Schriften des Zentrums für angewandte Rechtswissenschaft, Universität Karlsruhe (TH)
7

Fabian Schäfer

Der virale Effekt

Entwicklungsrisiken im Umfeld von Open Source Software

universitätsverlag karlsruhe

Fabian Schäfer

Der virale Effekt

Entwicklungsrisiken im Umfeld von Open Source Software

Schriften des Zentrums für angewandte Rechtswissenschaft

Band 7

ZAR │ Zentrum für angewandte Rechtswissenschaft

Universität Karlsruhe (TH)

Herausgeber der Schriftenreihe: *Prof. Dr. Thomas Dreier M.C.J.*

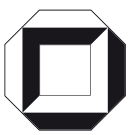
Prof. Dr. Jürgen Kühling LL.M.

Prof. Dr. Peter Sester Dipl.-Kfm.

Der virale Effekt

Entwicklungsrisiken im Umfeld von Open Source Software

von
Fabian Schäfer



universitätsverlag karlsruhe

Dissertation an der Universität Freiburg i. Br., Rechtswissenschaftliche Fakultät

Dekan der Rechtswissenschaftlichen Fakultät Freiburg i. Br.:
Prof. Dr. jur. Andreas Voßkuhle
Erstgutachter:
Prof. Dr. jur. Thomas Dreier, MCJ, Institut für Informationsrecht,
Universität Karlsruhe (TH)
Zweitgutachter:
Prof. Dr. jur. Maximilian Haedicke, LLM, Institut für Wirtschaftsrecht -
Arbeitsbereich Gewerblicher Rechtsschutz und Urheberrecht,
Universität Freiburg i. Br.

Ort und Tag der mündlichen Prüfung:
Freiburg i. Br., am 23. Mai 2007

Die Dissertation wurde am Institut für Informationsrecht der Universität Karlsruhe (TH) geschrieben. Stand Januar 2007.

Impressum

Universitätsverlag Karlsruhe
c/o Universitätsbibliothek
Straße am Forum 2
D-76131 Karlsruhe
www.uvka.de

Universitätsverlag Karlsruhe 2007
Print on Demand

ISSN: 1860-8744
ISBN: 978-3-86644-141-5

Vorwort

Die vorliegende Arbeit wurde im Wintersemester 2006/07 von der juristischen Fakultät der Albert-Ludwigs-Universität Freiburg als Dissertation angenommen. Sowohl Rechtssprechung als auch Literatur wurden bis Januar 2007 berücksichtigt. Bedanken möchte ich mich im Besonderen bei meinem Doktorvater, Herrn Prof. Dr. Thomas Dreier, M.C.J. dafür, dass er eine solch „techniklastige“ juristische Arbeit gefördert und mich bei der Erstellung umfassend unterstützt hat. Mein Dank gilt auch Herrn. Prof. Dr. Maximilian Haedicke, LL.M. für die rasche Erstellung des Zweitgutachtens.

Die Arbeit widme ich meiner Frau Nadja, die trotz der zahlreichen Entbehrungen mir mit unendlicher Geduld stets zur Seite stand und meiner Tochter Emma Louise, deren Lächeln mir jederzeit die nötige Leichtigkeit zurückgegeben hat.

Inhaltsübersicht

Inhaltsverzeichnis

Literaturverzeichnis

Achilles, Albrecht, *Betriebssysteme : eine kompakte Einführung mit Linux*, Berlin u.a. 2005.

Andréewitch, Markus, *Open Source und proprietäre Software: Das Verknüpfungsproblem* in: MR, 2005, Seite 240 ff.

Andréewitch, Markus, *Risiken bei firmen- und behördeninterner Bearbeitung von Open Source Software*, in: MR, 2005, Seite 36 ff.

Bamberger, Heinz Georg / Roth, Herbert, *Kommentar zum Bürgerlichen Gesetzbuch Elektronische Ressource*, München 2004.

Bartosz, Sujecki, *Open Source Software im deutschen Vertrags- und Urheberrecht*, in: MR, 2005, Seite 40 ff.

Bartsch, Michael, *Weitergabeverbote in AGB-Verträgen zur Überlassung von Standardsoftware*, in: CR, 1987, Seite 8 ff.

Baus, Christoph A., *Umgehung der Erschöpfungswirkung durch Zurückhaltung von Nutzungsrechten?*, in: MMR, 2002, Seite 14 ff.

Bechtold, Stefan, *Der Schutz des Anbieters von Information - Urheberrecht und Gewerblicher Rechtsschutz im Internet*, in: ZUM, 1997, Seite 427 ff.

Berelecon Research, *FLOSS Final Report – Part 2 Free/Libre Open Source Software: Survey and Study*, http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf.

Berger, Christian, *Die Erschöpfung des urheberrechtlichen Verbreitungsrechts als Ausprägung der Eigentumstheorie des BGB*, in: AcP, 2001, Seite 411 ff.

Berthold, Friedrich Joseph, *Filmrecht: ein Handbuch*, München 1957.

Bohr, Kurt, *Die Urheberrechtsbeziehungen der an der Filmherstellung Beteiligten*, Berlin 1978.

Brause, Rüdiger, *Kompendium der Informationstechnologie Hardware, Software, Client-Server-Systeme, Netzwerke, Datenbanken*, Berlin Heidelberg u.a. 2005.

Bydlinski, Peter, *Der Sachbegriff im elektronischen Zeitalter: zeitlos oder anpassungsbedürftig?*, in: AcP, 1998, Seite 287 ff.

Dadam, Peter, *Verteilte Datenbanken und Client/Server-Systeme : Grundlagen, Konzepte und Realisierungsformen*, Berlin u.a. 1996.

Dauner-Lieb, Barbara / Konzen, Horst / Schmidt, Karsten, *Das neue Schuldrecht in der Praxis: Akzente, Brennpunkte, Ausblick*, Köln u.a. 2003.

David, Paul A. / Waterman, Andrew / Arora, Seema, *FLOSS-US The Free/Libre/Open Source Software Survey for 2003*, als Online-Ausgabe unter http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf.

Deike, Thies, *Open-Source-Software: IPR-Fragen und Einordnung ins deutsche Rechtssystem*, in: CR, 2003, Seite 9 ff.

Determann, Lothar, *Softwarekombinationen unter der GPL*, in: GRUR Int., 2006, Seite 645 ff.

Diedrich, Kay, *Typisierung von Softwareverträgen nach der Schuldrechtsreform*, in: CR, 2002, Seite 473 ff.

Diedrich, Oliver, *In den Tiefen des Kernel Interview mit dem Linux-Entwickler Alan Cox*, als Online-Ausgabe unter http://www.heise.de/ct/99/25/034/default.shtml.

Diedrich, Oliver. *Streit um die neue GPL*, als Online-Ausgabe unter http://www.heise.de/open/artikel/78967.

Dieselhorst, Jochen, *Anwendbares Recht bei internationalen Online-Diensten*, in: ZUM, 1998, Seite 293 ff.

Dörner, Heinrich, *Anmerkung zu: BGH Entscheidung vom 09.10.1986 - II ZR 241/85*, in: JR 1987, Seite 201.

Dreier, Thomas, *Verletzung urheberrechtlich geschützter Software nach der Umsetzung der EG Richtlinie*, in: GRUR, 1993, Seite 781 ff.

Dreier, Thomas / Schulze, Gernot, *Urheberrechtsgesetz Urheberrechtswahrnehmungsgesetz, Kunsturhebergesetz ; Kommentar*, 2. Aufl., München 2006.

Endler, Maximilian / Daub, Jan, *Internationale Softwareüberlassung und UN-Kaufrecht*, in: CR, 1993, Seite 601 ff.

Ensthaler, Jürgen / Bosch, Wolfgang / Völker, Stefan, *Handbuch Urheberrecht und Internet*, Heidelberg 2002.

Erman, Walter, *Handkommentar zum Bürgerlichen Gesetzbuch in 2 Bänden*, 11., neubearb. Aufl., Münster 2004.

Ferid, Murad, *Internationales Privatrecht d. neue Recht ; e. Leitf. für Praxis u. Ausbildung*, 3., völlig neubearb. Aufl., Frankfurt am Main 1986.

Free Software Foundation, *GPL v3 Second Discussion Draft Rationale*, http://gplv3.fsf.org/rationale.

Free Software Foundation, *Opinion on Denationalization of Terminology*, http://gplv3.fsf.org/denationalization-dd2.pdf.

Frielingsdorf, Herbert, *Einfache IT-Systeme*, 4. Aufl., Troisdorf 2006.

Ganten, Peter H. / Alex, Wulf, *Debian GNU, Linux : Grundlagen, Installation, Administration und Anwendung ; mit 10 Tabellen ; Power Pack mit SARGE*, Nachdr. der 2., überarb. Aufl. 2004, Berlin u.a. 2005.

Gaster, Jens, *Urheberrecht und verwandte Schutzrechte in der Informationsgesellschaft*, in: ZUM, 1995, Seite 740 ff.

Geisler, Stephan, *Die engste Verbindung im internationalen Privatrecht*, Berlin 2001.

Grassmuck, Volker, *Freie Software Zwischen Privat- und Gemeineigentum*, als Online-Ausgabe unter http://freie-software.bpb.de/Grassmuck.pdf.

Gruber, Joachim, *Auslegungsprobleme bei fremdsprachigen Verträgen unter deutschem Recht*, in: DZWir, 1997, Seite 353 ff.

Grützmacher, Malte, *Anmerkung zu LG MannheimUrteil vom 27.06.2003 – 7 O 127/*03, in: CR 2004, Seite 814 ff.

Grützmacher, Malte, *Open-Source-Software - die GNU General Public License*, in: ITRB, 2002, Seite 84 ff.

Gulbins, Jürgen / Obermayr, Karl, *AIX UNIX : System V.4 ; Begriffe, Konzepte, Kommandos*, Berlin u.a. 1996.

Herold, Helmut, *Linux-Unix-Systemprogrammierung*, 3., aktualisierte Aufl., München u.a. 2004.

Heuer, Konrad, *Unix-Systemadministration Linux, Solaris, AIX, FreeBSD, Tru64-UNIX; mit 77 Tabellen*, Berlin, Heidelberg u.a. 2004.

Heussen, Benno, *Rechtliche Verantwortungsebenen und dingliche Verfügungen bei der Überlassung von Open Source Software*, in: MMR, 2004, Seite 445 ff.

Hilty, Reto M., *Der Softwarevertrag - ein Blick in die Zukunft - Konsequenzen der trägerlosen Nutzung und des patentrechtlichen Schutzes von Software*, in: MMR, 2003, Seite 3 ff.

Hilty, Reto M. / Peukert, *Interessenausgleich im Urheberrecht*, 1. Aufl., Baden-Baden 2004.

Hoeren, Thomas, *Anmerkung zu: LG München I, Urteil vom 19.05. 2004 – 21 O 6123/04*, in: CR 2004, Seite 776 ff.

Hoeren, Thomas, *Internet und Recht - Neue Paradigmen des Informationsrechts*, in: NJW, 1998, Seite 2849 ff.

Hoeren, Thomas*, Open Source und das Schenkungsrecht - eine durchdachte Liaison?,* in: Bork, Reinhard; Hoeren, Thomas; Pohlmann, Petra, Recht und Risiko : Festschrift für Helmut Kollhosser zum 70. Geburtstag Karlsruhe 2004.

Hoeren, Thomas, *Softwareüberlassung als Sachkauf : ausgewählte Rechtsprobleme des Erwerbs von Standardsoftware*, München 1989.

Hoeren, Thomas / Sieber, Ulrich, *Handbuch Multimedia-Recht Rechtsfragen des elektronischen Geschäftsverkehrs*, München.

Hoffmann, Bernd V., *Verträge über gewerbliche Schutzrechte im Internationalen Privatrecht*, in: RabelsZ, 1976, Seite 208 ff.

Ifross, *Die GPL kommentiert und erklärt*, Dt. Orig.-Ausg., 1. Aufl., Köln u.a. 2005.

Jaeger, Till, *Eine Lizenz entsteht*, als Online-Ausgabe unter http://www.heise.de/open/artikel/76248.

Jaeger, Till / Metzger, Axel, *Neues Recht für freie Software - GPLv3 - Gesetzgebung in Vertragsform*, in: c't, 2006, Seite 46 ff.

Jaeger, Till / Metzger, Axel, *Open Source Software Rechtliche Rahmenbedingungen der Freien Software*, 2. Aufl., München 2006.

Junker, Abbo / Benecke, Martina, *Computerrecht*, 3. Aufl., Baden-Baden 2002.

Katzenberger*, Urheberrechtsverträge im internationalen Privatrecht und Konventionsrecht,* in: Beier, Friedrich-Karl; Götting, Horst-Peter; Lehmann, Michael; Moufang, Rainer, Urhebervertragsrecht: Festgabe für Gerhard Schricker zum 60. Geburtstag, München 1995.

Kegel, Gerhard / Schurig, Klaus, *Internationales Privatrecht ein Studienbuch*, 9., neubearb. Aufl., München 2004.

Klima, Robert / Selberherr, Siegfried, *Programmieren in C*, Wien u.a. 2003.

Klimek, Oliver Alexander / Sieber, Stefanie, *Anwendbares Recht beim Vertrieb digitalisierbarer Waren über das Internet am Beispiel der Softwareüberlassung*, in: ZUM, 1998, Seite 902 ff.

Koch, Frank A., *Begründung und Grenzen des urheberrechtlichen Schutzes objektorientierter Software*, in: GRUR 2000, Seite 191 ff.

Koch, Frank A., *Computer-Vertragsrecht umfassende Erläuterungen, Beispiele und Musterformulare für Erwerb und Nutzung von EDV-Systemen*, 6., völlig neu bearb. Aufl., Freiburg i. Br. u.a. 2002.

Koch, Frank A., *Das neue Softwarerecht und die praktischen Konsequenzen*, in: NJW-CoR, 1994, Seite 293 ff.

Koch, Frank A., *Grundlagen des Urheberrechtsschutz im Internet und in Online-Diensten*, in: GRUR, 1997, Seite 417 ff.

Koch, Frank A., *Internationale Gerichtszuständigkeit und Internet*, in: CR, 1999, Seite 121 ff.

Koch, Frank A., *Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software*, in: CR, 2000, Seite 273 ff.

Koch, Frank A., *Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software*, in: CR, 2000, Seite 333 ff.

Koch, Frank A., *Urheberrechtliche Zulässigkeit technischer Beschränkungen und Kontrolle der Software-Nutzung*, in: CR, 2002, Seite 629 ff.

Koch, Frank A., *Urheberschutz für das Customizing von Computerprogrammen*, in: ITRB, 2005, Seite 140 ff.

Köhntopp, Kristian / Köhntopp, Marit / Pfitzmann, Andreas, *Sicherheit durch Open Source? - Chancen und Grenzen*, in: DuD, 2000, Seite 508 ff.

König, Mark M., *Das Computerprogramm im Recht technische Grundlagen, Urheberrecht und Verwertung, Überlassung und Gewährleistung*, Köln 1991.

Kotthoff, Jost, *Zum Schutz von Datenbanken beim Einsatz von CD-ROMs in Netzwerken*, in: GRUR, 1997, Seite 597 ff.

Kropholler, Jan, *Internationales Privatrecht einschließlich der Grundbegriffe des Internationalen Zivilverfahrensrechts*, 6., neubearb. Aufl., Tübingen 2006.

Küng, Julia, *Urheberrechtliche Aspekte Freier Software*, in: MR, 2004, Seite 21 ff.

Lange, Knut Werner, *Virtuelle Unternehmen : neue Unternehmenskoordinationen in Recht und Praxis*, Heidelberg 2001.

Lehmann, Michael, *Internet- und Multimediarecht (Cyberlaw)*, Stuttgart 1997.

Lehmann, Michael, *Rechtsschutz und Verwertung von Computerprogrammen : Urheberrecht, Patentrecht, Warenzeichenrecht, Wettbewerbsrecht, Kartellrecht, Vertrags- und Lizenzrecht, Arbeitsrecht, Strafrecht, Insolvenz- und Vollstreckungsrecht, Prozeßrecht, Produzentenhaftung, Recht der Datenbanken, Steuerrecht*, 2., völlig überarb. und erw. Aufl., Köln 1993.

Lejeune, Mathias, *Rechtsprobleme bei der Lizenzierung von Open Source Software nach der GNU GPL*, in: ITRB, 2003, Seite 10 ff.

Lenhard, Frank, *Vertragstypologie von Softwareüberlassungsverträgen Neues Urhebervertragsrecht und neues Schuldrecht unter Berücksichtigung der Open Source-Softwareüberlassung*, München 2006.

Loewenheim, Ulrich, *Handbuch des Urheberrechts*, München 2003.

Lurger, Brigitta, Internationaler Verbraucherschutz im Internet, in: Leible, Stefan, Recht und Neue Medien Stuttgart u.a. 2003.

Mäger, Stefan, *Der urheberrechtliche Erschöpfungsgrundsatz bei der Veräußerung von Software*, in: CR, 1996, Seite 522 ff.

Mankowski, Peter, *Das Internet im Internationalen Vertrags- und Deliktsrecht*, in: RabelsZ, 1999, Seite 203 ff.

Marly, Jochen, *Softwareüberlassungsverträge Erscheinungsformen, Pflichverletzungen, Vertragsgestaltung, Allgemeine Geschäftsbedingungen, Musterverträge, Text-CD-ROM*, 4., vollst. überarb. und erw. Aufl., München 2004.

Marly, Jochen, *Urheberrechtsschutz für Computersoftware in der Europäischen Union : Abschied vom überkommenen Urheberrechtsverständnis*, München 1995.

Masak, Dieter, *Legacysoftware : das lange Leben der Altsysteme ; mit 39 Tabellen*, Berlin u.a. 2006.

Mauerer, Wolfgang, *Linux Kernelarchitektur : Konzepte, Strukturen und Algorithmen von Kernel 2.6*, München u.a 2004.

Metzger, Axel / Jaeger, Till, *Open Source Software und deutsches Urheberrecht*, in: GRUR Int., 1999, Seite 839 ff.

Möhring, Philipp / Nicolini, *Urheberrechtsgesetz Elektronische Ressource Kommentar*, 2. Aufl., München 2000.

Moritz, Hans-Werner, *Vervielfältigungsstück eines Programms und seine berechtigte Verwendung - § 69d UrhG und die neueste BGH-Rechtsprechung*, in: MMR, 2001, Seite 94 ff.

Nordemann, Wilhelm / Fromm, Friedrich Karl, *Urheberrecht Kommentar zum Urheberrechtsgesetz und zum Urheberrechtswahrnehmungsgesetz ; mit den Texten der Urheberrechtsgesetze Österreichs und der Schweiz*, 9., überarb. und erg. Aufl., Stuttgart u.a. 1998.

Omsels, Hermann-Josef*, Open Source und das deutsche Vertrags- und Urheberrecht,* in: Schertz, Christian; Omsels, Hermann-Josef, Festschrift für Paul W. Hertin zum 60. Geburtstag, München 2000.

Palandt, Otto, *Bürgerliches Gesetzbuch mit Einführungsgesetz (Auszug), BGB-Informationspflichten-Verordnung, Unterlassungsklagengesetz, Produkthaftungsgesetz, Erbbaurechtsverordnung, Wohnungseigentumsgesetz, Hausratsverordnung, Lebenspartnerschaftsgesetz, Gewaltschutzgesetz (Auszug)*, 66., neu bearb. Aufl., München 2007.

Paulus, Christoph G., *Insolvenzverfahren, Sanierungsplan: Risiken und Vermeidungsstrategien*, in: CR, 2003, Seite 237 ff.

Plaß, Gunda, *Open Contents im deutschen Urheberrecht*, in: GRUR, 2002, Seite 670 ff.

Pres, Andreas, *Gestaltungsformen urheberrechtlicher Softwarelizenzverträge eine juristische und ökonomische Untersuchung unter besonderer Berücksichtigung des Zweiten Gesetzes zur Änderung des Urheberrechtsgesetzes vom 9. Juni 1993*, Köln 1994.

Raymond, Eric Steven, *The Cathedral and the Bazaar*, als Online-Ausgabe unter http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/.

Rebmann, Kurt / Säcker, Franz Jürgen / Rixecker, Roland,

München er Kommentar zum Bürgerlichen Gesetzbuch - Bd. 1. Allgemeiner Teil, 5. Aufl., München 2007

München er Kommentar zum Bürgerlichen Gesetzbuch - Bd. 3. Besonderes Schuldrecht I, 4. Aufl., München 2004

München er Kommentar zum Bürgerlichen Gesetzbuch - Bd. 10. Einführungsgesetz zum Bürgerlichen Gesetzbuche, Internationales Privatrecht, 4. Aufl., München 2006

Rehbinder, Manfred, *Urheberrecht: ein Studienbuch*, 14., neu bearb. Aufl., München 2006.

Reithmann, Christoph / Martiny, Dieter, *Internationales Vertragsrecht Das internationale Privatrecht der Schuldverträge*, 6. Aufl., Köln 2004.

Royla, Pascal / Gramer, Tobias, *Urheberrecht und Unternehmenskauf*, in: CR 2005, Seite 154 ff.

Ruh, William A. / Maginnis, Francis X. / Brown, William J., *Enterprise application integration : a Wiley tech brief*, New York u.a. 2001.

Sahin, Ali / Haines, Alexander, *Einräumung von Nutzungsrechten im gestuften Vertrieb von Standardsoftware*, in: CR, 2005, Seite 241 ff.

Schack, Haimo*, International zwingende Normen im Urhebervertragsrecht,* in: Lorenz, Stephan; Trunk, Alexander; Eidenmüller, Horst; Wendehorst, Christiane; Adolff, Johannes, Festschrift für Andreas Heldrich : zum 70. Geburtstag, München 2005.

Schack, Haimo, *Internationale Urheber-, Marken- und Wettbewerbsrechtsverletzungen im Internet*, in: MMR, 2000, Seite 59 ff.

