open Source-Lizenzierung der hinzugefügten Komponenten abhängig gemacht werden. Jeder Rechtsinhaber kann seine eigenen Komponenten alleine auch weiterhin unter Bedingungen seiner Wahl lizenzieren.