Schack, Haimo, *Urheber- und Urhebervertragsrecht*, 3., neu bearb. Aufl., Tübingen 2005.

Schack, Haimo, *Urheberrechtsverletzung im internationalen Privatrecht Aus der Sicht des Kollisionsrechts*, in: GRUR Int., 1985, Seite 523 ff.

Schiffner, Thomas, *Open Source Software Freie Software im deutschen Urheber- und Vertragsrecht*, 1. Aufl., Bad Feilnbach 2003.

Schlechtriem, Peter H., *Schuldrecht Besonderer Teil*, 6., weitgehend neubearb. Aufl., Tübingen 2003.

Schmaranz, Klaus, *Softwareentwicklung in C++*, Berlin u.a. 2003.

Schneider, Jochen, *Handbuch des EDV-Rechts IT-Vertragsrecht, Rechtsprechung, AGB, Vertragsgestaltung, Datenschutz, Rechtsschutz*, 3., umfassend überarb. u. erw. Aufl., Köln 2003.

Schnitzer, Adolf F.*, Die funktionelle Anknüpfung im internationalen Vertragsrecht,* in: Universitätsverl., Festgabe für Wilhelm Schönenberger zum 70. Geburtstag am 21. September 1968, Freiburg (Schweiz) 1968.

Schønning, Peter, *Anwendbares Recht bei grenzüberschreitenden Direktübertragungen*, in: ZUM, 1997, Seite 34 ff.

Schricker, *Urheberrecht, Kommentar Zweites Gesetz zur Regelung des Urheberrechts in der Informationsgesellschaft*, 3., neubearb. Aufl. 2006, München 2006.

Schricker, Gerhard, *Urheberrecht Kommentar*, 2., neubearb. Aufl., München 1999.

Schuhmacher, Dirk, *Wirksamkeit von typischen Klauseln in Softwareüberlassungsverträgen*, in: CR, 2000, Seite 641 ff.

Schulz, Carsten, *Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte Vertragsstrukturen in Open Source Modellen*, Köln u.a. 2005.

Sester, Peter, *Open-Source-Software: Vertragsrecht, Haftungsrisiken und IPR-Fragen*, in: CR, 2000, Seite 797 ff.

Siefert, Torsten, *Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht und deren Bedeutung für das Verwertungsrecht*, 1. Aufl., Baden-Baden 1998.

Soergel, Hans Th., *Bürgerliches Gesetzbuch mit Einführungsgesetz und Nebengesetzen ; Kohlhammer-Kommentar*, 13., völlig neu bearb. u. erw. Aufl., Stuttgart u.a.

Spindler, Gerald, *Morpheus, Napster & Co. - Die kollisionsrechtliche Behandlung von Urheberrechtsverletzungen im Internet*, in: Leible, Stefan, Die Bedeutung des Internationalen Privatrechts im Zeitalter der neuen Medien, 1. Aufl., Stuttgart u.a. 2003.

Spindler, Gerald, *Die kollisionsrechtliche Behandlung von Urheberrechtsverletzungen im Internet*, in: IPrax, 2003, Seite 412 ff.

Spindler, Gerald, *Europäisches Urheberrecht in der Informationsgesellschaft*, in: GRUR, 2002, Seite 105 ff.

Spindler, Gerald, *Open Source Software auf dem gerichtlichen Prüfstand - Dingliche Qualifikation und Inhaltskontrolle*, in: K&R, 2004, Seite 528 ff.

Spindler, Gerald, *Rechtsfragen bei Open Source*, Köln 2004.

Spindler, Gerald, *Rechtsfragen der Open Source Software*, Gutachten im Auftrag des Verbands der Softwareindustrie Deutschland e.V., als Online-Ausgabe unter www.vsi.de/inhalte/aktuell/studie_final_safe.pdf.

Spindler, Gerald*, Miturhebergemeinschaft und BGB-Gesellschaft,* in: Beier, Friedrich-Karl; Götting, Horst-Peter; Lehmann, Michael; Moufang, Rainer, Urhebervertragsrecht Festgabe für Gerhard Schricker zum 60. Geburtstag, München1995.

Spindler, Gerald / Wiebe, Andreas, *Open Source-Vertrieb Rechteeinräumung und Nutzungsberechtigung*, in: CR, 2003, Seite 873 ff.

Staudinger, Julius von, *J. von Staudingers Kommentar zum Bürgerlichen Gesetzbuch, EGBGB, Internationales Privatrecht : Einleitung zu Art. 27 ff EGBGB; Art. 27-33 EGBGB; Anhang zu Art. 33 EGBG: Internationales Factoring; Art. 34 EGBGB*, 13., bearb. Aufl., Berlin 2002.

Staudinger, Julius von / Großfeld, Bernhard, *Internationales Gesellschaftsrecht*, 14. Aufl., Berlin 1998.

Stichtenoth, Jonas, *Softwareüberlassungsverträge nach dem Schuldrechtsmodernisierungsgesetz*, in: K&R, 2003, Seite 105 ff.

Stickelbrock, Barbara, *Linux & Co - Gewährleistung und Haftung bei kommerziell vertriebener Open Source Software (OSS)*, in: ZGS, 2003, Seite 368 ff.

Stoll, Hans*, Rechtskollisionen bei Schuldnermehrheit,* in: Dieckmann, Albrecht; Frank, Rainer; Hanisch, Hans; Simitis, Spiros, Festschrift für Wolfram Müller-Freienfels, Baden-Baden 1986.

Stroh, Rolf, *Werkeinheit und Werkmehrheit im Urheberrecht*, München 1969.

Tanenbaum, Andrew S. / Baumgarten, Uwe, *Moderne Betriebssysteme*, 2., überarb. Aufl., Nachdr. /, München 2005.

Thum, Dorothee, *Das Territorialitätsprinzip im Zeitalter des Internet - Zur Frage des auf Urheberrechtsverletzungen im Internet anwendbaren Rechts*, in: Bartsch, Michael; Lutterbeck, Bernd, Neues Recht für neue Medien, Köln 1998.

Triebel, Volker / Balthasar, Stephan, *Auslegung englischer Vertragstexte unter deutschem Vertragsstatut - Fallstricke des Art. 32 I Nr. 1 EGBGB*, in: NJW, 2004, Seite 2189 ff.

Ulmer, Eugen, *Die Immaterialgüterrechte im internationalen Privatrecht : rechtsvergleichende Untersuchung mit Vorschlägen für die Vereinheitlichung in der Europäischen Wirtschaftsgemeinschaft*, Köln u.a. 1975.

Ulmer, Eugen, *Urheber- und Verlagsrecht*, 3., neu bearb. Aufl., Berlin u.a. 1980.

Waldenberger, Arthur, *Die Miturheberschaft im Rechtsvergleich: zugleich ein Beitrag zur Lehre von der Miturheberschaft nach deutschem Recht*, München 1991.

Waldenberger, Arthur, *Grenzen des Verbraucherschutzes beim Abschluss von Verträgen im Internet*, in: BB, 1996, Seite 2365 ff.

Waldenberger, Arthur, *Zur zivilrechtlichen Verantwortlichkeit für Urheberrechtsverletzungen im Internet*, in: ZUM, 1997, Seite 176 ff.

Wallner, Jürgen, *Die Insolvenz des Urhebers unter besonderer Berücksichtigung der Verträge zur Überlassung von Software*, Berlin 2002.

Wallner, Jürgen, *Softwarelizenzen in der Insolvenz des Lizenzgebers*, in: ZIP, 2004, Seite 2073 ff.

Wandtke, Artur-Axel / Bullinger, Winfried, *Praxiskommentar zum Urheberrecht Elektronische Ressource*, 1. Aufl., München 2002.

Weber, Rolf H., *Freie Software - Befreiung vom Vertragstypenkonzept,* in: Harrer, Friedrich; Portmann, Wolfgang; Zäch, Roger, Besonderes Vertragsrecht - aktuelle Probleme: Festschrift für Heinrich Honsell zum 60. Geburtstag, Zürich u.a. 2002.

Weitnauer, Wolfgang, *Der Vertragsschwerpunkt*, Frankfurt am Main 1981.

Wiebe, Andreas / Heidinger, Roman, *GPL 3.0 und EUPL*, in: MR, 2006, Seite 258 ff.

Wuermeling, Ulrich / Deike, Thies, *Open Source Software: eine juristische Risikoanalyse*, in: CR, 2003, Seite 87 ff.

Zahrnt, Christoph, *Überlassung von Software Produkten: Stand 1996*, in: NJW, 1996, Seite 1798 ff.

Zahrnt, Christoph, *Überlassung von Softwareprodukten nach neuem Urheberrecht*, in: CR, 1994, Seite 455 ff.

Abkürzungsverzeichnis

A

a.A.	anderer Ansicht
a.E.	am Ende
a.F.	alte Fassung
Abb.	Abbildung
ABI	Application Binary Interface
ABl.	Amtsblatt
AblEG	Amtsblatt der Europäischen Gemeinschaft
Abs.	Absatz
AcP	Archiv für civilistische Praxis
AGB	Allgemeine Geschäftsbedingungen
AGBG	Gesetz zur Regelung des Rechts der Allgemeinen Geschäftsbedingungen
Alt.	Alternative
amtl.	amtlich
amtl. Begr.	amtliche Begründung
Anh.	Anhang
Anm.	Anmerkung
API	Application Program Interface
Art.	Artikel
ASP	Application Service Providing
Aufl.	Auflage

B

BB	Der Betriebs-Berater
Bd.	Band
Begr.	Begründung
BGB	Bürgerliches Gesetzbuch
BGBl.	Bundesgesetzblatt
BGH	Bundesgerichtshof

BGHZ	Entscheidungen des Bundesgerichtshofs in Zivilsachen
BSD	Berkeley Software Distribution
bspw.	beispielsweise
BT-Drucks.	Drucksachen des Deutschen Bundestages
bzgl.	bezüglich
bzw.	beziehungsweise

C

CISG	United Nation Convention on Contracts für the International Sale of Goods
CPU	Central Processing Unit
CR	Computer und Recht
CVS	Concurrent Versions System

D

d.h.	das heißt
ders.	derselbe
DIN	Deutsches Institut für Normung
DNS	Domain Name Service
Dok.	Dokument
DRM	Digital Rights Management
DuD	Datenschutz und Datensicherheit. Recht und Sicherheit in Informationsverarbeitung und Kommunikation
DZWir	Die Deutsche Zeitschrift für Wirtschafts- und Insolvenzrecht

E

e.V.	eingetragener Verein
EDV	Elektronische Datenverarbeitung

EG	Europäische Gemeinschaften
Einf.	Einführung
Einl.	Einleitung
etc	et cetera
EU	Europäische Union
EVÜ	Übereinkommen über das auf vertragliche Schuldverhältnisse anzuwendende Recht
EWG	Europäische Wirtschaftsgemeinschaft

F

f / ff	folgende (Seite) / folgende (Seiten)
FAQ	Frequently Asked Questions
Festg.	Festgabe
Fn.	Fußnote
Fraunhofer ISI	Fraunhofer-Institut für Systemtechnik und Innovationsforschung
FS	Festschrift
FSF	Free Software Foundation

G

gem.	gemäß
ggf.	gegebenenfalls
GNU	Gnu's not UNIX
GPL	General Public License
GRUR	Gewerblicher Rechtsschutz und Urheberrecht
GRUR Int.	Gewerblicher Rechtsschutz und Urheberrecht, Internationaler Teil
GRUR-RR	Gewerblicher Rechtsschutz und Urheberrecht – Rechtsprechungs-Report

H

h.M.	herrschende Meinung
Hdb.	Handbuch
Hrsg.	Herausgeber

I

i.d.R.	in der Regel
i.E.	im Ergebnis
i.S.d.	im Sinne des/ im Sinne der
i.S.v.	im Sinne von
i.V.m.	in Verbindung mit
ifross	Institut für Rechtsfragen der Freien und Open Source Software
IIC	Interational Review of Intellectual Property and Competition Law
insb.	insbesondere
InsO	Insolvenzordnung
IPR	Internationales Privatrecht
ITRB	Der IT-Rechtsberater

J

JR	Juristische Rundschau
JurPC	JurPC – Internet-Zeitschrift für Rechtsinformatik
JZ	Juristische Zeitung

K

K&R	Kommunikation und Recht
Kap.	Kapitel
KG	Kammergerich
KOM	Dokumente der Kommission der Europäischen Gemeinschaften

L

LG	Landgericht
LGPL	Lesser General Public License / Library General Public License
lit.	litera
LUG	Gesetz vom 19.06.1901 betreffend das Urheberrecht an Werken der Literatur und der Tonkunst

M

m.w.N.	mit weiteren Nachweisen
MDR	Monatsschrift für deutsches Recht
MIT	Massachusetts Institute of Technology
MMR	Multimedia und Recht
MPL	Mozilla Public License
MR	Medien und Recht
Müko	Münchener Kommentar

N

n.F.	neue Fassung
NJW	Neue Juristische Wochenschrift
NJW-RR	NJW-Rechtsprechungs-Report Zivilrecht
Nr.	Nummer

O

o.ä.	oder ähnliches
o.g.	oben genannte
OEM	Original Equipment Manufacturer
OLG	Oberlandesgericht

OSD	Open Source Definition
OSI	Open Source Initiative
P	
PERL	Practical Evaluation and Reporting Language
R	
RBÜ	Revidierte Berner Übereinkunft zum Schutz von Werken der Literatur und Kunst
RegE	Regierungsentwurf
RegE	Regierungsentwurf
RIW	Recht der internationalen Wirtschaft
RL	Richtlinie
RL	Richtlinie
Rn.	Randnummer
Rspr.	Rechtsprechung
S	
S.	Seite
Sec.	Section
sog.	so genannte / so genannter
str.	streitig
T	
TCP/IP	Transmission Control Protocol / Internet Protocol
TRIPS	Agreement on Trade-related Aspects of Intellectual Property Rights in Counterfeit Goods

U

u.a.	unter anderem / und andere
U.S.	United States
U.S.C.	United States Code
u.U.	unter Umständen
UFITA	Archiv für Urheber-, Film-, Funk- und Theaterrecht
UrhG	Gesetz über Urheberrecht und die verwandten Schutzrechte (Urheberrechtsgesetz)
usw.	und so weiter

V

v.	von
vgl.	vergleiche
Vol.	Volume
VSI	Verband der Softwareindustrie Deutschland e.V.

W

WIPO	World Intellectual Property Organization

X

XML	Extensible Mark-up Language

Z

z.B.	zum Beispiel
ZGS	Zeitschrift für das gesamte Schuldrecht
ZIP	Zeitschrift für Wirtschaftsrecht
ZPO	Zivilprozessordnung
ZUM	Zeitschrift für Urheber- und Medienrecht

A. Einleitung

I.Einführung in die Thematik

Die Open-Source-Wirtschaft ist ihren Kinderschuhen entwachsen. Auch große internationale Unternehmen wie IBM oder Sun[1] setzen auf Entwicklungen dieser Branche und selbst Microsoft zeigt, nicht zuletzt durch seine Kooperation mit Novell,[2] dass Linux mittlerweile zu einer festen Größe avanciert ist.

Der europäische Open-Source-Markt ist laut Marktforscher IDC in den letzten Jahren bedeutend gewachsen. IDC schätzt das westeuropäische Marktvolumen auf € 75 Mio., mit einer Wachstumsprognose von € 175 Mio. im Jahr 2008.
Open Source Software findet mittlerweile immer mehr Anwendungsfelder und stellt immer öfter den wirtschaftlichen Kernbereich von Unternehmen dar. So hat eine aktuelle Studie des Fraunhofer Instituts gezeigt, dass für immerhin rund 55 % der befragten Unternehmen Open Source Software eine mittlere bis sehr große Rolle spielt und mindestens 25 % Umsatzanteil ausmacht.[3] Immerhin noch 18 % der Unternehmen gaben im Übrigen an, dass ihr Umsatz nahezu vollständig aus Open Source Dienstleistungen und – Produkten bestehen würde.
Viele Firmen sind mittlerweile dazu übergegangen kommerzielle Softwarelösungen im Open Source Bereich anzubieten.[4] Hierbei sei nur der Bereich der sog. value-added Produkte genannt. Nahezu unbemerkt von der Öffentlichkeit blieb häufig der Einsatz von Open Source Software im Markt der Embedded Devices. Dazu zählen Dinge, die jeder kennt und benutzt: Mobiltelefone, Router oder Switches, PDAs, Settop-Boxen für das digitale Fernsehen, aber auch viele Geräte in der Fertigungsindustrie, Telematik oder Kassensysteme im Einzelhandel.

Neben den gestiegenen Einsatzbereichen für Open Source Software ist in jüngster Zeit jedoch auch ein Trend unter den beteiligten Entwicklern zu erkennen, sich gegen Verletzungen der

[1] Sun hat aus diesem Grund erst jüngst zahlreichen Java-Applikationen unter der GPL veröffentlicht; Einzelheiten unter http://www.sun.com/software/opensource/java/faq.jsp#a.
[2] Vgl. http://www.heise.de/open/news/meldung/80458.
[3] Vgl. http://www.iao.fraunhofer.de/d/oss_studie.pdf, S. 45.
[4] Eine Studie des Fraunhofer Instituts hat bei einer Umfrage unter Unternehmen ergeben, dass 46 % der Unternehmen auf Basis von OSS-Technologien wie beispielsweise Linux, Apache, mySQL und PHP eigene proprietäre Anwendungen entwickelt, vgl. http://www.iao.fraunhofer.de/d/oss_studie.pdf, S. 45.

von ihnen verwendeten Lizenzen zu wehren.[5] Aufgrund dessen hatte sich das LG München bereits im Mai 2004[6] - weltweit erstmals – und jüngst das LG Berlin[7] sowie das LG Frankfurt[8] mit der GPL zu befassen.

Diesem Trend sind auch Schlagzeilen wie: „GPL-Verstöße: 13 Unternehmen auf der CeBIT im Visier“[9] zuzurechnen, die insbesondere wegen der Liste der betroffenen Unternehmen für öffentliche Aufmerksamkeit gesorgt haben. Diese reicht von Motorola über AOpen, Micronet und Buffalo bis hin zu Trendware, allesamt namhafte Hersteller im Bereich der Software- und Computerbranche.

Nicht zuletzt aufgrund dieses Trends müssen sich wohl alle Unternehmen, die der Open Source Branche zuzurechnen sind, künftig mehr als zuvor um die Einhaltung der maßgebenden Lizenzen sorgen, wenn sie eine wirtschaftliche Verwertbarkeit ihrer kommerziellen Produkte nachhaltig gewährleisten wollen.

Dies insbesondere vor dem Hintergrund, dass die im Open Source Bereich vorherrschende General Public License (GPL) grundsätzlich die kostenfreie Einräumung von Nutzungsrechten an Entwicklungen vorsieht, die von ihrem Wirkungsbereich erfasst werden.

So sollen Eigenentwicklungen im Bereich von Software, die unter der GPL lizenziert wurden, beispielsweise dann ebenfalls unter der GPL zu lizenzieren sein, wenn sie von dieser *„abgeleitet“* oder gegenüber dieser *„vernünftigerweise“* nicht als *„unabhängige und eigenständige Datenwerke“* zu betrachten sind,[10] die neben dem GPL-lizenzierten Teil der Software selbstständig bestehen.

Bereits bei einem ersten Blick auf diese Voraussetzungen der GPL wird ersichtlich, dass eine genaue Bestimmbarkeit ihres Wirkungsbereichs aufgrund der verwendeten allgemeinen Begriffe problematisch ist.

Was ist unter einem *„abgeleiteten“* Werk zu verstehen und ab wann hat man *„vernünftigerweise von unabhängigen und eigenständigen“* Werkteilen auszugehen?

Die Suche nach konkreten Abgrenzungskriterien gestaltet sich erwartungsgemäß schwierig.

---

[5] Zu diesem Zweck wurde das gpl-violations Projekt gegründet, vgl. http://www.gpl-violations.org; mit dem nun auch FSF-Europe im Rahmen der Freedom Task Force zusammenarbeitet. Erklärtes Ziel dieser Freedom Task Force ist sowohl die Verbreitung von Wissen über juristische Aspekte Freier Software aber auch die Durchsetzung der Lizenzen, vgl. http://www.germany.fsfeurope.org/projects/ftf/.

[6] LG München I CR 2004, 774.

[7] Beschluss des LG Berlin CR 2006, 735.

[8] LG Frankfurt/M CR 2006, 729.

[9] Vgl. http://www.golem.de/0503/36913.html.

[10] Dies folgt letztlich aus einem Umkehrschluss von Ziff. 2 GPL: *”...If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works.”*; Einzelheiten hierzu unter Kapitel 5, S. 115.

In der Literatur werden derzeit kaum konkrete Abgrenzungskriterien sondern lediglich eine Hand voll Indizien vorgeschlagen, anhand derer eine Abgrenzung stattfinden könnte. Im Übrigen wird vertreten, dass genaue Abgrenzungskriterien aufgrund des momentanen Stands der Rechtsprechung und wissenschaftlichen Ausarbeitungen zu diesem Thema nicht möglich wären.[11]

Die Probleme, die das Auffinden solcher Abgrenzungskriterien mit sich bringen, liegen auf der Hand. Um eine saubere Abgrenzung von Open Source Software und proprietärer Eigenentwicklung zu ermöglichen, ist es unerlässlich, eine an den technischen Gegebenheiten ansetzende Auslegung der GPL vorzunehmen. Nur wenn die technischen Zusammenhänge und Grundlagen zwischen Open Source Software und proprietärer Eigenentwicklung geklärt sind, kann sich eine juristische Bewertung anschließen. Auch die Free Software Foundation (FSF), als Herausgeberin der GPL, geht davon aus, dass maßgebliche Kriterien zur Abgrenzung nur in einer Zusammenschau von technischer Kommunikationsart und Inhalt der ausgetauschten Informationen liegen können.[12]

II.Gang der Untersuchung

Mit der vorliegenden Arbeit wird das Ziel verfolgt, den Rahmen der zulässigen wirtschaftlichen Nutzung von GPL-lizenzierten Programmen zu konkretisieren. Dabei geht es insbesondere um die Frage, unter welchen Voraussetzungen sich GPL-lizenzierte Programme von Unternehmen nutzen lassen, ohne dadurch von der Pflicht betroffen zu werden die eigenen Entwicklungen ebenfalls unter den Bestimmungen der GPL lizenzieren zu müssen. Es geht darum, den Umfang der durch die GPL vermittelten Pflichten und insbesondere die Reichweite des sog. viralen Effekts näher zu bestimmen.

Eine solche Bestimmung hat bislang nur in unzureichendem Maß stattgefunden, was nicht zuletzt darauf zurückzuführen ist, dass es derzeit sowohl an Entscheidungen aus der Rechtsprechung, als auch an wissenschaftlichen Ausarbeitungen fehlt, in denen die Reichweite der durch die GPL vermittelten Lizenzierungspflichten vertieft behandelt worden wäre.

---

[11] *Lejeune*, ITRB 2003, 10, 11.

[12] Auf der FAQ Seite der FSF äußert sich diese wie folgt: *„…This is a legal question, which ultimately judges will decide. We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged).“*; vgl. http://www.fsf.org/licensing/licenses/gpl-faq.html#MereAggregation .

Um in diesem Problembereich eine Konkretisierung zu ermöglichen, sind im Rahmen der vorliegenden Arbeit zunächst die rechtlichen Rahmenbedingungen zu untersuchen, die den Open Source Bereich prägen. Insofern ist zu klären, welche Rechtsgeschäfte dem Erwerb von Open Source Programmen zugrunde liegen, zwischen wem sie zustande kommen und nach welchem Recht sich die bestehenden Vertragsbeziehungen richten.

Nach einer allgemeinen Einführung zum Begriff der Open Source Software sowie einer Abgrenzung zu ähnlichen Verbreitungsformen (Kapitel 1), liegt der Ausgangspunkt der Untersuchung zunächst bei der Frage, wodurch sich die typischen Open Source Projekte aus urheberrechtlicher Sicht auszeichnen (Kapitel 2). Dabei wird vor allem auf die Besonderheiten eingegangen, die aus dem vorherrschenden dezentralen und kollaborativen Softwareentwicklungsprozess für die Urheberschaftsstrukturen folgen.
Die Urheberschaft an Open Source Software ist im Hinblick auf die Überlassung von Open Source Software vor allem dafür maßgeblich, von wem der Nutzer die Nutzungsrechte eingeräumt bekommen kann und gibt somit Aufschluss darüber, wer als Vertragspartner beim Abschluss der GPL in Betracht kommt.
Im Rahmen der Untersuchung knüpft sich eine Darstellung der vertragsrechtlichen Besonderheiten an (Kapitel 3), die den Erwerb von GPL-lizenzierter Open Source Software bestimmen, wobei vor allem geklärt wird, welche Rechtsgeschäfte dem Erwerb von Open Source Software zugrunde liegen und wer hieran beteiligt ist.

Bevor sich die Untersuchung sodann mit der Frage nach dem Umfang der durch die GPL vermittelten Bindung befassen kann, ist weiterhin zu klären, nach welchem Recht die dafür erforderliche Auslegung der GPL zu erfolgen hat (Kapitel 4). Gerade im Hinblick darauf, dass die Urheber von Open Source Programmen häufig nicht aus Deutschland kommen und der Erwerb der Programme, aufgrund der durch das Internet gegebenen internationalen Vertriebsmöglichkeiten, sich oftmals ebenfalls grenzüberschreitend vollzieht, zeigt sich, dass die Geltung der Deutschen Rechtsordnung nicht einfach als gegeben vorausgesetzt werden kann.
Nachdem die typischen Erwerbsvorgänge auf das darauf anzuwendende Recht untersucht wurden, wird im Anschluss hieran das Augenmerk der Untersuchung auf der Frage nach der Reichweite des „viralen Effekts“ liegen (Kapitel 5).
Um die bestehenden Rechtsunsicherheiten im Hinblick auf die Reichweite der in der GPL enthaltenen Pflicht zur Offenlegung des Quelltextes zu mindern, werden dabei zunächst

diejenigen Fallgruppen dargestellt, in denen eine Verletzung der GPL eindeutig ausgeschlossen ist, sich der „virale Effekt“ nicht auswirken kann.
Danach liegt der Schwerpunkt der Begutachtung auf einer Konkretisierung der in der GPL enthaltenen Reichweite der Lizenzierungspflicht. Dabei liegt das besondere Augenmerk auf der Frage, unter welchen Voraussetzungen die proprietäre Verwertung von Entwicklungen im Umfeld von GPL-lizenzierter Open Source Software möglich ist.

Kapitel 1

B. Begriff von Freier und Open Source Software

Die Entwicklung von Freier Software begann 1984, durch die Gründung des GNU-Projekts.[13] Ziel des GNU-Projekts war es ein Unix kompatibles Betriebssystem zu entwickeln, das keinen Auflagen der Softwarefirmen unterlag, also komplett frei war. Diese Idee wurde vom Gründer des Projekts, Richard Stallmann, mit dem Kürzel GNU umschrieben, was für was für „GNU's not Unix" steht und andeuten sollte, dass ein Konkurrenzprodukt zu Unix angestrebt wurde. Seit 1984 sind viele Programme durch das GNU-Projekt entwickelt worden, nicht zuletzt seien hierbei die vielen Systemtools zu nennen, die heute einen festen Bestandteil von Linux darstellen.[14] Das GNU-Projekt prägte den Begriff Freier Software (engl. „*Free Software*").

Im Gegensatz zum Begriff der Freien Software, den es somit bereits seit Mitte der 80er Jahre gibt, wurde die Bezeichnung „Open Source Software" erst 1998 anlässlich der Ankündigung von Netscape eingeführt, den Quellcode seines Internet-Browsers zu veröffentlichen.[15]

Der Grund für die Abkehr vom Begriff der Freien Software lag dabei in der missverständlichen Bedeutung des Wortes „frei", die häufig zu der Ansicht führte, dass die entsprechende Software für den Erwerber stets kostenlos erhältlich sei. Diesem Verständnis von Freier Software ist wohl auch die anfängliche Zurückhaltung der Softwarewirtschaft zuzurechnen, welche Freie Software überwiegend als geschäftsschädigend empfand.

Um hier einen Wandel zu bewirken, trafen sich 1998 einige der einflussreichsten Entwickler der Freien Software-Szene,[16] um über eine neue Marketingstrategie zu beraten. Deren Ziel sollte ein Wandel des Begriffsverständnisses sein, damit Freie Software nicht primär mit deren Unentgeltlichkeit verbunden wird, sondern mit der darin enthaltenen Freiheit, die Software zu nutzen, zu bearbeiten und zu verbreiten. Als Folge dieses Treffens wurde der Begriff Open Source Software eingeführt und die Open Source Initiative gegründet.[17]

---

[13] Zu den Motiven und zur Entstehungsgeschichte, vgl. http://www.gnu.org/gnu/thegnuproject.html.

[14] Auf die Unterscheidung zwischen Linux als Betriebssystemkern und den zahlreichen Systemtools, die vom GNU-Projekt entwickelt wurden, weist ausdrücklich das Debian-Projekt hin, dass das Systems daher auch als „GNU/Linux" bezeichnet; vgl. http://www.debian.org/intro/about.

[15] Vgl. http://www.mozilla.org/mission.html.

[16] An diesem Treffen nahmen unter anderem Eric Raymond, Bruce Perens und Tim O'Reilly, teil.

[17] Vgl. http://www.opensource.org/docs/history.php.

Ausgehend von diesem Begriffswechsel, hat sich in der Folgezeit nicht nur das Begriffsverständnis sondern auch die wirtschaftliche Akzeptanz von Freier Software gewandelt. Einige der größten Computerunternehmen[18] begannen damit, sowohl Hard- als auch Software auf Open Source Programme zu portieren.
An der inhaltlichen Bedeutung von Freier Software hat sich jedoch trotz dieses Wandels und des neuen Begriffs der Open Source Software im Wesentlichen nichts verändert.[19] Heute werden beide Begriffe nebeneinander verwendet und beschreiben das gleiche Softwareentwicklungs- und Vertriebsmodell, weshalb sie im Folgenden synonym verwendet werden.

I.Merkmale der Open Source Definition

Der Begriff Open Source Software steht heute für ein Softwareentwicklungs- und Vertriebsmodell, in dem der Nutzer die Software nicht nur unbeschränkt nutzen, sondern darüber hinaus auch weiterentwickeln und -verbreiten kann. Die umfassende Nutzungsmöglichkeit des Nutzers wird dadurch sichergestellt, dass ihm durch die entsprechenden Open Source Lizenzen ein uneingeschränkter Zugang zum Quellcode gewährleistet wird.
Dabei hat sich im Open Source Umfeld jedoch bislang noch keine einheitliche „Open Source Lizenz“ herausgebildet. Vielmehr existieren zahlreiche Lizenzen,[20] die zwar allesamt eine umfassende Rechtseinräumung an den jeweiligen Nutzer vorsehen, diese jedoch teilweise an sehr unterschiedliche Bedingungen knüpfen. Als grobe Untergliederung hat sich in der Praxis eine Unterscheidung zwischen Open Source Lizenzen mit und ohne Copyleft-Effekt eingebürgert.[21]
Um eine Übereinstimmung darüber zu erreichen, was eine Open Source Lizenz auszeichnet, hat die Open Source Initiative (OSI) in der Open Source Definition[22] (OSD) einige Merkmale festgelegt, bei deren Vorliegen von einer Open Source Lizenz ausgegangen werden kann.

[18] Vorreiter waren Corel, Sun Microsystems, IBM und Oracle.
[19] Bedenken an dem Begriff der Open Source Software wurden vor allem von der Free Software Foundation (FSF) und ihrem Begründer Richard Stallman geäußert. Befürchtet wurde insbesondere, dass durch den Begriff der Open Source Software ein Schwerpunkt auf den Aspekt des offenen Quellcodes gesetzt werde und hierdurch letztlich auch solche Software unter den Begriff gefasst werden könnte, deren Quellen zwar offen, die aber im Übrigen „unfrei“ ist; vgl. http://www.fsf.org/licensing/essays/free-software-for-freedom.html.
[20] Allein die Open Source Initiative unterscheidet derzeit 59 Lizenzen, vgl. http://www.opensource.org/licenses/.
[21] Hierzu sogleich S. 10 ff.
[22] Vgl. unter http://www.opensource.org/docs/definition.php.

Weiterhin bietet die OSI auch eine Zertifizierungsmöglichkeit für neue Lizenzen an, welche nach den Kriterien der OSD erstellt wurden.[23]
Damit eine Lizenz als Open Source Lizenz bezeichnet werden kann, müssen nach der OSD folgende Voraussetzungen erfüllt sein:

- Die unbeschränkte Weiterverbreitung der Software muss dem Nutzer gestattet werden. Die Nutzung der Software darf nicht durch die Erhebung von Lizenzgebühren eingeschränkt werden.[24]

- Durch die Lizenz muss sowohl der Vertrieb der Software in kompilierter Form als auch im Quellcode gestattet sein. Wird die Software in kompilierter Form überlassen, muss durch die Lizenz sichergestellt werden, dass der Quellcode für den Nutzer erhältlich ist.[25]

- Die Lizenz muss weiterhin Veränderungen und Weiterentwicklungen an der Software gestatten und es zulassen diese unter der gleichen Lizenz weiter zu vertreiben.[26]

- Eine Beschränkung beim Vertrieb solcher Veränderungen ist ausschließlich hinsichtlich der Weiterverbreitung von verändertem Quellcode zulässig, sofern die Lizenz die Möglichkeit einräumt, den Originalcode zusammen mit einer Patch-Datei[27] zu vertreiben. Weiterhin kann in der Lizenz vorgesehen sein, dass Software, die auf

[23] Einzelheiten hierzu unter http://www.opensource.org/docs/certification_mark.html.
[24] § 1 OSD: *„The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.“*
[25] § 2 OSD: *„The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost–preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.“*
[26] § 3 OSD: *„The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.“*
[27] Unter einem Patch wird eine Datei verstanden, die auf den Originalcode angewendet werden kann und hierdurch die beabsichtigten Veränderungen durchführt. Eine Patch-Datei wird dabei mit dem Unix-Programm Diff erstellt und stellt sich als Textdatei dar, die lediglich die Unterschiede zwischen dem Originalcode und der modifizierten Version enthält. Betrachtet man eine Patch-Datei mit einem Texteditor, werden diejenigen Teile, die aus dem Originalcode entfernt werden sollen typischerweise mit einem Minuszeichen und die neuen Programmteile mit einem Pluszeichen gekennzeichnet sein. Bei der Anwendung des Patches auf den Originalcode werden diese Anweisungen dann nacheinander abgearbeitet und die entsprechenden Veränderungen am Originalcode erstellt. Einzelheiten zu Patches und Diffs siehe unten Kapitel 5, S. 124 f.

dem Originalcode basiert, nur unter einem abweichenden Namen oder einer anderen Versionsnummer vertrieben wird.[28]

- Durch die Lizenz darf weder eine Person oder Personengruppe benachteiligt oder von der Nutzung ausgeschlossen[29] noch der Vertrieb oder die Nutzung der Software auf bestimmte Anwendungsbereiche beschränkt werden.[30]

- Die mit der Lizenz eingeräumten Rechte müssen für jeden gelten, der die Software erworben hat, ohne dass der Nutzer auf den Abschluss eines zusätzlichen Lizenzvertrags angewiesen ist.[31]

- Die Nutzungsrechte, die der Nutzer durch den Abschluss der Lizenz eingeräumt bekommt, dürfen nicht davon abhängen, dass die Software Teil einer Softwaredistribution ist.[32]

- Andere Programme, die zusammen mit der Software vertrieben werden, dürfen durch die Lizenz nicht eingeschränkt werden, insbesondere darf die Lizenz nicht fordern, dass jegliche Software, die auf demselben Datenträger verbreitet wird, ebenfalls Open Source Software ist.[33]

- Letztendlich darf die Lizenz keine Bestimmung enthalten, nach der eine bestimmte Technologie oder Schnittstelle vorgeschrieben wird.[34]

---

[28] § 4 OSD: *„The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.“*

[29] § 5 OSD: *„ The license must not discriminate against any person or group of persons.”*

[30] § 6 OSD: *„The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.“*

[31] § 7 OSD: *„ The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.“*

[32] § 8 OSD: *„The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.“*

[33] § 9 OSD: *„The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.“*

[34] § 10 OSD: *“No provision of the license may be predicated on any individual technology or style of interface.”*

II.Merkmale von Copyleft Lizenzen

Die einzelnen Voraussetzungen der OSD haben in der Vergangenheit zwar einen ersten Beitrag zu einer Vereinheitlichung der Open Source Lizenzen geleistet, allerdings statuiert die OSD lediglich Mindestanforderungen, weshalb auch weiterhin noch erhebliche Unterschiede zwischen den jeweiligen Open Source Lizenzen bestehen.

Besonders ausgeprägt sind die Unterschiede zwischen den Open Source Lizenzen dabei hinsichtlich der darin enthaltenen Bedingungen, unter denen eine Verwertung weiterentwickelter Programmversionen gestattet wird. Ausgehend von den hierbei eingeräumten Freiheiten, hat sich in der Praxis eine Untergliederung der Open Source Lizenzen, in solche mit und ohne Copyleft-Effekt etabliert.[35]

Über Copyleft-Klauseln soll dabei sichergestellt werden, dass auch Weiterentwicklungen der entsprechenden Programme wiederum unter der gleichen Lizenz weitergegeben werden.

Im Gegensatz zu sog. Non-Copyleft-Lizenzen,[36] die eine solche Regelung nicht enthalten, soll bei Copyleft Lizenzen somit verhindert werden, dass Weiterentwicklungen unter einer proprietären Lizenz vermarktet, also lediglich im Objektcode – nicht aber im Quellcode – weitergegeben werden und der Nutzer zudem keine Vertriebs- und Entwicklungsrechte erhält.

Um dieses Ergebnis zu erreichen, knüpfen Copyleft-Lizenzen die Befugnis zum Vertrieb von modifizierten Programmversionen an die Pflicht, dass dieser ausschließlich unter gleichen Lizenz erfolgen darf, unter der das ursprüngliche Programm überlassen wurde.[37] Probleme bereitet in der Praxis dabei vor allem die Frage, in welchen Fällen noch von einer Modifikation des ursprünglichen Programms und in welchen bereits von einem eigenständigen Programmzusatz auszugehen ist, der unter einer anderen Lizenz vermarktet werden dürfte.[38]

Diesem Abgrenzungsproblem kommen die Copyleft-Lizenzen lediglich zum Teil durch eine genauere Bestimmung der Grenzziehung entgegen.[39] Gerade die am weitesten verbreitete

[35] Vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 23 ff, die allerdings zudem noch zwischen Lizenzen mit strengen und solchen mit beschränkten Copyleft-Klauseln unterscheiden.
[36] Am weitesten verbreitet sind die BSD-Lizenz und die Apache Software License.
[37] Unzutreffend insoweit *Koch*, CR 2000, 273, 274, und CR 2000, 333, welcher der Ansicht ist, dass bereits das Behalten von eigenen Programmmodifikationen unzulässig wäre, unabhängig davon, ob ein Vertrieb überhaupt beabsichtigt ist.
[38] Hierzu unter Kapitel 5, S. 135.
[39] So ist die Abgrenzung nach der Mozilla Public License (MPL) gem. Ziff. 1.9 beispielsweise danach zu vollziehen, ob die Änderungen in einer gesonderten Datei gespeichert wurden. Ist dies der Fall, so wirkt sich der Copyleft-Effekt nicht aus.

Copyleft-Lizenz,[40] die GPL, ist im Hinblick auf diese Abgrenzung alles andere als eindeutig.[41] Im Gegensatz zu anderen Copyleft-Lizenzen macht die GPL die Grenzziehung nicht von eindeutigen technischen Umständen abhängig, sondern knüpft das Eingreifen des Copyleft-Effekts in Ziff. 2 Abs. 1 b GPL allein daran, dass die eigenen Entwicklungen ganz oder teilweise von dem Programm oder Teilen davon abgeleitet sind. Für die genaue Bestimmung der Grenze kommt es darauf an, wann ein Werk von einem anderen abgeleitet ist. Weitergehende und vor allem an den technischen Gegebenheiten anknüpfende Regelungen enthält die GPL nicht,[42] weshalb die Auslegung des unbestimmten Begriffs „abgeleitet“ in der Praxis erwartungsgemäß umstritten ist.[43]

III. Abgrenzung zu anderen Formen autodistributiver Software

Die OSD enthält eine Reihe an Mindestanforderungen, bei deren Vorliegen gemeinhin von einer Open Source Lizenz auszugehen ist. Im Umfeld von Open Source Lizenzen lassen sich jedoch auch andere Lizenz- und Vertriebsmodelle finden, die zumindest Teile dieser Anforderungen erfüllen und deren Abgrenzung zum Open Source Bereich daher oftmals zu Schwierigkeiten führt.

1. Public-Domain-Software

Mit Public Domain Software besteht neben dem Open Source Bereich ein weiteres Vertriebsmodell, bei dem Software kostenlos an den Nutzer überlassen wird und dieser die Software frei bearbeiten und vertreiben darf. Eine ausdrückliche Einräumung umfassender Nutzungsrechte erfolgt bei der Überlassung von Public Domain Software im Gegensatz zum Open Source Bereich zumeist nicht, da diese gem. 17 U.S.C. § 105 bereits kraft Gesetzes vom Urheberschutz ausgeschlossen ist.

[40] Beim wohl größten Hosting-Anbieter von Open Source Projekten waren im Januar 2003 rund 50 % der Open Source Programme unter der GPL lizenziert, vgl. http://openfacts.berlios.de/index.phtml?title=Open-Source-Lizenzen, beim Hosting-Anbieter freshmeat haben derzeit rund 66 % der Projekte ihre Entwicklungen unter der GPL lizenziert, vgl. http://freshmeat.net/stats/#license.
[41] Eingehend hierzu unter Kapitel 5, S. 96 ff.
[42] Abgesehen von den Regelungen in Ziff. 2 Abs. 2 bis 4 GPL, die jedoch ebenfalls keine technischen Kriterien liefern, welche eine eindeutige Konkretisierung zulassen würden.
[43] Einzelheiten hierzu unter Kapitel 5, S. 115 ff.

Im Zusammenhang mit Public Domain Software ist allerdings zu berücksichtigen, dass der damit einhergehende – und vor dem Hintergrund des US-Rechts auch mögliche – vollständige Verzicht auf Urheberrechte dem kontinentaleuropäischen Recht aufgrund dessen persönlichkeitsrechtlicher Komponente fremd ist.[44] Insofern sind Public Domain Lizenzen im deutschen Schutzbereich als Einräumung eines einfachen Nutzungsrechts an jedermann auszulegen, mit dem es dem Nutzer gestattet wird, die entsprechende Software umfassend zu nutzen und zu verbreiten.[45]

Im Gegensatz zu Open Source Software wird Public Domain Software jedoch häufig ohne den entsprechenden Source Code überlassen.

2. Freeware

Sofern Freeware überlassen wird, werden dem Nutzer hierdurch zumeist umfassende Nutzungs- und Vertriebsrechte eingeräumt. Diese Rechtseinräumung ist ebenso wie bei der Überlassung von Open Source Software nicht an die Zahlung einer Lizenzgebühr geknüpft, weshalb nicht zuletzt aufgrund der Begriffsähnlichkeit zwischen „Freeware“ und „Free Software“ häufig davon ausgegangen wird, dass der Nutzer von Freeware die gleichen Rechte eingeräumt erhalte.

Mit der Überlassung von Freeware wird jedoch primär die Weiterverbreitung, nicht aber die Weiterentwicklung des Programms angestrebt,[46] weshalb man bei dem Erwerb von Freeware nicht zugleich darauf schließen kann, dass der Nutzer auch die Bearbeitungsrechte eingeräumt bekommt.[47] Deshalb erhält der Nutzer regelmäßig auch keinen Quellcode des Programms.[48] Teilweise wird selbst die bestimmungsgemäße Nutzung auf eine nicht-kommerzielle

[44] Für Deutschland folgt dies explizit aus § 29 Abs. 1 UrhG.

[45] So auch OLG Stuttgart, CR 1994, 743, das zwar nicht von einem einfachen Nutzungsrecht spricht, aber bei der Bezeichnung als Public Domain Software von einem Verzicht auf die Ansprüche wegen Vervielfältigung, Vertrieb und Nutzung ausgeht; vgl. auch *Loewenheim* in: Schricker, § 69 c Rn. 3, der allerdings nicht zwischen Public Domain Software und Freeware unterscheidet; ebenso *Marly*, Softwareüberlassungsverträge, Rn. 324 ff; die Einräumung eines einfachen Nutzungsrechts wird ausdrücklich von *Jaeger / Metzger*, Open Source Software, Rn. 8, genannt.

[46] Aus diesem Grund wurde beispielsweise der Vertrieb als Freeware von Microsoft genutzt, um die Konkurrenz durch das Verschenken des eigenen Internet Explorers aus dem Markt zu drängen vgl. *Jaeger / Metzger*, Open Source Software, Rn. 9; zur Strategie *Rushkoff* http://www.heise.de/tp/r4/artikel/2/2185/1.html.

[47] *Jaeger / Metzger*, Open Source Software, Rn. 9; ähnlich auch *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 305, der darauf hinweist, dass eine Weiterentwicklung der Software durch Dritte weder intendiert noch gestattet ist; gegen die Einräumung von Bearbeitungsrechten auch *Schiffner*, Open Source Software, S. 23; sowie *Weber* in: Festschrift Honsell, S. 45, der die verschiedenen Vertragskonzepte allerdings vor dem Hintergrund des Schweizer Rechts untersucht.

[48] Hierauf weist zu Recht *Jaeger / Metzger*, Open Source Software, Rn. 9, hin.

Verwendung eingeschränkt, weshalb der Vertrieb von Freeware letztendlich dem proprietären Bereich zuzurechnen ist.[49]

3. Shareware

Ebenfalls dem proprietären Softwarevertrieb ist, neben der Überlassung von Freeware, auch der von Shareware zuzurechnen.

Unter Shareware wird dabei ein Vertriebskonzept verstanden, bei dem der Nutzer die Software während einer gewissen „Testphase" kostenlos nutzen darf. Nach dem Ende der „Testphase", dass zumeist zeitlich, teilweise aber auch durch die Intensität der Nutzung begrenzt wird, muss der Nutzer eine bestimmte Nutzungsgebühr bezahlen, um das Programm auch weiterhin nutzen zu können.[50]

Im Gegensatz zum Open Source Modell, erhält der Nutzer an der Shareware zwar häufig Vertriebs- jedoch keine Bearbeitungsrechte eingeräumt.

4. Shared Source Software

Bei der Überlassung von Shared Source Software erhält der Nutzer, ähnlich wie auch im Open Source Bereich, einen Einblick in den Quellcode des entsprechenden Programms.[51] Diese Möglichkeit wird dem Nutzer von Shared Source Software primär zu dem Zweck eingeräumt, den Quellcode studieren und analysieren zu können, sowie um notwendige Anpassungen an der Software vorzunehmen.[52]

Im Gegensatz zum Open Source Bereich erhält der Nutzer von Shared Source Software aber regelmäßig keine Befugnis dazu, Änderungen an der Software vornehmen oder solche weiter verbreiten zu können.

---

[49] So auch *Jaeger / Metzger*, Open Source Software, Rn. 9.

[50] Einzelheiten zu den Rechtsproblemen des Sharewarevertriebs, vgl. *Marly*, Softwareüberlassungsverträge, Rn. 367 ff, m.w.N.

[51] Bekanntestes Beispiel der Shared Source Initiative ist Microsoft, vgl. http://www.microsoft.com/resources/sharedsource/Licensing/default.mspx.

[52] Vgl. zu den verschiedenen Shared Source Lizenzformen http://www.microsoft.com/resources/sharedsource/Licensing/default.mspx

IV. Wirtschaftliche Bedeutung von Open Source Software

Gerade in den letzten Jahren ist eine ständig wachsende wirtschaftliche Bedeutung von Open Source Software zu erkennen. Dabei haben sich eine Reihe von neuen Geschäftsmodellen entwickelt, die sich zwar überwiegend mit der Entwicklung und Vermarktung von Open Source Programmen beschäftigen, jedoch ihre Umsätze nicht mit der Erhebung von Lizenzgebühren erzielen.[53]

1. Distributionen

Wirtschaftlich etabliert hat sich in der Vergangenheit bereits der Vertrieb von Open Source Software in sog. Distributionen. Unternehmen, wie beispielsweise Red Hat[54] oder SUSE/Novell,[55] bündeln in solchen Distributionen überwiegend Open Source Programme, die zusammen mit einer Installationsroutine in einem Paket vertrieben werden, in dem der Nutzer sowohl Linux als Betriebssystem erhält, als auch eine umfangreiche Auswahl an weiteren Anwendungsprogrammen.[56]

Neben der reinen Verschaffung der Programme, die im Internet zudem häufig kostenlos erfolgt,[57] werden in den kostenpflichtigen Distributionen eine Reihe von Zusatzleistungen erbracht. Regelmäßig werden die Distributionen mit einem umfangreichen Benutzerhandbuch vertrieben und dem Nutzer zudem ein (zeitlich begrenzter) Installationssupport geboten.[58]

2. Dienstleistungsmarkt

Aufgrund der fehlenden Möglichkeit Lizenzgebühren für die Überlassung von Open Source Programmen zu fordern, hat sich im Open Source Umfeld bereits zu einem sehr frühen Zeitpunkt ein umfangreicher Dienstleistungsmarkt etabliert. Die Angebote sind dabei vielfältig und reichen von Schulungen und Beratungen bis hin zu technischem Support, wobei

---

[53] Zu den Gründen für ein unternehmerisches Engagement im Open Source Bereich vgl. *Berelecon Research*, FLOSS Final Report – Part 2 Free/Libre Open Source Software: Survey and Study, S. 17 ff, http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf.

[54] http://www.redhat.com.

[55] http://www.novell.com/linux/

[56] Die Anwendungsprogramme sind dabei zwar überwiegend aber nicht zwangsläufig auch Open Source Software. Gerade die Installationsroutine, mit der dem Nutzer die Installation des Linux – Betriebssystems erleichtert werden soll, ist zumeist kein Open Source Programm, bspw. Yast bei Novell, vgl. *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 355, http://freie-software.bpb.de/Grassmuck.pdf.

[57] Vgl. Novell hält beispielsweise eine kostenlose Variante der Distribution unter http://de.opensuse.org/Willkommen_auf_openSUSE.org zum Download bereit.

[58] Vgl. nur die Distribution Suse, bei der Novell, neben einem ausführlichen Handbuch einem 90-tägigen Installationssupport anbietet.

gerade durch die wachsende Akzeptanz von Open Source Software in Unternehmen auch der Bedarf an entsprechenden Dienstleistungen gestiegen ist.

3. Embedded Systeme

Für die vorliegende Untersuchung von besonderem Interesse ist die Verbreitung von Linux im Bereich der sog. Embedded-Systeme.[59] Hierzu zählen beispielsweise Mess-, Steuerungs- und Regelungseinheiten für ABS-Systeme, Haushaltsgeräte, Produktionssteuerungen, Mobilfunkgeräte und Unterhaltungselektronik. Aus rechtlicher Sicht ist der Vertrieb solcher Embedded-Systeme vor allem aufgrund der Koppelung von Hard- und Software interessant. Diese kann immer dann zu rechtlichen Problemen führen, wenn neben dem Linux Betriebssystem auch eigenständige Entwicklungen der Unternehmen auf den Geräten enthalten sind. Diese wollen die Hersteller zumeist nicht unter einer Open Source Lizenz veröffentlichen, obwohl sie technisch mit der Open Source Software verknüpft ist und auch gemeinsam mit dieser vertrieben wird. Die Software soll vielmehr proprietär und deren Quellcode unveröffentlicht bleiben.[60]

Der Einsatz eines Linux-Betriebssystems ist im Bereich der Embedded-Systeme vor allem auf die gestiegenen technischen Anforderungen zurückzuführen, die an diese gestellt werden und die dazu geführt haben, dass selbst kleinste Systeme vielfältige Aufgaben bewältigen sollen. Um dies zu ermöglichen, müssen die Ressourcen solcher Systeme durch den Einsatz von Betriebssystemen koordiniert werden. Auf technischer Seite wurde der Einsatz von Linux auf solchen Systemen dadurch ermöglicht, dass verschiedene Hersteller dazu übergegangen sind für Embedded-Systeme Mikrokontroller anzubieten, die über einen x86 Kern verfügen. Dieser ist zur x86 CPU-Architektur kompatibel und ermöglicht somit einen Einsatz von Linux.

Dass sich im Bereich von Embedded-Systemen gerade Linux einer weiten Verbreitung erfreut,[61] liegt vor allem an dem Bestreben die begrenzten Ressourcen solcher Geräte zu schonen. Insofern liegt bei der Auswahl eines passenden Betriebssystems ein besonderes Augenmerk darauf, dass dieses möglichst wenig Speicherplatz und Ressourcen beansprucht.

[59] Embedded Systeme sind Kleinstcomputer, die in Geräten Steuerungs- und Kontrollfuntionen übernehmen. Oftmals sind sie auf spezielle Anwendungen zugeschnitten und arbeiten ohne die sonst übliche Peripherie, insbesondere ohne Monitor, Tastatur, Maus, o.ä.

[60] Einzelheiten zu diesem Problembereich unten Kapitel 5, S. 96 ff.

[61] Nach einer Studie von *LinuxDevices.com* ist Linux seit 2003 das am häufigsten eingesetzte Betriebssystem im Embedded Bereich, vgl. http://www.linuxdevices.com/articles/AT7070519787.html.

Diese Voraussetzungen werden mit vielen abgespeckten Linux-Versionen[62] erfüllt. Zudem fallen durch den Einsatz von Open Source Betriebssystemen keine Lizenzgebühren an, wodurch sich insbesondere bei der industriellen Produktion von kleinen Embedded-Geräten ein erhebliches Einsparungspotenzial ergibt.[63]

4. Softwareentwicklung

Letztlich spielt Open Source Software aber auch in der Softwareentwicklung eine bedeutende Rolle.[64] Aus rechtlicher Sicht sind dabei vor allem solche Entwicklungen interessant, in denen Unternehmen proprietäre Software entwickeln, die im Umfeld von Open Source Software eingesetzt werden soll. Dabei ist an sog. „value-added" Produkte zu denken, bei denen Unternehmen beispielsweise proprietäre Nischenanwendungen entwickeln, die aber auf ein Open Source Programm aufsetzen oder zumindest Linux als Betriebssystem voraussetzen.
In rechtlicher Hinsicht ist der gemeinsame Vertrieb solcher „value-added" Produkte zusammen mit Open Source Programmen vor allem dann problematisch, wenn diese unter einer Copyleft-Lizenz stehen. In diesen Fällen stellt sich nämlich die Frage, ob der gemeinsame Vertrieb von proprietären Programmkomponenten mit Open Source Software unter verschiedenen Lizenzen zulässig ist.[65] Insofern kommt es im Zusammenhang mit der GPL wiederum auf die Grenzziehung zwischen zustimmungsbedürftiger Modifikation eines Open Source Programms und der Erstellung eines eigenständigen Programms an, dass unabhängig von den Voraussetzungen der GPL vertrieben werden darf.

[62] Von führenden Unternehmen im Embedded Bereich wird bereits seit geraumer Zeit die Schaffung eines Industriestandards für Embedded-Systeme vorangetrieben. Zu diesem Zweck wurde das Embedded Linux Consortium (ELC) geschaffen, vgl. www.embedded-linux.org.
[63] Nicht zuletzt hierauf führen *Jaeger / Metzger*, Open Source Software, Rn. 20, den gestiegenen Einsatz von Linux in Embedded Systemen zurück; ebenso *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 334.
[64] Eine Studie des Fraunhofer Instituts hat bei einer Umfrage unter Unternehmen, deren Tätigkeitsfelder im Umfeld von Open Source Software liegen, ergeben, dass 46 % der Unternehmen auf Basis von OSS-Technologien wie beispielsweise Linux, Apache, mySQL und PHP eigene proprietäre Anwendungen entwickeln, vgl. http://www.iao.fraunhofer.de/d/oss_studie.pdf, S. 45.
[65] Einzelheiten hierzu unter Kapitel 5, S. 115 ff.

Kapitel 2

C. Urheberschaft an Open Source Software

Der Frage nach der Urheberschaft an Open Source Software kommt erhebliche Bedeutung zu. Im Zusammenhang mit der Überlassung von Open Source Software ist sie vor allem dafür maßgeblich, von wem der Nutzer die Nutzungsrechte eingeräumt bekommt.[66] Darüber hinaus entscheidet die Frage nach der Urheberschaft aber auch darüber, wer im Fall von Urheberrechtsverletzungen klagebefugt ist.[67]

Die Urheber von Open Source Programmen lassen sich jedoch – nicht zuletzt aufgrund der Verwendung von kollaborativen und dezentralen Entwicklungsmethoden – mitunter nur mit erheblichen Schwierigkeiten ermitteln.[68] Zwar ist die kollaborative und dezentrale Entwicklung von Software mittlerweile keine Besonderheit mehr, die ausschließlich den Open Source Bereich prägen würde,[69] allerdings ist die Entwicklungsstruktur in Open Source Projekten dadurch gekennzeichnet, dass es im Gegensatz zur proprietären Softwareentwicklung weder einen festen Entwicklerstamm noch feste Aufgabenzuweisungen, sondern vielmehr ausschließlich freiwillige Mitarbeit gibt.

---

[66] In Ziff. 6 GPL ist dies ausdrücklich erwähnt: *„...the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions..."*

[67] Die Klagebefugnis steht gem. § 97 UrhG sämtlichen Urhebern gemeinschaftlich zu. Sie können auf Unterlassung oder auf Schadensersatz klagen, wobei für eine Schadensersatzforderung die Nennung sämtlicher Urheber erforderlich ist, die im Rahmen größerer Projekte häufig nur schwer zu realisieren sein wird. Eingehend zur Problematik vgl. *Koch*, CR 2000, 273, 279, am Beispiel von Linux; allgemein zu Open Source Programmen *Jaeger / Metzger*, Open Source Software, Rn. 165 ff; *Omsels* in: FS Hertin, S. 168; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 17 ff.

[68] Hierbei ist vor allem an die Projekte zu denken, in denen die daran beteiligten Entwickler nicht ausreichend protokolliert werden; zu den Voraussetzungen einer solchen Protokollierung siehe unten S. 37 ff.

[69] Auch größere Firmen entwickeln ihre Software zunehmend in Teams, die – ähnlich wie die Entwickler in Open Source Projekten – mitunter weltweit verstreut sitzen. Gerade am Beispiel des Offshorings wird ersichtlich, wie flexibel die Entwicklung von Software aufgrund der zahlreichen und preiswerten Möglichkeiten der Kommunikation über das Internet geworden ist. Nicht selten werden besonders die zeitintensiven Entwicklungsarbeiten, die zugleich nur ein Mindestmaß an technischem Know-how voraussetzen, nach Indien bzw. vergleichbare Niedriglohnländer verlagert, während die Kernbereiche auch weiterhin firmenintern entwickelt werden. Um einen kollaborativen und dezentralen Entwicklungsprozess zu ermöglichen, werden dabei Entwicklungsplattformen (z.B. Sourceforge Enterprise, CollabNet Enterprise Edition, Rational Rose Enterprise und Codebeamer) eingesetzt, die in oftmals ähnlicher Form auch zur Koordination in Open Source Projekten verwendet werden.

Ebenso wie im proprietären Umfeld tauschen sich die Entwickler von Open Source Software über Entwicklungsplattformen[70] aus und kommunizieren ausschließlich via Internet. Mehr noch als im proprietären Bereich ist es mangels vertraglicher Bindungen zwischen den Beteiligten für die Entwicklung von Open Source Software unerheblich, wo sich die jeweiligen Entwickler aufhalten, solange sie nur über einen Internetzugang verfügen. Aus diesem Grund sind die größeren Open Source Projekte typischerweise dadurch gekennzeichnet, dass die daran beteiligten Entwickler weltweit verstreut sind.[71] Die Schwierigkeiten, die sich hieraus für die Fragen des Internationalen Privatrechts ergeben, werden später behandelt.[72]

Berücksichtigt man jedoch, dass als Urheber in dieser Form der Softwareentwicklung grundsätzlich alle Entwickler in Betracht kommen, wird ersichtlich, wie komplex sich die Urheberschaft bei der kollaborativen und dezentralisierten Entwicklung von Programmen gestalten kann.

Im proprietären Bereich führt dies indes nur selten zu Problemen, da sämtliche Entwickler zumeist angestellt oder beauftragt sind und deren Rechte somit gem. § 69 b UrhG auf den Arbeitgeber bzw. Dienstherrn übergehen.[73] Im Open Source Projekten fehlt es hingegen regelmäßig sowohl an einem Arbeitgeber,[74] als auch an Verwertungsregelungen im Hinblick auf die gemeinsamen Entwicklungen.[75]

Für die Frage nach der Urheberschaft an Open Source Programmen ist daher zunächst nur entscheidend, ob ein schutzfähiges Werk vorliegt und ob dieses durch einen einzelnen oder mehrere Entwickler hergestellt wurde.

---

[70] Diese Entwicklungsplattformen werden den Entwicklern von Open Source Software zumeist unentgeltlich von Hosting-Anbietern zur Verfügung gestellt; vgl. http://www.sourceforge.net, http://www.berlios.de, http://www.freshmeat.net und http://www.javaforge.com.

[71] Zur Beteiligungsstruktur in Open Source Projekten vgl. *David / Waterman, u.a.*, FLOSS-US The Free/Libre/Open Source Software Survey for 2003, S. 5 ff, http://www.stanford.edu/group/floss-us/report/FLOSS-US-Report.pdf; allgemein zur Entwicklung von Open Source Projekten *Raymond*, The Cathedral and the Bazaar, S. http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/; zu den üblichen Projektstrukturen sowie den dabei zum Einsatz kommenden Entwicklungswerkzeugen *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 233 ff, http://freie-software.bpb.de/Grassmuck.pdf; zu den urheberrechtlichen Konsequenzen *Jaeger / Metzger*, Open Source Software, Rn. 143; mit Bezugnahme auf Linux *Koch*, CR 2000, 273, 277.

[72] Siehe unten Kapitel 4, S. 64.

[73] Ebenso auch die Einschätzung bei *Jaeger / Metzger*, Open Source Software, Rn. 143; *Küng*, MR 2004, 21, 23.

[74] Eine Ausnahme ist der Fall, in dem Open Source Software zunächst ausschließlich firmenintern entwickelt und erst danach als Open Source Software freigegeben wurde. Solche Entwicklungsprozesse ließen sich erst jüngst bei Novell und Red Hat beobachten, die hinsichtlich GNOME Oberflächen zunächst eine völlige Eigenentwicklung vollzogen haben und diese erst nach deren Vollendung zur Weiterbearbeitung freigaben, vgl. http://www.heise.de/newsticker/meldung/69447Kde.

[75] Zum Teil hat sich dies in der Vergangenheit verändert; so setzt beispielsweise die Beteiligung an der Entwicklung von Open Office die vorherige Übertragung der Verwertungsrechte voraus, vgl. http:// www.openoffice.org/licenses/jca.pdf.

I.Schutzfähigkeit von Software

1.Urheberrechtliche Schutzfähigkeit von Computerprogrammen

Computerprogramme wurden durch die Urheberrechtsnovelle 1985[76] erstmals – als Programme für die Datenverarbeitung – ausdrücklich in den gesetzlichen Katalog der geschützten Werke aufgenommen. Ein weitergehender urheberrechtlicher Schutz für Computerprogramme wurde mit dem Zweiten Gesetz zur Änderung des Urheberrechts 1993 erreicht, durch das die Richtlinie 91/250/EWG vom 14. Mai 1991 über den Rechtsschutz von Computerprogrammen umgesetzt wurde. Mit der Richtlinie wurde eine Harmonisierung des urheberrechtlichen Schutzes innerhalb der EU bezweckt,[77] was für Deutschland, aufgrund der damaligen Rechtsprechung, eine Absenkung der Schutzvoraussetzungen für Computerprogramme bedeutete.[78] Um der Richtlinie eindeutig nachzukommen und unzweifelhaft einen Wandel der Schutzvoraussetzungen zu statuieren, entschloss sich der damalige Umsetzungsgeber dazu, die materiellrechtlichen Regelungen der Richtlinie wörtlich zu übernehmen, indem er diese nicht etwa in die jeweiligen Vorschriften des Urheberrechtsgesetzes einarbeitete, sondern vollständig und „en block“ in einem gesonderten Abschnitt für Computerprogramme in den §§ 69 a ff UrhG übernahm.[79] Trotz dieser Sonderstellung der urheberrechtlichen Regelungen für Computerprogramme in den §§ 69 a ff. UrhG stellen diese keine abschließenden Vorschriften dar, so dass in den ungeregelten Bereichen die allgemeinen Regelungen des Urheberrechts anwendbar bleiben.[80]

[76] Gesetz zur Änderung von Vorschriften auf dem Gebiet des Urheberrechts, BGBl. 1985, Teil I, S. 1137.
[77] Vgl. amtl. Begründung BT-Drucks. 12/4022, S. 9.
[78] Zu den hohen Anforderungen der früheren Rechtsprechung an die Schöpfungshöhe vgl. BGH GRUR 1985, 1041, 1047 f. – Inkasso-Programm.
[79] Vgl. *Dreier* in: Dreier / Schulze, § 69 a Rn. 25; ebenso *Erdmann / Bornkamm*, Schutz von Computerprogrammen - Rechtslage nach der EG-Richtlinie, GRUR 1991,877, die eine richtlinienkonforme Absenkung der Schutzvoraussetzungen allerdings bereits anhand einer entsprechenden Auslegung von § 2 Abs. 2 UrhG für vertretbar halten.
[80] Vgl. § 69 a Abs. 4 UrhG, der die Bestimmungen für Sprachwerke für anwendbar erklärt. Einzelheiten unter *Loewenheim* in: Schricker, § 69 a Rn. 23; sowie *Dreier*, GRUR 1993, 781, 783 f.

2.Computerprogramme als Schutzgegenstand

Vom Anwendungsbereich der §§ 69 a ff UrhG werden Computerprogramme erfasst, ohne dass sich dem Gesetz eine Legaldefinition davon entnehmen lässt, was hierunter zu verstehen ist. Von einer Legaldefinition hat der Gesetzgeber dabei bewusst abgesehen, um nicht durch die technischen Entwicklungen überholt zu werden.[81]

Definitionen zum Begriff des Computerprogramms sind hingegen sowohl in der DIN 4430 als auch in § 1 der Mustervorschriften der WIPO enthalten. Während ein Computerprogramm nach DIN 4430 eine vollständige Anweisung ist, die zusammen mit allen erforderlichen Vereinbarungen zur Lösung einer Aufgabe geeignet ist, definiert § 1 der Mustervorschriften der WIPO ein Computerprogramm als eine Folge von Befehlen, die nach Aufnahme in einen maschinenlesbaren Träger fähig sind, dass eine Maschine mit informationsverarbeitenden Fähigkeiten eine bestimmte Funktion oder Aufgabe oder ein bestimmtes Ergebnis anzeigt, ausführt oder erzielt.

Unabhängig von diesen Definitionsansätzen ist nach allgemeiner Ansicht im Rahmen der §§ 69 a ff UrhG der Begriff des Computerprogramms weit zu verstehen.[82] Er umfasst Computerprogramme in jeder Gestalt und Ausdrucksform, einschließlich des Entwurfsmaterials, § 69 a Abs. 1 UrhG. Eine Abgrenzung muss regelmäßig zu reinen Daten erfolgen, die im Gegensatz zum Computerprogramm keine Befehls- oder Steuerungsfunktionen, sondern ausschließlich Informationen enthalten.[83]

3.Schutzvoraussetzungen

Der von den §§ 69 a ff UrhG eingeräumte umfassende Schutz von Computerprogrammen wird lediglich durch § 69 a Abs. 3 S. 1 UrhG begrenzt. Hiernach setzt der urheberrechtliche Schutz ein individuelles Werk voraus, dass zudem ein Mindestmaß an Schöpfungshöhe erreicht.

Im Hinblick auf die Individualität des Programms ist dabei einzig zu fordern, dass es sich bei dem Werk nicht um eine Kopie handelt, sondern die erbrachte Entwicklungsleistung das

[81] Vgl. amtl. Begründung BT-Drucks. 12/4022, S. 9.
[82] Vgl. *Dreier* in: Dreier / Schulze, § 69 a Rn. 12; *Junker / Benecke*, Computerrecht, Rn. 30 f.
[83] *Dreier* in: Dreier / Schulze, § 69 a Rn. 12; *Hoeren* in: Möhring / Nicolini, § 69 a Rn. 3 weisen zu Recht darauf hin, dass nicht jede Datei eines Softwarepakets unter den Computerprogrammbegriff fällt, sondern hier eine differenzierte Betrachtungsweise erforderlich ist.

Ergebnis der eigenen geistigen Fähigkeiten ist. Individualität liegt damit vor, wenn ein bestehender Gestaltungsspielraum bei der Entwicklung einer Problemlösung durch einen eigenen Lösungsansatz ausgeschöpft wurde.[84] Sie wird immer dann gegeben sein, wenn dem Entwickler lediglich das Problem vorgegeben, er hinsichtlich der Art und Weise der Lösung jedoch ungebunden ist.

Von einem Erreichen der erforderlichen Schöpfungshöhe wird immer dann auszugehen sein, wenn die Konzeption der Software Eigentümlichkeiten aufweist, die nicht als trivial, banal und von der Sachlogik her zwingend vorgegeben erscheinen.[85]

Mit der Umsetzung der Softwarerichtlinie 91/250/EWG wurden die Anforderungen an die Schöpfungshöhe somit so weit reduziert, dass der urheberrechtliche Schutz nunmehr die Regel ist.[86]

II. Urheberschaft an Open Source Software

Open Source Projekte sind typischerweise durch kollaborative Zusammenarbeit mehrerer Entwickler gekennzeichnet. Im Gegensatz zum Entwicklungsprozess proprietärer Programme, sind die verschiedenen Entwickler in Open Source Projekten jedoch in aller Regel nicht durch einen gemeinsamen Arbeitgeber, sonder lediglich in der Weise miteinander verbunden, dass sie sich über das Internet untereinander austauschen. Die meisten Projekte unterstützen diesen Austausch dadurch, dass sie themenspezifische Mailinglisten und Newsgroups mitunter aber auch sog. Channels im Internet Relay Chat (IRC) bereithalten.[87]

Die Beteiligungsstrukturen an Open Source Projekten sind zumeist dadurch geprägt, dass zu Projektbeginn lediglich einzelne Initiatoren vorhanden sind, die einen ersten Entwurf der

[84] OLG Düsseldorf CR 1997, 337, 339; OLG Karlsruhe CR 1996, 606, 610; *Dreier* in: Dreier / Schulze, § 69 a Rn. 26.
[85] Amtl. Begründung BT-Drucks. 12/4022 S. 10; OLG München CR 2000, 429, 430; OLG München CR 1999, 688, 689; OLG Frankfurt CR 1999, 7, 9 – Update-Software; OLG Düsseldorf CR 1997, 337, 338 – Dongle-Umgehung; *Loewenheim* in: Schricker, § 69 a Rn. 19; *Dreier* in: Dreier / Schulze, § 69 a Rn. 27; gegen den pauschalen Ausschluss banaler Schöpfungen *Hoeren* in: Möhring / Nicolini, § 69 a Rn. 16.
[86] Dies entspricht der Intention des Gesetzgebers, amtl. Begründung BT-Drucks. 12/4022 S. 9; *Loewenheim* in: Schricker, § 69 a Rn. 19.
[87] Exemplarisch sei hier nur auf die Kommunikationsmöglichkeiten der Linux Distributionen Ubuntu, http://www.ubuntu.com/community/irc; Fedora, http://fedoraproject.org/wiki/Communicate; und Suse, http://de.opensuse.org/Kommunikation verwiesen.

Software weitgehend ohne Unterstützung entwickeln[88] und diesen danach zur gemeinschaftlichen Weiterentwicklung freigeben.[89] Ab diesem Zeitpunkt hängt die Zahl der beteiligten Personen stark vom Erfolg und der Verbreitung der Software ab. Die Initiatoren des Projekts bilden im Rahmen der Weiterentwicklungen des Projekts oftmals das sog. Core-Team, das die allgemeine Richtung der Entwicklung festlegt, neue Entwickler zur Mitarbeit motiviert und die eingereichten Entwicklungsbeiträge koordiniert.[90]

Die Beteiligung der Entwickler an der Weiterentwicklung des Programms erfolgt auf unterschiedliche Arten[91] und wird häufig durch die verwendeten Entwicklungsplattformen vorgegeben.[92] Viele Entwicklungsplattformen sehen sog. Tracker vor, in denen Dritte die Möglichkeit haben Fehler zu melden (sog. Bug Reports), die Lösung eines Fehlers als sog. Patch[93] einreichen und Wünsche für die Erweiterungen der Software äußern (sog. Feature Request) können.[94] Daneben besteht zumeist die Möglichkeit sich über Mailinglisten auszutauschen und hierüber auch größere Funktionserweiterungen einzusenden, die mitunter noch nicht in einem Feature Request enthalten waren.[95]

[88] In der Grundannahme wohl ähnlich *Schiffner*, Open Source Software, S. 117; eingehend zum Entstehungsprozess und zur Struktur von Open Source Projekten *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 235 ff, http://freie-software.bpb.de/Grassmuck.pdf.

[89] Zu den „Regeln", die bei der Weiterentwicklung beachtet werden, sollten *Raymond*, The Cathedral and the Bazaar, S. http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/.

[90] Eingehend zu den typischen Entwicklungsstrukturen in Open Source Projekten *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 237, http://freie-software.bpb.de/Grassmuck.pdf

[91] Zu den Aufgaben eines „Projektleiters" *Raymond*, The Cathedral and the Bazaar, S. http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/; zur Struktur in Open Source Projekten *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 233 ff., http://freie-software.bpb.de/Grassmuck.pdf.

[92] Entwicklungsplattformen bzw. Softwareentwicklungsmanagement werden den Entwicklern von verschiedenen Hosting-Anbietern kostenfrei zur Verfügung gestellt; beispielsweise bei http://www.sourceforge.net, http://www.berlios.de, http://www.freshmeat.net und http://www.javaforge.com.

[93] Allgemein zum Begriff Patch siehe oben Fn. 27; zum Aufbau von Diffs und Patches siehe unten Kapitel 5, S. 124.

[94] So können die Entwickler beispielsweise bei der Entwickungsplattform von Sourceforge http://www.sourceforge.net/ sämtliche Kommunikationswege anbieten, was von den meisten Projekten auch wahrgenommen wird; vgl. beispielsweise das Projekt Gaim, vgl. http://sourceforge.net/projects/gaim/ oder auch das Projekt phpMyAdmin, http://sourceforge.net/projects/phpmyadmin.

[95] Die Einrichtung von solchen Mailinglisten wird von sämtlichen Entwicklungsplattformen als optionale Kommunikationsmöglichkeit zwischen den Entwicklern angeboten; vgl. beispielsweise bei http://www.sourceforge.net, http://www.berlios.de und http://www.freshmeat.net; eingehend zu den Kommunikationwerkzeugen *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 241 ff, http://freie-software.bpb.de/Grassmuck.pdf.

1.Urheberrechtliche Relevanz der Entwicklungsbeiträge

Unter Zugrundelegung dieser Entwicklungsstrukturen und im Hinblick auf die Voraussetzungen, unter denen ein urheberrechtlicher Schutz für einen Entwicklungsbeitrag in Betracht kommt, lässt sich für die Beteiligung an Open Source Projekten folgendes festhalten:

Allein das Melden von Fehlern kann keine urheberrechtlich relevante Handlung darstellen, sofern nicht zugleich auch ein Lösungsvorschlag unterbreitet wird.

Ebenso verhält es sich mit den Äußerungen von Erweiterungswünschen (sog. Feature Requests). Auch diesen fehlt es an einem schöpferischen Gehalt, so dass urheberrechtlicher Schutz hieran ausscheidet.

Eindeutig ist zudem, dass diejenigen Projektmitglieder, welche die eingesendeten Patches oder Funktionserweiterungen lediglich in den Quellcode einfügen, hierdurch keine urheberrechtlich relevante Handlung vornehmen.[96]

Wird hingegen ein Fehler nicht nur gemeldet, sonder auch ein Patch eingereicht, so sind die Schutzvoraussetzungen genauer zu betrachten. Je nachdem wie umfangreich und schwierig eine solche Fehlerbeseitigung ist, kann diese unter Umständen urheberrechtlich schutzwürdig sein,[97] muss jedoch immer auch zur bloßen Gehilfenstellung abgegrenzt werden.

Eine solche kommt in Betracht, wenn dem Entwickler kein Raum für gestalterische Eigenleistungen bleibt, weil ihm beispielsweise feste Vorgaben bezüglich der Art und Weise der Gestaltung durch die Urheber gemacht wurden.[98] Diese Konstellation kann bei der Fehlerbeseitigung vorkommen, wenn lediglich kleinere Fehler vorliegen, die sich in dem konkreten Zusammenhang ohne größeren Aufwand vernünftigerweise nur auf eine bestimmte Art beseitigen lassen.[99] Ebenso sprechen kleinere Fehler eher dafür, dass dem Entwickler die Möglichkeit fehlen wird, bei deren Behebung die erforderliche Schöpfungshöhe zu erreichen.[100] Allein der Umfang einer Fehlerbehebung spricht dann bereits gegen das

[96] Etwas anderes mag gelten, sofern zusätzliche Anpassungsleistungen erforderlich werden, weil etwa mehre unterschiedliche Veränderungen an einer Quellcodedatei vorgenommen wurden, die im Rahmen der Implementierung sodann in der Weise umgestaltet werden müssen, dass sie zueinander kompatibel werden. Vorliegend wird jedoch unterstellt, dass sowohl die Patches als auch die Funktionserweiterungen in einer Form eingereicht werden, die deren Verwendung ohne größeren Aufwand ermöglicht.

[97] Vgl. auch *Schiffner*, Open Source Software, S. 123, der allerdings im Ergebnis davon ausgeht, dass die Behebung eines Fehlers tendenziell keinen schöpferischen Gehalt aufweist; zur urheberrechtlichen Relevanz von Patches und Bugfixes im Zusammenhang mit der Entstehung von Linux *Koch*, CR 2000, 273, 279.

[98] *Schulze* in: Dreier / Schulze, § 7 Rn. 9; *Loewenheim* in: Schricker, § 8 Rn. 8; *Rehbinder*, Urheberrecht, Rn. 170.

[99] Von einer fehlenden schöpferischen Gestaltungsmöglichkeit bei Fehlerbeseitigungen geht daher tendenziell *Schiffner*, Open Source Software, S. 123, aus.

[100] So auch *Schiffner*, Open Source Software, S. 123.

Vorliegen der erforderlichen schöpferischen Gestaltung und mithin gegen eine Urheberschaft des Entwicklers.

Dabei dürfen die Anforderungen an die Schöpfungshöhe jedoch nicht überspannt werden. Insbesondere im Hinblick auf die neuere Gesetzeslage kann von einer fehlenden schöpferischen Tätigkeit nur noch dann ausgegangen werden, wenn sich die Fehlerbeseitigung auf gänzlich Banales bezogen hat.[101] Dies ist etwa anzunehmen, wenn lediglich Syntaxfehler oder die Werte von Variablen korrigiert werden,[102] selbst wenn das Auffinden des Fehlers aufwendig war.

Neben der reinen Fehlerbeseitigung kann aber auch die Zusendung von zusätzlichen Funktionserweiterungen urheberrechtlich relevant werden. Funktionserweiterungen werden dabei häufig von Entwicklern eingereicht, die sich nicht damit begnügen, Erweiterungen in den Mailinglisten vorzuschlagen, sondern diese bereits teils funktionsfähig, teils aber auch nur in einer rudimentären Erstversion im Quellcode einsenden. Im Gegensatz zu kleineren Patches ist die Entwicklung von Funktionserweiterungen regelmäßig aufwendiger, so dass diese öfter über eine ausreichende Schöpfungshöhe verfügen werden.[103]

2.Formen der Urheberschaft

Neben der Frage der Schutzfähigkeit der einzelnen Werkbeiträge stellt sich im Open Source Bereich jedoch im besonderen Maß die Frage danach, welche Form der Urheberschaft durch deren Implementierung begründet wird. Das UrhG sieht verschiedene Formen der Urheberschaft vor, von denen im Rahmen der Softwareentwicklung vor allem die Miturheberschaft gem. § 8 UrhG, das Bearbeiterurheberrecht gem. § 3 UrhG sowie die Werkverbindung gem. § 9 UrhG eine Rolle spielen. Die von § 7 UrhG geregelte Einzelurheberschaft kommt im Open Source Bereich zumeist nur in sehr kleinen und eher unbedeutenden Projekten vor, solange der Entwicklungsaufwand überschaubar ist.

Im Gegensatz zur Werkschaffung durch einen einzelnen Entwickler zeichnet sich sowohl die proprietäre als auch die Open Source Softwareentwicklung häufig durch die Beteiligung

[101] Vgl. nur *Dreier* in: Dreier / Schulze, § 69 a Rn. 27; eingehend hierzu m.w.N. bereits oben S. 20.
[102] Vgl. auch *Schiffner*, Open Source Software, S. 123, der in solchen Fällen eine urheberrechtliche Gehilfenschaft für nahe liegend erachtet.
[103] Insoweit ähnlich *Schiffner*, Open Source Software, S. 123 f, der zwar nicht den Begriff der Funktionserweiterung benutzt, sich allerdings vergleichbar auf eigenständigen Module und Unterprogrammen bezieht und von deren urheberrechtlicher Schutzfähigkeit ausgeht.

mehrerer Entwickler aus, die in einem dezentralen und kollaborativen Entwicklungsprozess zusammenarbeiten.[104]

Sofern sich die Entwickler dabei an der Weiterentwicklung von Open Source Programmen beteiligen, indem sie schöpferische Patches oder Funktionserweiterungen einreichen, kann durch deren Implementierung in das Programm, sowohl eine Miturheberschaft, aber auch ein Bearbeiterurheberrecht begründet werden.

3.Miturheberschaft

Voraussetzung für das Vorliegen einer Miturheberschaft ist, neben der gemeinschaftlichen Werkschöpfung, dass die einzelnen Beiträge einer Gesamtidee untergeordnet werden, die einer einheitlichen Konzeption der Aufgabe entspricht.[105] Weiterhin setzt Miturheberschaft gem. § 8 Abs. 1 UrhG voraus, dass das hergestellte Werk einheitlich ist, also die jeweiligen Einzelbeiträge der Urheber nicht selbstständig verwertbar sind.[106]

a.Unterordnung unter die Gesamtidee

Aufgrund der Entwicklungsstrukturen in Open Source Projekten, die sich insbesondere dadurch auszeichnen, dass die Softwareentwicklung hier nicht nur horizontal, sondern auch zeitlich gestaffelt verläuft, stellt sich gerade vor dem Erfordernis der Unterordnung unter die Gesamtidee die Frage danach, welche Voraussetzungen hieran zu stellen sind und insbesondere zu welchem Zeitpunkt diese gegeben sein muss.[107]

---

[104] Hierzu bereits oben S. 17 m.w.N. in Fn. 71.

[105] *Loewenheim* in: Schricker, § 8 Rn. 9; ebenso *Dreier* in: Dreier / Schulze, § 8 Rn. 2; *Ahlberg* in: Möhring / Nicolini, § 8 Rn. 4; *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 39; Waldenberger, die Miturheberschaft im Rechtsvergleich, S. 26.

[106] So bereits zum früheren Recht BGH GRUR 1959, 335, 336 – Wenn wir alle Engel wären; *Loewenheim* in: Schricker, § 8 Rn. 5 ff; *Dreier* in: Dreier / Schulze, § 8 Rn. 4; Bedenken am Kriterium der „gesonderten Verwertbarkeit“ hat *Ahlberg* in: Möhring / Nicolini, § 8 Rn. 15 ff; zur Abgrenzung von Werkeinheit zu Werkmehrheit bei Computerprogrammen *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 152; im spezifischen Zusammenhang mit den Regelungen der GPL *Jaeger / Metzger*, Open Source Software, Rn. 47 ff.

[107] Hierzu sogleich ab S. 28

aa)Gemeinsame Gesamtidee?

Im Open Source Bereich wird die Gesamtkonzeption der Entwicklungen und somit auch die maßgebliche Gesamtidee häufig von Projektmitgliedern, einem Komitee oder auch einzelnen Maintainern vorgegeben. Diese erbringen im Rahmen der Weiterentwicklung allerdings nicht zwangsläufig auch eigene schöpferische Werkbeiträge,[108] so dass sich die Frage stellt, ob eine Miturheberschaft auch bei der Unterordnung unter eine Gesamtidee entsteht, die von keinem der Miturheber stammt.

Dies wird in der Literatur teilweise mit der Begründung abgelehnt, dass für eine gemeinsame Werkerstellung eine Gesamtidee Voraussetzung sei, die von allen Urhebern gemeinschaftlich – nicht aber von einzelnen Außenstehenden – konzipiert wurde.[109] Die gemeinsame Werkschaffung erfordere wechselseitige Einflussnahme, welche aber nur dann gegeben wäre, wenn die Gesamtidee von sämtlichen Urhebern konzipiert und getragen wird.[110]

Gegen eine solche Auslegung des § 8 UrhG scheint jedoch bereits dessen Wortlaut zu sprechen, der kein Anzeichen dafür enthält, von wem das Ziel des Schaffens kommen muss.[111] In § 8 Abs. 1 UrhG ist ausschließlich von einem „*gemeinsam geschaffenen*“ Werk die Rede, ohne dass hieraus hervorgehen würde, dass dieses an weitere Kriterien gebunden wäre. Zudem wird auch im gewöhnlichen Sprachgebrauch noch von einem gemeinsamen Schaffen ausgegangen, wenn das Ziel des Schaffens von einem Dritten vorgegeben wurde.[112]

Auch aus der systematischen Stellung des § 8 UrhG lässt sich letztlich kein Grund für das Erfordernis einer gemeinschaftlichen Gesamtidee anführen.[113] Die systematische Stellung würde vielmehr nur dann eine enge Auslegung des § 8 UrhG erfordern, wenn das Erfordernis einer gemeinschaftlichen Zielbestimmung durch alle Urheber erforderlich wäre, um eine Abgrenzung zwischen der Miturheberschaft und den sonstigen Formen der Mehrurheberschaft zu gewährleisten. Als Abgrenzungsfälle kämen in diesem Zusammenhang jedoch

[108] Ein gutes Beispiel hierfür ist Linux, über dessen Weiterentwicklung letztlich Linus Torvalds zusammen mit einigen vertrauten Entwicklern (u.a. Dave Miller, Stephen Tweedie und Alan Cox) entscheidet, deren Hauptaufgabe mittlerweile jedoch nicht mehr in der eigenen Entwicklung von Code, sondern vielmehr in dem Sichten, Testen und Bewerten der eingesendeten Patches besteht; vgl. *Diedrich*, In den Tiefen des Kernel Interview mit dem Linux-Entwickler Alan Cox, S. 34, als Online-Ausgabe unter http://www.heise.de/ct/99/25/034/default.shtml.

[109] In diese Richtung *Loewenheim* in: Schricker, § 8 Rn. 9, der eine gemeinschaftliche Konzeption fordert; so auch *Thum* in: Wandtke / Bullinger, § 8 Rn. 17, die neben der gemeinschaftlichen Konzeption noch eine gemeinschaftliche Strukturierung fordert; deutlich *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 37, „der Plan eines Dritten verbindet die Beitragsurheber nicht“.

[110] *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 36.

[111] Ähnlich *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 70, der darauf hinweist, dass sich unter den Wortlaut des § 8 UrhG auch das arbeitsteilige Schaffen von Einzelteilen nach der Planung und Einteilung eines Dritten subsumieren lässt.

[112] *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 70.

[113] So im Ergebnis auch *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 70.

ausschließlich die Unterscheidung zwischen einem arbeitsteilig geschaffenen Werk und einem Sammelwerk im Sinne von § 4 UrhG sowie der Werkverbindung nach § 9 UrhG in Betracht.

Von einem Sammelwerk lässt sich die Miturheberschaft hingegen eindeutig anhand der „gesonderten Verwertbarkeit“ der Einzelbeiträge abgrenzen,[114] bei deren Vorliegen eine Miturheberschaft ausgeschlossen ist. Anhand der „gesonderten Verwertbarkeit“ der Beiträge kann die Miturheberschaft zudem von der reinen Werkverbindung nach § 9 UrhG abgegrenzt werden,[115] weshalb sich aus der systematischen Stellung des § 8 UrhG somit kein Grund dafür ergibt, die Voraussetzungen der Miturheberschaft über den Wortlaut des § 8 UrhG hinaus einzuschränken.

Dieses Ergebnis wird auch durch die Entstehungsgeschichte des § 8 UrhG bestätigt. So lässt sich der Begründung zum Entwurf des UrhG entnehmen, dass der klassische Anwendungsfall des § 8 Abs. 1 UrhG derjenige sein sollte, in dem ein Gruppenwerk aus nicht gesondert verwertbaren Beiträgen vorliege.[116] Von einem Gruppenwerk sollte dabei auszugehen sein, wenn mehrere Urheber unter Leitung eines Herausgebers ein Werk schaffen. Die eigene Urheberschaft des Herausgebers ist hierbei ausdrücklich keine Voraussetzung.[117] Da zu der Leitung des gemeinschaftlichen Schaffensprozesses aber auch die Planung, Koordination und mithin auch die Erstellung einer Gesamtidee zu fassen ist, wird hieraus ersichtlich, dass die Leitung des gemeinschaftlichen Schaffensprozesses auch Dritten zustehen kann, ohne dass die Miturheberschaft der Schöpfer hierdurch zwangsläufig ausgeschlossen würde.[118]

Insgesamt bleibt somit festzuhalten, dass das Erfordernis des gemeinsamen Schaffens allein die Unterordnung aller Urheber unter eine Gesamtidee voraussetzt,[119] ohne dass diese auch von allen Urhebern herrühren muss.

Für Softwareentwicklungen im Open Source Bereich bedeutet dies, dass es für die Begründung von Miturheberschaft an der Software unerheblich ist, wenn die Gesamtidee nicht ausschließlich von den Urhebern, sondern von Projektmitgliedern, einem Komitee, oder einem einzelnen Maintainer vorgegeben wird. Es reicht für ein gemeinsames Schaffen im

[114] *Loewenheim* in: Schricker, § 4 Rn. 17; *Dreier* in: Dreier / Schulze, § 4 Rn. 5.
[115] *Loewenheim* in: Schricker, § 9 Rn. 6; *Thum* in: Wandtke / Bullinger, § 9 Rn. 8.
[116] Begründung BT-Drucks. IV/270, S. 42.
[117] Begründung BT-Drucks. IV/270, S. 42.
[118] So auch *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 71.
[119] Vgl. *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 71; im Ansatz ähnlich *Bohr*, Die Urheberrechtsbeziehungen der an der Filmherstellung Beteiligten, S. 39, der zwar grundsätzlich von einer Gesamtkonzeption ausgeht, allerdings nicht ausschließt, dass „Personen, die im Rahmen eines Arbeits-, Dienst- oder Werkvertrages an der Herstellung“ des Werks mitwirken hierdurch Miturheber werden können.

Sinne von § 8 Abs. 1 UrhG aus, wenn sich sämtliche Urheber einer vorgegebenen Gesamtidee unterordnen.

bb)Maßgeblicher Zeitpunkt

Ungeklärt ist damit jedoch noch, welcher Zeitpunkt für die Gesamtidee maßgeblich ist, oder anders ausgedrückt in welchem Zeitraum die Verständigung über diese stattgefunden haben muss, um von einer Miturheberschaft ausgehen zu können. Diese Frage stellt sich insbesondere im Hinblick auf die den Open Source Bereich prägenden Entwicklungsstrukturen, bei denen die Einsendung von Patches und Funktionserweiterungen häufig unaufgefordert und erst zu einem Zeitpunkt erfolgt, in dem das Grundprogramm schon einen gewissen Entwicklungsstand aufweist. Insofern könnten Zweifel daran bestehen, dass diese Beiträge noch der Gesamtidee untergeordnet werden können und durch deren Implementierung in den Quellcode eine Miturheberschaft am Gesamtprogramm entsteht.

Eindeutig zu einer Miturheberschaft führt die Implementierung solcher Entwicklungen zunächst einmal, wenn die Urheber bereits vor der Werkerstellung einen Gesamtplan aufstellen, unter den sie sich unterordnen.[120] Ebenso kann von einem gemeinsam geschaffenen Werk ausgegangen werden, wenn die Urheber sich während der Schöpfung gegenseitig beeinflussen, selbst wenn zuvor noch kein Gesamtplan bestanden hat.[121]

Die Begründung einer Miturheberschaft könnte hingegen Schwierigkeiten bereiten, wenn die Softwareentwicklung nicht nur horizontal verteilt, sondern auch zeitlich gestaffelt verläuft. Open Source Projekte geben regelmäßig keine Laufzeit vor, sondern sind auf eine ständige Weiterentwicklung angelegt, die zudem nicht nur auf die ursprüngliche Gesamtidee beschränkt bleibt, sondern mit jeder Funktionserweiterung auf neue Bereiche ausgeweitet wird.

Insofern stellt sich gerade hier die Frage danach, ob die Entwickler, die sich nach der eigentlichen Projektgründung und damit nach der Vereinbarung einer ersten Gesamtkonzeption an der Entwicklungsarbeit beteiligen, ihre Beiträge noch der ursprünglichen Gesamtidee unterordnen können.

[120] Dies ist sicherlich der Regelfall von § 8 UrhG; vgl. insofern nur *Schulze* in: Dreier / Schulze, § 8 Rn. 2 ff.; *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht und deren Bedeutung für das Verwertungsrecht, S. 80 f.

[121] *Thum* in: Wandtke / Bullinger, § 8 Rn. 17, hinsichtlich spontaner musikalischer Improvisation; *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 80.

Dies ist insbesondere dann problematisch, wenn die Urheber sich erst zu einem Zeitpunkt über die Zusammenfügung einzelner Teile verständigen, in dem bereits eines der Werke vollendet wurde. Ob unter diesen Voraussetzungen noch von einem gemeinsamen Schaffen im Sinne von § 8 Abs. 1 UrhG auszugehen ist, erscheint zweifelhaft.

(1)*Zum Zeitpunkt des Zusammenfügens ist bereits ein Werkteil vollendet*

Aus dem Wortlaut des § 8 Abs. 1 UrhG lässt sich hierzu keine eindeutige Wertung entnehmen. Das Tatbestandsmerkmal des *„gemeinsamen Schaffens“* scheint keine Begrenzungen hinsichtlich des Zeitpunkts zu enthalten.

Allerdings sprechen systematische Erwägungen dafür, in solchen Fällen eine Miturheberschaft im Sinne des § 8 Abs. 1 UrhG abzulehnen.[122] Dies ist insbesondere im Hinblick auf die Abgrenzung der Miturheberschaft zur Werkverbindung angebracht. Mit § 9 UrhG besteht nämlich eine Regelung für Konstellationen, in denen mehrere Urheber ein Werk schaffen, sich aber erst nach der Vollendung der jeweiligen Werke über deren Verbindung verständigen.

Dem Gesetz lässt sich insofern die Wertung entnehmen, dass sowohl die jeweiligen Werke als auch die Urheberrechtsbeziehungen getrennt bleiben. Durch die Annahme von Miturheberschaft in solchen Entwicklungskonstellationen würde hierdurch nicht nur die Grenzziehung zwischen Miturheberschaft und Werkverbindung ungenau, sondern im Hinblick auf § 9 UrhG auch die entgegengesetzte Rechtsfolge erreicht. Dies widerspricht jedoch den in §§ 8 und 9 UrhG zum Ausdruck kommenden gesetzlichen Wertungen, weshalb eine Anwendung von § 8 Abs. 1 UrhG aus systematischen Erwägungen ausgeschlossen ist.[123]

Sobald eine Verständigung zwischen den Urhebern erst zu einem Zeitpunkt erfolgt, in dem eines der Werke bereits vollendet ist, kann das Implementieren des Werkes in den Quellcode keine Miturheberschaft mehr begründen, da eine Unterordnung unter den Willen der Miturheber unmöglich wird, nachdem diese zuvor von einer Vollendung des Werks ausgegangen waren.[124] In der Zusammenfügung der Werke kann daher „lediglich“ eine

[122] *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 81 f.
[123] So im Ergebnis auch *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 81 f.
[124] Die Vollendung eines Werkteils steht der Miturheberschaft hingegen nicht entgegen, wenn dieser bereits als Teil eines größeren Ganzen und mit dem Ziel der Ergänzung durch weitere Urheber geschaffen wurde und somit die Voraussetzungen einer Unterordnung unter die Gesamtidee bereits während des Schöpfungsprozesses erfüllt sind; so auch *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 81 Fn. 371.

Werkverbindung im Sinne von § 9 UrhG oder eine Bearbeitung im Sinne von § 3 UrhG liegen.[125]

(2)*Zum Zeitpunkt des Zusammenfügens ist noch kein Werkteil vollendet*

Fraglich bleibt allerdings, ob eine Miturheberschaft auch in denjenigen Fällen ausgeschlossen ist, in denen die jeweiligen Werkteile nach dem Willen der Urheber noch nicht vollendet waren und es wiederum erst nach dem eigentlichen Schöpfungsbeginn zu einer Verständigung zwischen den Urhebern gekommen ist.[126]

Eine Miturheberschaft ist in solchen Fällen zweifellos hinsichtlich der Entwicklungsbeiträge gegeben, die *nach* der Verständigung entstanden sind, da sich die Gesamtidee zu diesem Zeitpunkt bereits auf die „neuen" Werkteile bezieht und sich die späteren Entwicklungen somit unter diese unterordnen.

Problematisch ist hingegen, ob sich die Miturheberschaft auch auf das gesamte Werk, insbesondere auch auf diejenigen (unvollendeten) Teile erstrecken kann, die noch *vor* der Verständigung geschaffen wurden.[127] Hieran könnten Bedenken bestehen, da diese Werkteile zu einem Zeitpunkt erstellt wurden, zu dem noch keine Gesamtidee bestand, unter welche sie hätten untergeordnet werden können.

Die Annahme einer Miturheberschaft am Gesamtprogramm ist letztlich aber auch hinsichtlich der vorbestehenden Werkteile gerechtfertigt, sofern der Urheber diese in die Gesamtidee einbezogen hat und dies dadurch erkenntlich wird, dass an den vorbestehenden Werkteilen Veränderungen im Hinblick auf die Gesamtidee vorgenommen, oder eine Veränderungsmöglichkeit zumindest eingeräumt wird.[128] Sind diese Voraussetzungen gegeben, werden die vorbestehenden Werkteile im Hinblick auf die Vereinbarung einer Gesamtidee – zumindest theoretisch – erneut zur Disposition gestellt und sollen somit zusammen mit den neuen Entwicklungen ein einheitliches Ganzes bilden. Die Voraussetzungen der Miturheberschaft sind dadurch erfüllt, obwohl es erst nach dem

[125] Zu diesen beiden Formen der Urheberschaft sogleich ab S. 36.
[126] Dies Problematisieren bislang nur *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 33 f; und *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 82.
[127] Vgl. *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 81 f.
[128] *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 82; im Ergebnis auch *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 34.

Schöpfungsbeginn – aber eben noch vor der Vollendung des Werks – zu einer Verständigung zwischen den Urhebern gekommen ist.[129]

b.Unverwertbarkeit der Einzelbeiträge

Neben der Unterordnung unter eine Gesamtidee setzt Miturheberschaft weiterhin voraus, dass das hergestellte Werk einheitlich ist, also die jeweiligen Einzelbeiträge der Urheber nicht selbstständig verwertbar sind.[130]

Das Kriterium der Unverwertbarkeit der Einzelbeiträge führt im Rahmen der Softwareentwicklung regelmäßig zu Abgrenzungsschwierigkeiten,[131] da Computerprogramme als Industrieprodukte darauf angelegt sind, mit anderen Elementen eines Datenverarbeitungssystems zusammen zu arbeiten.[132] Insofern ist die Abgrenzung danach, ob lediglich eine funktionsbezogene Kommunikation zwischen den verschiedenen Softwarekomponenten beabsichtigt ist oder die einzelnen Komponenten getrennt verwertbar sind, oftmals schwierig. Allerdings ist gerade im Zusammenhang mit der objektorientierten Programmierung zu berücksichtigen, dass allein die Weiterverwendbarkeit einzelner Objekte oder Klassen nicht dazu führt, dass zwangsläufig die Miturheberschaft an der Gesamtlösung abzulehnen ist. Das Kriterium der Unverwertbarkeit dient vielmehr der Abgrenzung gegenüber der von § 9 UrhG erfassten Werkverbindung.[133] Insofern widerspricht es der Annahme von Miturheberschaft nicht, wenn sich einzelne Bestandteile später auch selbstständig verwerten lassen, sofern sie zum Zeitpunkt der Werkerstellung der gemeinsamen Zielsetzung unterstellt wurden.[134]

Ebenso ist für eine Miturheberschaft nicht erforderlich, dass an sämtlichen Programmteilen gemeinschaftlich gearbeitet wurde,[135] so dass die im Open Source Bereich vorherrschende dezentrale Softwareentwicklung eine Miturheberschaft nicht per se ausschließt. Werden

[129] So auch *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 82.
[130] Hierzu bereits S. 25 m.w.N. in Fn. 106.
[131] Eingehend zur gesonderten Verwertbarkeit vgl. auch Kapitel 5 S. 135 ff.
[132] Vgl. amtl. Begründung BT-Drucks. 12/4022, S. 7 f.
[133] *Loewenheim* in: Schricker, § 9 Rn. 6; *Thum* in: Wandtke / Bullinger, § 9 Rn. 8.
[134] *Schulze* in: Dreier / Schulze, § 8 Rn. 4; *Nordemann* in: Nordemann / Fromm, § 8 Rn. 12; *Loewenheim* in: Schricker, § 8 Rn. 5; *Thum* in: Wandtke / Bullinger, § 8 Rn. 12.
[135] BGH GRUR 2003, 231, 234 - *Staatsbibliothek*; BGH GRUR 1994, 39, 40 – *Buchhaltungsprogramm*; BGHZ 94, 276, 287 f – *Inkasso-Programm*; BGH GRUR 1963, 40, 41 – Straßen – gestern und morgen; zur Miturheberschaft bei vertikaler Arbeitsteilung *Thum* in: Wandtke / Bullinger, § 8 Rn. 9; *Ahlberg* in: Möhring / Nicolini, § 8 Rn. 6 f.

einzelne Komponenten getrennt entwickelt, steht dies der Miturheberschaft am Gesamtprogramm nicht entgegen, sofern die gemeinschaftliche Softwareentwicklung weiterhin angestrebt wird und die Unterordnung unter die Gesamtidee dabei nicht verloren geht.[136] Unter den gleichen Voraussetzungen widerspricht auch die reale Trennbarkeit des Werks, wie sie im Umfeld der Softwareentwicklung durch die Aufspaltung der Quellen in verschiedene Dateien (bspw. Module, Klassen und Objekte) gegeben ist, nicht zwangsläufig der Miturheberschaft an der Gesamtentwicklung.[137]

Gegen eine Miturheberschaft an Programmen kann im Einzelfall jedoch die von § 8 Abs. 1 UrhG implizit vorausgesetzte Einheitlichkeit der Werkkategorie sprechen.[138] Mehrere Werkkategorien kommen dabei in Programmen vor, die über eine grafische Benutzeroberfläche oder sogar ein audiovisuelles Design verfügen. Sofern dies der Fall ist, werden die Quellcodes, die der Benutzeroberfläche bzw. dem audiovisuellen Design zugrunde liegen, nicht vom Schutzbereich der §§ 69 a ff. UrhG umfasst, sondern unterliegen dem Schutz der entsprechenden Werkart.[139] Dies hat zur Folge, dass sich die Miturheberschaft mangels Einheitlichkeit der Werkbeiträge nicht auf den gesamten Quellcode erstrecken kann.

Ergebnis zur Miturheberschaft

Überträgt man diese Ergebnisse, auf die im Open Source Bereich vorherrschenden Entwicklungsstrukturen, ergibt sich folgendes Bild:

Wird ein Patch in den Quellcode eingefügt, der über die erforderliche Schöpfungshöhe verfügt, kann der Werkschöpfer hierdurch Miturheber des Gesamtprogramms werden. Dies

[136] Allgemein zur Softwareentwicklung BGH GRUR 2005, 860, 863 – *Fash 2000*; BGH GRUR 1994, 39, 40 – *Buchhaltungsprogramm*; *Loewenheim* in: Schricker, § 8 Rn. 9; *Marly*, Softwareüberlassungsverträge, Rn. 411; vor dem Hintergrund der Entwicklungsstrukturen in Open Source Projekten vgl. *Jaeger / Metzger*, Open Source Software, Rn. 145; *Schiffner*, Open Source Software, S. 118 f; *Deike*, CR 2003, 9, 15.

[137] Im Hinblick auf die reale Trennbarkeit der Kapitel eines Buches oder Szenen eines Bühnenwerks *Thum* in: Wandtke / Bullinger, § 8 Rn. 8; *Schulze* in: Dreier / Schulze, § 8 Rn. 4; *Nordemann* in: Nordemann / Fromm, § 8 Rn. 3; im Zusammenhang mit den verschiedenen Modulen von Programmen *Schiffner*, Open Source Software, S. 118.

[138] Vgl. amtl. Begründung M. Schulze Materialien 428, aus der ersichtlich wird, dass bei der Verbindung verschiedener Werkkategorien stets eine Werkverbindung im Sinne von § 9 UrhG vorliegen soll; am Beispiel des Textes und der Musik eines Liedes *Loewenheim* in: Schricker, § 8 Rn. 6; *Thum* in: Wandtke / Bullinger, § 8 Rn. 11; *Nordemann* in: Nordemann / Fromm, § 8 Rn. 12; zu grafischen Oberflächen und audiovisuellen Designs *Spindler*, Rechtsfragen bei Open Source, C. Rn. 11.

[139] So die h.M. vgl. nur *Loewenheim* in: Schricker, § 69 a Rn. 7 m.w.N.; a.A. im Hinblick auf Bildschirmmasken, *Hoeren* in: Möhring / Nicolini, § 69 a Rn. 6; sowie OLG Karlsruhe GRUR 1994, 726, 729 f, das darauf hinweist, dass Computerprogramme in jeder Gestalt und Ausdrucksform vom Schutz des § 69 a UrhG erfasst seien.

setzt voraus, dass sich das Programm zum Zeitpunkt der Fehlerbehebung noch in der Entwicklungsphase befunden hat, also nach dem Willen der Miturheber noch nicht fertig gestellt war.[140] In der Praxis wird davon auszugehen sein, solange noch keine Version des Programms vorhanden ist, mit welcher die Urheber ihren Vollendungswillen kenntlich gemacht haben.[141] Miturheberschaft kann insofern nur hinsichtlich derjenigen Werkteile entstehen, die sich nach dem Willen der Urheber noch in der Entwicklung befinden. Dies wird in der Regel damit zum Ausdruck gebracht, dass die entsprechende Programmversion als alpha-, beta- oder auch rc-Release bezeichnet wird.

Die für eine Miturheberschaft erforderliche Unterordnung unter die Gesamtidee findet dadurch statt, dass Fehler an Programmteilen behoben werden, die von der Gesamtidee umfasst waren. Durch die Einsendung eines Patches beteiligt sich der Werkschöpfer an der Weiterentwicklung und Verbesserung des Programms und ordnet sich mit seinen Bemühungen somit konkludent unter die Gesamtidee unter.

Sofern demgegenüber eine Funktionserweiterung eingesandt wird, stellt sich ebenfalls die Frage, ob durch deren Implementierung in das Gesamtprogramm eine Miturheberschaft des Entwicklers begründet wird.

Eindeutig ausgeschlossen ist eine Miturheberschaft jedenfalls an solchen Werkteilen, die zum Zeitpunkt der Implementierung bereits vollendet waren, da eine Unterordnung unter die Gesamtidee dann nicht mehr möglich ist. Sofern also bereits eine Endversion des Programms verfügbar ist, kann eine Miturheberschaft hieran nicht mehr entstehen.[142]

Eine Miturheberschaft des Entwicklers ist weiterhin ausgeschlossen, wenn er eine Funktionserweiterung – losgelöst von den Entwicklungszielen im Projekt – selbstständig und ohne ausdrücklichen Auftrag geschaffen und fertiggestellt hat.[143] In diesen Fällen fehlt es an einer Unterordnung unter die Gesamtidee. Eine „nachträgliche“ Einbeziehung der neuen Funktionserweiterung unter die Gesamtidee scheidet aufgrund deren Vollendung ebenfalls

[140] Hiervon geht *Plaß*, GRUR 2002, 670, 672 aus, sofern das Programm noch als Arbeitsversion oder Experimental Release veröffentlicht wurde.

[141] Von einem Vollendungswillen wird im Open Source Bereich immer dann auszugehen sein, wenn die Programmversion als Stable oder auch offizielles Release bezeichnet wird. Vor der Fertigstellung des Programms finden sich unterschiedliche Bezeichnungen; als Arbeitsversionen und Experimental Releases bezeichnet bei *Plaß*, GRUR 2002, 670, 672; ebenso bei *Spindler*, Rechtsfragen bei Open Source, C. Rn. 9.

[142] Dies schließt indes nicht aus, dass eine Miturheberschaft hinsichtlich derjenigen Werkteile entsteht, die während der Weiterentwicklung des „Stable Releases“, bis zur nächsten vollendeten Programmversion entstehen. Diese Weiterentwicklungen stellen sich zwar im Hinblick auf das „Stable Release“ als Bearbeitung im Sinne von § 3 UrhG dar, allerdings erfolgt die Weiterentwicklung des „Stable Release“ im Open Source Bereich oftmals durch mehrere Urheber und entsprechend einer Gesamtidee, so dass eine Miturheberschaft an der Bearbeitung in Betracht kommt. Ähnlich auch *Koch*, CR 2000, 273, 278; sowie *Omsels* in: FS Hertin, S. 165.

[143] Dies ist etwa bei der Entwicklung von Modulen und Programmbibliotheken denkbar, die unabhängig von dem konkreten Programm auch anderweitig einzusetzen sind; vgl. auch *Schiffner*, Open Source Software, S. 123.

aus. Abhängig von der konkreten Gestalt der Funktionserweiterung kommt insofern lediglich eine Werkverbindung im Sinne von § 9 UrhG oder eine Bearbeitung im Sinne von § 3 UrhG in Betracht.

Durch die Einsendung einer Funktionserweiterung wird eine Miturheberschaft somit häufig nur dann entstehen können, wenn sich diese auf die Implementierung einer Funktionalität bezieht, die sich mit der vom Projekt verfolgten Gesamtidee deckt. Wird eine solche Funktionserweiterung in den Quellcode des Programms implementiert, wird der Urheber der Funktionserweiterung hierdurch Miturheber an denjenigen Teilen des Werks, die zu diesem Zeitpunkt noch nicht vollendet waren.[144]

Sofern sich die Gesamtidee hingegen, vor der Einsendung der Funktionserweiterung noch nicht auf die darin enthaltenen Funktionalitäten erstreckt hat, kommt eine Miturheberschaft lediglich unter engen Voraussetzungen in Betracht.

Voraussetzung für eine Miturheberschaft ist in diesen Fällen, – neben der fehlenden Vollendung der Entwicklungen – dass die Urheber die neue Funktionalität in ihre Gesamtidee einbeziehen[145] und dies dadurch kenntlich machen, dass an den vorbestehenden Werkteilen Veränderungen im Hinblick auf die Funktionserweiterung vorgenommen, oder eine Veränderungsmöglichkeit zumindest eingeräumt wird.

Diese Voraussetzungen werden im Open Source Bereich oft erfüllt sein, indem eine Einbeziehung unter die Gesamtidee dadurch erfolgt, dass im Rahmen der Implementierung der neuen Funktionserweiterung häufig auch Veränderungen am Grundprogramm vorgenommen werden. Hieraus wird ersichtlich, dass ein einheitliches Werk angestrebt und sämtliche Werkteile der „neuen“ Gesamtidee untergeordnet werden.

Für eine Miturheberschaft am Gesamtprogramm spricht zudem, dass die Funktionserweiterungen von deren Urhebern zumeist nicht völlig losgelöst vom eigentlichen Programm geschaffen werden. Der gesamte Schöpfungsprozess erfolgt vielmehr bereits mit dem Ziel der späteren Implementierung. Der Entwickler schafft das Werk ganz bewusst als Teil eines größeren Ganzen, so dass die entwickelte Komponente für sich betrachtet, nicht

[144] Eine Vollendung der Funktionserweiterung steht einer Miturheberschaft in diesen Fällen nicht entgegen, da diese als Teil des Gesamtprogramms und mit dem Ziel der Ergänzung durch weitere Urheber geschaffen wurde und somit bereits während der Schöpfung die Voraussetzungen einer Unterordnung unter die Gesamtidee erfüllt; vgl. auch *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 81, Fn. 71, S. 69 ff, freilich ohne Bezug zur Softwareentwicklung.

[145] In den Entwicklungsplattformen wird dies dadurch erkenntlich, dass sich ein Projektmitglied für den Betreffenden Feature Request für zuständig erklärt und sodann der Request-ID zugeordnet wird.

selbstständig lauffähig ist und zudem nur selten eigenständig Sinn macht.[146] Mit der Offenlegung und Kommentierung[147] seines Quellcodes bezweckt der Entwickler zudem die Weiterentwicklung und Anpassung seiner Funktionserweiterung durch Andere. Insofern liegt zum Zeitpunkt der Einsendung der Funktionserweiterung aus Sicht des Entwicklers noch keine Vollendung seines Werks vor.

Dies wird zudem daraus ersichtlich, dass aufgrund des kollaborativen Entwicklungsprozesses häufig auch andere Entwickler zeitgleich und voneinander unabhängig an der gleichen Quellcodedatei arbeiten. Bevor nun die jeweiligen Veränderungen in die entsprechende Quellcodedatei im Repository übertragen werden können, muss in diesen Fällen die Kompatibilität der verschiedenen Veränderungen zueinander sichergestellt werden.[148] Hierfür sind zwangsläufig Anpassungen an den jeweiligen Werkteilen erforderlich,[149] worüber sich die Entwickler im Regelfall auch bewusst sind. Insofern kann ein Entwickler im Open Source Bereich nur bei der Entwicklung vollständig abgeschlossener Module davon ausgehen, dass andere Entwickler nicht bereits im Zuge der Implementierung, spätestens aber im Laufe der Programmweiterentwicklung Änderungen an seinem Werk vornehmen werden.

Insgesamt ist daher festzuhalten, dass Entwickler im Open Source Bereich zumeist einen gemeinsamen Schaffensprozess anstreben, wie er eigentlich für die Miturheberschaft kennzeichnend wäre. Zweifel am Vorliegen einer Miturheberschaft ergeben sich allein aus dem Umstand, dass die Verständigung über die Gesamtidee erst nach dem eigentlichen Schöpfungsbeginn erfolgt. Gleichwohl ist es vertretbar eine Miturheberschaft der beteiligten Entwickler auch dann anzunehmen, wenn sowohl das Grundprogramm, als auch die Funktionserweiterung als unvollendet zu betrachten sind, und die Funktionserweiterung nicht bereits vor dem Schöpfungsbeginn, sondern erst während deren Implementierung, unter die Gesamtidee untergeordnet wird.[150]

---

[146] Eine Ausnahme wird für solche Funktionserweiterungen zu machen sein, die beispielsweise aufgrund der in ihnen enthaltenen Möglichkeit, sie über eine Standardschnittstelle einzubinden, derart Modular sind, dass sie unabhängig vom konkreten Programm geschaffen wurden und in mehreren Programmen einzusetzen sind. Einzelheiten hierzu unter Kapitel 5 S. 135 ff.

[147] Der Quelltext lässt sich letztlich erst durch solche Kommentierungen nachvollziehen und verändern. Diese sind im Quelltext an sämtlichen Stellen zur Erläuterung enthalten, an denen Variablen bestimmt und Werte gesetzt werden. Aus funktionaler Sicht verändern die Kommentierungen hingegen nichts am Quelltext, da sie regelmäßig bei dessen Übersetzung in ein ausführbares Programm durch den Compiler gelöscht werden, aber auch ansonsten nicht zur Ausführung gelangen.

[148] Dies wird offensichtlich, wenn man sich vorstellt, dass ein Entwickler Veränderungen an einem Teil des Quellcodes durchführt, deren ursprüngliche Form jedoch von einem Programmzusatz eines anderen Entwicklers vorausgesetzt wird.

[149] Zum Entwicklungsprozess und zur Arbeit mit dem zentralen Repository vgl. *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 241, http://freie-software.bpb.de/Grassmuck.pdf.

[150] Eingehend hierzu bereits oben S. 30.

4.Bearbeitung

Wie bereits dargestellt wurde, sind Open Source Projekte dadurch gekennzeichnet, dass die Programme horizontal verteilt und zeitlich gestaffelt entwickelt werden. Gerade die sukzessiv erfolgende Programmentwicklung führt hierbei zu Abgrenzungsschwierigkeiten zwischen Miturheberschaft und Bearbeitung. Eine Miturheberschaft am Gesamtprogramms ist dabei anzunehmen, wenn sich die Entwicklungsarbeit auf eine Programmversion bezog, die noch nicht vollendet war.

Eine Bearbeitung wird hingegen vorliegen, wenn Veränderungen an einer Programmversion vorgenommen werden, die bereits als offizielles Release[151] veröffentlicht wurde. In diesen Fällen haben die Urheber des Programms erkenntlich gemacht, dass sie von einer Vollendung des Programms ausgehen,[152] weshalb die nachträgliche Weiterentwicklung nicht mehr unter die Gesamtidee untergeordnet werden kann. Bei Erreichen der erforderlichen Schöpfungshöhe steht dem Entwickler „lediglich“ ein eigenes Bearbeiterurheberrecht zu, das gem. § 3 UrhG urheberrechtlichen Schutz genießt.

Dies führt im Open Source Bereich dazu, dass neben den Miturhebern des Programms häufig auch Bearbeiter vorhanden sind, die eigenständige Rechte an der späteren Version haben. Als Folge des kollaborativen Entwicklungsprozesses ergibt sich für das Open Source Entwicklungsmodell folgendes Bild.

Die Entwickler, die bis zur ersten Endversion in schöpferischer Weise beteiligt waren, werden in der Regel Miturheber dieser Programmversion sein.

Die Entwickler, die wiederum in einem gemeinschaftlichen Schaffensprozess das Programm bis zur nächsten Endversion weiterentwickelt haben, sind im Hinblick auf ihre Entwicklungen Miturheber,[153] im Verhältnis zum Gesamtprogramm jedoch „lediglich“ Bearbeiter der vorbestehenden Endversion.

[151] Im proprietären Umfeld wird die Endversion eines Programms häufig als Release to Manufacturing (RtM) bezeichnet.
[152] Im Rahmen von Open Source Projekten erfolgt dies in der Regel durch eine Zweiteilung der Entwicklungen. Während der Quellcode des offiziellen Releases eingefroren wird, besteht daneben eine Entwicklerversion, an der die Weiterentwicklungen bis zum nächsten offiziellen Release durchgeführt werden; vgl. *Grassmuck*, Freie Software Zwischen Privat- und Gemeineigentum, S. 255, http://freie-software.bpb.de/Grassmuck.pdf.
[153] Ähnlich in der Einschätzung *Küng*, MR 2004, 21, 26.

5.Werkverbindung

Neben der Miturheberschaft und der hiervon abzugrenzenden schöpferischen Bearbeitung eines Open Source Programms lässt sich in der Praxis oftmals auch eine urheberrechtlich relevante Werkverbindung im Sinne von § 9 UrhG vorfinden. Davon ist auszugehen, wenn mehrere Urheber ihre Werke, die jeweils gesondert geschaffen wurden und eigenständig urheberrechtlichen Schutz genießen, nachträglich zur gemeinsamen Verwertung verbinden.[154]

Im Open Source Bereich findet sich eine solche Werkverbindung vor allem bei der Einbindung von Programmbibliotheken oder objektorientierten Klassenbibliotheken.[155] Aber auch bei der Verbindung von Programmen und audiovisueller Benutzeroberfläche ist von einer Werkverbindung auszugehen.[156]

Auf die Urheberrechte an den jeweiligen Programmteilen hat eine solche Werkverbindung keine Auswirkung. Durch die Verbindung der Werke entsteht zwischen den Urhebern lediglich eine schuldrechtliche Bindung im Sinne von §§ 9, 69 a Abs. 4 UrhG.

Zusammenfassung

Für die Urheberschaft an Open Source Programmen ergeben sich daher eine Reihe an unterschiedlichen Konstellationen der Rechtsinhaberschaft.

In den Anfangsphasen[157] werden sich Open Source Projekte oftmals durch eine Miturheberschaft der beteiligten Entwickler auszeichnen.[158] Eine solche ist zunächst bei den Entwicklern festzustellen, die gemeinsam eine erste (alpha-) Version des Programms entwickelt haben. Aber auch danach kommt eine Miturheberschaft der Entwickler in Betracht, die sich durch die Zusendung von Patches und Funktionserweiterungen an der Weiterentwicklung des Programms bis zur ersten Endversion beteiligen.

---

[154] Allgemein zu den Voraussetzungen der Werkverbindung vgl. nur *Loewenheim* in: Schricker, § 9 Rn. 4, m.w.N.; im Hinblick auf die Entwicklungsstrukturen im Open Source Bereich *Deike*, CR 2003, 9, 15; *Marly*, Softwareüberlassungsverträge, Rn. 420.

[155] Für objektorientierte Klassenbibliotheken vgl. *Koch*, CR 2000, S. 273, 277.

[156] Zumindest nach h.M. liegt in audiovisuellen Benutzeroberflächen eine vom restlichen Computerprogramm zu trennende Werkart vor, so dass die gemeinsame Verwertung mit dem Computerprogramm eine Werkverbindung im Sinne von § 9 UrhG darstellt; vgl. nur *Loewenheim* in: Schricker, § 69 a Rn. 7 m.w.N.; zweifelnd in Bezug auf Bildschirmmasken, *Hoeren* in: Möhring / Nicolini, § 69 a Rn. 6; sowie OLG Karlsruhe GRUR 1994, 726, 729 f, das darauf hinweist, dass Computerprogramme in jeder Gestalt und Ausdrucksform vom Schutz des § 69 a UrhG erfasst seien.

[157] Damit ist der Zeitraum bis zum ersten „stable Release“ umfasst.

[158] Zu diesen Ergebnis kommen auch *Omsels* in: FS Hertin, S. 166; *Plaß*, GRUR 2002, 670, 672; *Schiffner*, Open Source Software, S. 121 f.

Sofern eingesandte Patches in den Quellcode aufgenommen werden, hängt die Miturheberschaft maßgeblich von der erforderlichen Schöpfungshöhe ab. Behandelt der Patch, wie zumeist, das Auffinden und Korrigieren kleinerer Fehler, reicht dies für eine urheberrechtliche Schutzwürdigkeit nicht aus. Aus diesem Grund wird das Einfügen von Patches nur in Ausnahmefällen zu einer Miturheberschaft am Gesamtprogramm führen.

Im Gegensatz hierzu werden eingesendete Funktionserweiterungen häufiger die erforderliche Schöpfungshöhe erreichen. Allerdings ist für eine Miturheberschaft stets gesondert zu prüfen, ob die entsprechende Funktionserweiterung der Gesamtidee untergeordnet wurde. Eine solche Unterordnung kann nur dann vorliegen, wenn weder die Funktionserweiterung, noch das Gesamtprogramm zum Zeitpunkt des Einfügens in den Quellcode vollendet waren. Sollen die vor und nach der Verständigung geschaffenen Werkteile einen einheitlichen Werkbeitrag bilden und ist eine Bearbeitungsmöglichkeit der Werkteile im Rahmen der Implementierung vorgesehen, so ist von einer Miturheberschaft am gesamten Werk auszugehen.

Während der weiteren Projektlaufzeit werden schöpferische Entwicklungsbeiträge hingegen zunehmend als Bearbeitungen im Sinne von § 3 UrhG einzustufen sein. Da allerdings auch die Weiterentwicklung der Programme in einem gemeinschaftlichen Entwicklungsprozess erfolgt, wird häufig auch das Bearbeiterurheberrecht mehreren Miturhebern gemeinschaftlich zustehen.

Im Rahmen der sukzessiven Softwareentwicklung in Open Source Projekten wird es also häufig zu mehreren Gruppen von Miturhebern kommen, die jeweils Bearbeiter der vorbestehenden Endversionen sind.

Hält man sich die Zahl der Entwickler vor Augen, die an der Fortentwicklung von Open Source Programmen beteiligt sind, wird ersichtlich, wie komplex die Urheberschaft an Open Source Programmen ausfallen kann. Zu dieser schwierigen urheberrechtlichen Zuordnung kommen zudem noch die internationalen Probleme hinzu, die durch die dezentrale Verteilung der Entwickler hervorgerufen werden.[159]

Um diese komplexen Beteiligungskonstellationen im Rahmen von größeren Projekten auch nach längerer Laufzeit noch korrekt nachvollziehen zu können, ist eine aufwendige Dokumentation der jeweiligen Entwicklungsschritte unumgänglich. Viele Projekte verwenden

[159] Diese werden in Kapitel 4 ab S. 64 thematisiert.

zu diesem Zweck Programme zur Versionsverwaltung,[160] die eine Nachverfolgbarkeit der Veränderungen gewährleisten sollen.

Diese funktionieren regelmäßig in der Art, dass sämtliche Änderungen automatisch protokolliert und später auf Differenzen zwischen verschiedenen Versionen ausgewertet werden können.

Die Urheberschaft lässt sich jedoch trotz dieser Protokollierung nicht zwingend anhand eines Versionsverwaltungssystems nachvollziehen. Protokolliert wird nämlich nur derjenige, der tatsächlich Schreibzugriff auf den Quellcode hat und Änderungen an diesem herbeiführt. Der Schreibzugriff auf den Quellcode im Repository steht jedoch regelmäßig nur wenigen Projektmitgliedern zu,[161] weshalb diese durch das Versionsverwaltungssystem protokolliert werden, unabhängig davon, ob sie auch Urheber des eingefügten Codes sind. Die Urheberschaft des eingearbeiteten Teils lässt sich somit häufig nicht anhand der Protokollierung des Versionsverwaltungssystems nachvollziehen. Hierfür ist vielmehr erforderlich, dass der entsprechende Bearbeiter des Quellcodes zusätzliche Informationen zur Urheberschaft dokumentiert. Eine solche Dokumentation erfolgt bislang nur in größeren Projekten.[162] Verwirklicht wird sie dadurch, dass der Bearbeiter in einen Kommentar im Quellcode bzw. im Revision-Log zusätzlich noch die Patch- bzw. Feature-ID oder die Mail-Adresse desjenigen aufführt, von dem dieser Teil eingesandt wurde.

Eine andere Möglichkeit die Urheberschaft an Open Source Projekten nachvollziehbar zu gestalten besteht darin, sich vor dem Einfügen von „Fremdcode“ die Verwertungsrechte hieran einräumen zu lassen.[163]

[160] Im Umfeld von Open Source Entwicklungen sind besonders das sog. Concurrent Versions System (CVS) und Subversion (SVN) verbreitet.

[161] Einzelheiten zum Einsatz von Concurrent Versions Systemen, vgl. *Köhntopp / Köhntopp, u.a.*, DuD 2000, 508, 510 f.

[162] Vgl. beispielsweise das Change Log des Linux Kernels, http://www2.kernel.org/pub/linux/kernel/v2.6/ChangeLog-2.6.19.2, dass sämtliche Autoren, sowie die eingefügten Werkteile aufführt. Daneben findet sich ein Hinweis auf die Urheber aber auch jeweils zu Beginn der einzelnen Dateien, die zum Linux-Kernel gehören.

[163] So wird z.B. bei der von Sun initiierten Entwicklung von Open Office verfahren, bei der diejenigen Entwickler, die ein Patch einsenden wollen, zuvor schriftlich die Verwertungsrechte einräumen müssen; vgl. das Copyright Assignment von Sun, http://www.openoffice.org/licenses/jca.pdf.

Kapitel 3

D. Vertragsrechtliche Besonderheiten im Open Source Bereich

In der Literatur lassen sich mittlerweile einige Untersuchungen finden, die sich mit vertragsrechtlichen Fragen im Umfeld von Open Source Software beschäftigen. Dabei liegt ein thematischer Schwerpunkt bei der Frage nach der wirksamen Einbeziehung der GPL als AGB[164] in den Erwerbsvorgang,[165] welcher insbesondere gegenüber Verbrauchern aufgrund des in englischer Sprache verfassten Lizenztextes bedenklich erscheint,[166] vorliegend jedoch nicht vertieft werden soll, da jedenfalls in den hier interessierenden Fällen, in denen Unternehmen auf Erwerberseite stehen, die Möglichkeit der Einbeziehung englischsprachiger AGB gemeinhin anerkannt ist.[167]

Ein weiterer Schwerpunkt zahlreicher Untersuchungen betrifft die Frage der Wirksamkeit einzelner Klauseln der GPL vor dem Hintergrund der §§ 305 ff BGB, wobei vor allem der in

---

[164] Dass es sich bei der GPL um AGB handelt, ist wohl unbestritten; vgl. nur LG Frankfurt/M CR 2006, 729, 731; LG München CR 2004, 774, 775; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 45; *Koch*, CR 2000, 333, 339; *Omsels* in: FS Hertin, S. 147; *Jaeger / Metzger*, Open Source Software, Rn. 179; *Marly*, Softwareüberlassungsverträge, Rn. 429; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 684; *Plaß*, GRUR 2002, 670, Rn. 676; *Metzger / Jaeger*, GRUR Int. 1999, 839, 846.

[165] Eingehend zur wirksamen Einbeziehung der GPL in den Softwareüberlassungsvertrag, vgl. nur *Spindler*, Rechtsfragen bei Open Source, Rn. 44 ff; *Omsels* in: FS Hertin, S. 147 ff. Demgegenüber differenziert die wohl h.M. (vgl. vor allem *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 681 ff; *Marly*, Softwareüberlassungsverträge, Rn. 429; *Jaeger / Metzger*, Open Source Software, Rn. 180 ff) zu Recht zwischen der Einbeziehung der GPL in den Softwareüberlassungsvertrag – an deren Wirksamkeit aufgrund der von § 305 Abs. 2 BGB vorgeschriebenen Hinweispflicht, sowie der Möglichkeit einer zumutbaren Kenntnisnahme oftmals Zweifel berechtigt sein werden – und der davon zu trennenden (Einzelheiten hierzu sogleich ab S. 42) rechtsgeschäftlichen Einräumung der Bearbeitungs- und Vertriebsrechte, die mitunter erst zu einem späteren Zeitpunkt stattfindet und in deren Rahmen die Voraussetzungen der §§ 305 BGB jedenfalls dann erfüllt werden, wenn sie in der Form überlassen wurden, wie es von den Bestimmungen der GPL vorgesehen ist, vgl. *Marly*, Softwareüberlassungsverträge, Rn. 429.

[166] Hier wird es im Rahmen der Einbeziehung in den Softwareüberlassungsvertrag – nicht aber im Rahmen des Nutzungsvertrags – wohl in der Regel an der Möglichkeit der zumutbaren Kenntnisnahme im Sinne von § 305 Abs. 2 BGB fehlen, vgl. u.a. *Spindler*, Rechtsfragen bei Open Source, Rn. 53 f; *Omsels* in: FS Hertin, S. 149; *Plaß*, GRUR 2002, 670, 678 f; a.A. *Schiffner*, Open Source Software, S. 185 f; *Sester*, CR 2000, 797, 805; differenzierend *Marly*, Softwareüberlassungsverträge, Rn. 429 und 1408.

[167] Dies liegt daran, dass die englische Sprache, gerade im Bereich der EDV, im internationalen Geschäftsverkehr als üblich angesehen werden kann, weshalb von der im Unternehmensverkehr – unabhängig von der Geltung des § 305 Abs. 2 BGB – Geforderten (vgl. hierzu BGH NJW-RR 1989, 1104; BGH NJW 1988, 1210, 1212; BGHZ 102, 289, 304; *Heinrichs* in: Palandt, § 305 Rn. 54) Möglichkeit zumutbarer Kenntnisnahme stets auszugehen ist, vgl. *Schmidt* in: Lehmann, Die Kontrolle Allgemeiner Geschäftsbedingungen in Programmüberlassungsverträgen, Kap. XV Rn. 22; sowie *Marly*, Softwareüberlassungsverträge, Rn. 1411, jeweils m.w.N.; ebenso *Spindler*, K&R 2004, 528, 532.

Ziff. 11 und 12 GPL enthaltene Haftungs- und Gewährleistungsausschluss problematisiert wird,[168] dessen Unwirksamkeit im deutschen Recht gemeinhin anerkannt ist.

Im Hinblick darauf, dass sich die vorliegende Untersuchung vertieft mit der Reichweite des sog. „viralen Effekts" beschäftigt, wäre im Zusammenhang mit den §§ 305 ff BGB einzig die Wirksamkeit des in Ziff. 2 GPL enthaltenen Copyleft-Effekts interessant, der allerdings zu Recht weder von der Rechtsprechung[169] noch – von einer Ausnahme abgesehen –[170] von der Literatur als bedenklich angesehen wird[171] und von dessen Wirksamkeit daher auszugehen ist.[172]

Gerade im Kontext der nachfolgenden Untersuchung, welche sich mit der Frage nach dem anwendbaren Recht innerhalb der unterschiedlichen Beziehungen im Open Source Modell beschäftigt,[173] beschränkt sich die Darstellung der vertragsrechtlichen Besonderheiten daher vorliegend auf eine vertragstypologische Einordnung des Open Source Modells.

---

[168] Vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 219 ff; *Spindler*, Rechtsfragen bei Open Source, D. Rn. 14 ff; *Marly*, Softwareüberlassungsverträge, Rn. 440 ff; *Schiffner*, Open Source Software, S. 241 ff; *Omsels* in: FS Hertin, S. 147 ff.

[169] Ziff. 2 GPL wurde bislang einhellig für wirksam erachtet, vgl. mit ausführlicher Erörterung LG Frankfurt/M CR 2006, 729, 731 ff; sowie LG Berlin CR 2006, 735; LG München CR 1994, 774, 776.

[170] Vgl. *Spindler*, K&R 2004, 528, 533; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 126; sowie der die Wirksamkeit der Ziff. 2 GPL Hinblick auf das in § 307 Abs. 1 S. 2 BGB enthaltenen Transparenzgebot bezweifelt. Hiergegen bestehen allerdings erhebliche Bedenken, die sich vor allem daraus ergeben, dass die Transparenzanforderungen nicht überspannt werden dürfen und der Verwender daher lediglich dazu verpflichtet ist die Regelungen im Rahmen des Möglichen klar und verständlich zu formulieren (vgl. BGH NJW 3114, 3116 m.w.N.). Aufgrund der Vielzahl an denkbaren Kombinationsformen zwischen Open Source Programmen und proprietären Entwicklungen erscheint es hingegen nicht möglich zu sein, eine Copyleft-Regelung zu erstellen, die für sämtliche Fallgruppen eindeutig ist; hierauf weist zu Recht auch *Jaeger / Metzger*, Open Source Software, Rn. 188, hin. Im Übrigen wird mit dem Begriff des *„derivative works"* an dem die Reichweite der Lizenzierungspflicht in der GPL maßgeblich festgemacht ist, lediglich ein unbestimmter Gesetzesbegriff übernommen (vgl. 17 U.S.C. § 101), was gemeinhin für zulässig erachtet wird, vgl. nur *Heinrichs* in: Palandt, § 307 Rn. 18 m.w.N.

[171] Vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 188, welche die Einhaltung des Transparenzgebots eingehend darlegen; ebenso in der Einschätzung *Wiebe / Heidinger*, MR 2006, 258, 258.

[172] Selbst wenn man in Ziff. 2 GPL einen Verstoß gegen das in § 307 Abs. 1 S. 2 enthaltene Transparenzgebot sehen wollte (so *Spindler*, Rechtsfragen bei Open Source, C. Rn. 126), hätte dieser nicht etwa zur Folge, dass der Erwerber die Open Source Programme sodann ohne den Copyleft-Effekt nutzen könnte. Vielmehr wurde in der Rechtsprechung (vgl. LG Frankfurt/M CR 2006, 769, 732 f; sowie LG München CR 2004, 774, 776) und auch in der Literatur (*Jaeger / Metzger*, Open Source Software, Rn. 188)) zu Recht darauf hingewiesen, dass die Regelung der Ziff. 2 GPL derart bestimmend für die Einräumung der Nutzungsrechte sei, dass deren Unwirksamkeit zu einer Gesamtnichtigkeit des Vertrages führen würde, weshalb dem Nutzer ausschließlich Nutzungs- nicht aber Entwicklungs- und Vertriebsrechte zustünden; a.A. wohl Hoeren, Anm. zum Urteil des LG München CR 2004, 774, 777.

[173] Einzelheiten hierzu werden im Kapitel 4 ab S. 64 behandelt.

I.Rechtsgeschäftliche Grundlagen des Erwerbs GPL-lizenzierter Software

Trotz zahlreicher Untersuchungen, die sich bereits um eine vertragstypologische Einordnung des Open Source Modells bemüht haben,[174] hat sich in der Literatur hierzu noch keine einheitliche Auffassung herausgebildet. Lediglich im Grundsatz besteht eine weitgehende Einigkeit darüber, dass dem Erwerb von Open Source Software kein einheitlicher „Open Source Vertrag“ zugrunde liegt,[175] sondern zwischen dem Erwerb der Programmkopie und dem der Nutzungsrechte zu unterscheiden ist.[176] Dies folgt nicht zuletzt auch aus Ziff. 6 GPL und führt bei der Weitergabe einer Programmkopie dazu, dass nicht etwa Unterlizenzen vom Weitergebenden vergeben werden, sondern dieser – soweit er nicht selbst Urheber des Programms ist – im Hinblick auf die Nutzungsrechte als Bote des Urhebers tätig wird.[177] In der Praxis stellt diese Verbreitungsform den Regelfall dar, da der Nutzer die Programmkopie zumeist nicht unmittelbar vom Urheber erhält.

So ist beispielsweise der Distributor, von dem der Nutzer die Programmkopie erwirbt, regelmäßig nicht selbst Urheber der vertriebenen Programme, weshalb es einer gesonderten Rechtseinräumung durch die Urheber bedarf. Ebenso stellt sich die Sachlage dar, wenn der Nutzer die Software von einem Downloadserver herunterlädt, der nicht von den Urhebern betrieben wird. Auch hier erhält der Nutzer die Programmkopie von einer Person, die nicht selbst Rechtsinhaber ist, weshalb er für die Rechtseinräumung auf einen weiteren Vertrag mit den Urhebern angewiesen ist.

Aber selbst wenn der Nutzer das Programm direkt von der Projektseite bezieht, folgt hieraus nicht zwingend, dass er die Programmkopie von demjenigen erhält, der ihm auch die Nutzungsrechte einräumt. Zwar kommt in diesen Fällen grundsätzlich ein direkter Erwerb der Software vom Urheber in Betracht, wird in der Praxis jedoch gleichwohl nur die Ausnahme bilden, da sich die kollaborative Softwareentwicklung im Open Source Bereich dadurch auszeichnet, dass die Urheberschaft am Programm nicht zugleich auch an die

[174] Speziell vor dem Hintergrund der GPL, vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 171 ff; *Schiffner*, Open Source Software, S. 200 ff; *Spindler*, Rechtsfragen bei Open Source, D. 1 ff; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 347 ff; allgemein zu den schuldrechtlichen Beziehungen in Open Source Modellen *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 740 ff.

[175] Von einem einheitlichen Vertrag gehen lediglich *Sester*, CR 2000, 797, 800 ff; und wohl auch *Koch*, CR 2000, 333, 335, aus.

[176] Eine Unterscheidung zwischen dem Erwerb der Programmkopie und dem der Nutzungsrechte wird vorgenommen bei: *Jaeger / Metzger*, Open Source Software, Rn. 171 ff; *Küng*, MR 2004, 21, 23; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn 761 ff; von drei Rechtsgeschäften (Programmerwerb, sog. „kleine Open Source Berechtigung“ und sog. „große Open Source Berechtigung) gehen *Schiffner*, Open Source Software, S. 217 ff; sowie *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 311 ff aus.

[177] *Spindler / Wiebe*, CR 2003, 873, 874.

Projektmitgliedschaft gekoppelt ist.[178] Insofern werden häufig auch Entwickler, die außerhalb des Projekts stehen, durch das Einsenden von Patches oder Funktionserweiterungen, Rechte an der jeweiligen Programmkopie besitzen.[179] Selbst beim direkten Download von der Projektseite wird die Programmkopie deshalb – zumindest hinsichtlich dieser Programmteile – von Dritten erworben.

Beim Erwerb von Open Source Software ist sich die Literatur deshalb überwiegend darüber einig, dass regelmäßig nicht von einem einheitlichen Gesamtvertrag auszugehen ist, sondern zwischen dem Erwerb der Programmkopie auf der einen Seite und der Rechtseinräumung durch die Urheber auf der anderen Seite zu unterscheiden ist.[180]

Umstritten ist demgegenüber die Frage, ob der Abschluss der GPL bereits für die bestimmungsgemäße Nutzung der Programmkopie erforderlich wird.[181] Die auftretenden Differenzen sind dabei, vor allem auf ein unterschiedliches Verständnis von § 69 d UrhG zurückzuführen.

Gegen das Erfordernis, dass der Nutzer bereits für die bestimmungsgemäße Nutzung der Programmkopie auf den Abschluss der GPL angewiesen ist, scheint zwar Ziff. 0 GPL[182] zu sprechen, wonach die bloße Nutzung der Software vom Anwendungsbereich der Lizenz ausgeschlossen sein soll. Gleichwohl geht ein Teil der Literatur davon aus, dass der Nutzer, neben dem Vertrag, welcher der Programmüberlassung zugrunde liegt, auch für die bestimmungsgemäße Nutzungsmöglichkeit der Open Source Software, auf eine Nutzungsrechtseinräumung durch die Urheber angewiesen ist.[183] Dieser Ansicht liegt ein Verständnis von § 69 d UrhG zugrunde, wonach die Vorschrift nur denjenigen begünstigt, der bereits zuvor durch den Rechtsinhaber berechtigt wurde.[184] Es würde somit nicht ausreichen, dass der Nutzer die GPL-lizenzierte Programmkopie von einem Dritten rechtmäßig erhalten

[178] Zur Urheberschaft siehe bereits oben S. 17.
[179] Vgl. die Ausführungen zu den Entwicklungsbeiträgen in Open Source Projekten S. 17.
[180] *Jaeger / Metzger*, Open Source Software, Rn. 171 ff; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 761 ff; *Plaß*, GRUR 2002, 670, 677.
[181] So wird dies vertreten von *Schiffner*, Open Source Software, S. 218 ff; sowie *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 311 ff; ähnlich auch *Deike*, CR 2003, 9, 13, indem er davon ausgeht, dass der Erwerber bereits für die erste Kopie, die beim Download des Programms entsteht auf den Abschluss der GPL angewiesen sei.
[182] Ziff. 0 GPL: *„Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted…“*
[183] Vgl. *Schiffner*, Open Source Software, S. 218 ff; sowie *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 311 ff; ähnlich auch *Deike*, CR 2003, 9, 13, der zwar nicht ausdrücklich auf die bestimmungsgemäße Nutzung eingeht, allerdings bereits die der Programmausführung vorgelagerte Erstellung der Programmkopie auf dem Computer für zustimmungsbedürftig hält.
[184] Speziell im Hinblick auf den Erwerb von Open Source Software vgl. *Schiffner*, Open Source Software, S. 218 ff; sowie *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 311 ff; allgemein zum Erfordernis einer Berechtigung des Nutzers durch einen Nutzungsvertrag *Pres*, Gestaltungsformen urheberrechtlicher Softwarelizenzverträge, S. 120 ff; *Moritz*, MMR 2001, 94, 95; *Koch*, Computer-Vertragsrecht Rn. 1952.

hat. Um das Programm auch tatsächlich nutzen zu können, wäre der Nutzer immer auch auf die Einräumung von Nutzungsrechten durch die Rechtsinhaber angewiesen.

Ob dies allerdings von § 69 d UrhG gefordert wird, erscheint zweifelhaft. Es geht dabei um die Frage, ob der zur Verwendung des Programms Berechtigte im Sinne von § 69 d UrhG zur bestimmungsgemäßen Nutzung der Software noch zusätzlich ein Nutzungsrecht vom Rechtsinhaber eingeräumt bekommen muss.[185]

Auf ein solches Erfordernis könnte zunächst der Wortlaut des § 69 d UrhG hindeuten, indem auf den zur „…*Verwendung (...) des Programms Berechtigten*…“ Bezug genommen wird. Eine ähnliche Formulierung findet sich auch in § 31 Abs. 2 UrhG, in dem derjenige als zur Nutzung „*berechtigt*“ gilt, dem ein einfaches Nutzungsrecht eingeräumt wurde. Insofern liegt die Vermutung nahe, dass auch im Rahmen des § 69 d UrhG eine vorherige Einräumung eines (einfachen) Nutzungsrechts zu fordern ist.

Zweifel an einer solchen Auslegung des § 69 d UrhG bestehen indes vor allem im Hinblick auf das Zusammenwirken mit dem gesetzlich bindenden[186] Erschöpfungsgrundsatz.[187] Sofern im Rahmen des § 69 d UrhG selbst für die schlichte Programmbenutzung eine Berechtigung im Sinne eines Nutzungsrechts gefordert würde, hätte dies nämlich dessen weitgehende Wirkungslosigkeit zur Folge.[188] Zwar könnte die durch den Erschöpfungsgrundsatz gewährleistete Weiterverbreitung der Programmkopie an Dritte nicht verhindert, jedoch deren Nutzung von der vorherigen Rechtseinräumung durch den Urheber abhängig gemacht werden. Die vom Erschöpfungsgrundsatz bezweckte Verkehrsfähigkeit würde sich danach lediglich auf die Programmdateien beziehen, ohne dass die zur bestimmungsgemäßen Nutzung erforderlichen Rechte vorhanden wären. Hierdurch würde der Erwerber zwar ein technisches Grundgerüst[189] erhalten, dass mangels Nutzbarkeit aber wertlos ist. Eine Konsequenz, die im Hinblick auf den Erschöpfungsgrundsatz nicht akzeptabel ist.[190]

Der Verkehrsschutz, der mit der gesetzlich zwingenden Anordnung der Erschöpfung des Verbreitungsrechts in § 69 c Abs. 3 S. 2 UrhG verfolgt wurde, lässt sich somit nur dadurch

[185] Eingehend zum Normcharakter des § 69 d UrhG, vgl. *Pres*, Gestaltungsformen urheberrechtlicher Softwarelizenzverträge, S. 120 ff; *Haberstumpf* in: Lehmann, Der urheberrechtliche Schutz von Computerprogrammen, Kap. II Rn. 159; *Koch*, Computer-Vertragsrecht Rn. 1952.
[186] *Loewenheim* in: Schricker, § 69 c Rn. 32.
[187] Allgemein zum Erschöpfungsgrundsatz unter Einbeziehung der geschichtlichen Hintergründe vgl. Reimer, GRUR Int. 1972, 221, 224 ff; *Berger*, AcP 2001, 411, 414 ff.
[188] Als unerträgliche Behinderung des freien Warenverkehrs bezeichnet bei *Loewenheim* in: Schricker, § 17 Rn. 36; ebenso *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 428; *Junker / Benecke*, Computerrecht, Rn. 71.
[189] So vertritt dies *Schiffner*, Open Source Software, S. 140 ff.
[190] Im Ergebnis ebenso *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 430 f; *Baus*, MMR 2002, 14, 16.

sicherstellen, dass neben dem reinen Materialwert der Programmdateien, auch deren bestimmungsgemäße Nutzbarkeit umfasst wird. Um dem Zweck des Erschöpfungsprinzips somit Rechnung zu tragen, muss es daher ausreichen, wenn der Nutzer die Programmkopie rechtmäßig erworben hat.[191]

Eine solche Auslegung wird zudem durch die Entstehungsgeschichte der Computerprogrammrichtlinie bestätigt, auf die § 69 d UrhG letztlich zurückzuführen ist. Dabei ist vor allem die Entwicklung des Wortlauts des „*…zur Verwendung eines Vervielfältigungsstücks des Programms Berechtigten…*“ relevant, da dieser Aufschluss darüber gibt, wer nach Auffassung des Gesetzgebers als Berechtigter im Sinne der Vorschrift anzusehen sein sollte.

So wurde in der Begründung des Kommissionsentwurfs noch davon ausgegangen, dass eine Software, die verkauft oder überlassen wurde, danach ohne Zustimmung des Rechtsinhabers verwendet werden dürfe.[192] Auch in Art. 5 des Richtlinienvorschlags des Rates lässt sich diese Wertung wieder finden, indem nicht auf einen zur Verwendung des Programms „*Berechtigten*“, sondern auf den „*Erwerber“* einer Kopie des Computerprogramms abgestellt wird. Insofern wurde gerade nicht an eine Berechtigung im Sinne einer Rechtsübertragung angeknüpft, sondern der Erwerb einer Kopie für ausreichend befunden.

Der Begriff des Erwerbers wurde letztlich erst im Zuge der deutschen Umsetzung durch den Begriff des Berechtigten ersetzt.[193] Hintergrund für diese Abänderung war allerdings nicht das Bestreben zusätzliche Anforderungen an den Anwendungsbereich des § 69 d UrhG zu stellen. Vielmehr sollte die, aus Sicht des deutschen Gesetzgebers, unklare Wortwahl des „*rechtmäßigen Erwerbers*“ verdeutlicht werden.[194] Bezweckt wurde eine Klarstellung dahin gehend, dass der Inhaber der Programmkopie nicht nur durch deren Kauf zum Berechtigten wird, wie dies durch das Wort „*Erwerber*“ vermittelt werden könnte, sondern auch durch die bloße Lizenzierung.[195] Der Deutsche Gesetzgeber wollte lediglich hervorheben, dass vom Anwendungsbereich des § 69 d UrhG sowohl der Erwerb als auch die Lizenzierung einer

[191] Str. im Ergebnis wie hier *Sahin / Haines*, CR 2005, 241, 244 f; *Baus*, MMR 2002, 14, 16; *Zahrnt*, NJW 1996, 1798, 1799; *Berger*, AcP 2001, 411, 446 ff; *Köhler / Fritzsche* in: Lehmann, Herstellung und Überlassung von Software im bürgerlichen Recht, Kap. XIII Rn. 47 ff, 61; *Vinck* in: Nordemann / Fromm, § 69 d Rn. 2; a.A. *Marly*, Softwareüberlassungsverträge, Rn. 484; *Koch*, Computer-Vertragsrecht Rn. 1952; *Moritz*, MMR 2001, 94, 95; *Pres*, Gestaltungsformen urheberrechtlicher Softwarelizenzverträge, S. 120 ff, 127.
[192] ABlEG 1989 Nr. C 91, S. 4 ff, 12.
[193] Amtl. Begründung, BT-Drucks 12/4022, S. 12.
[194] Amtl. Begründung, BT-Drucks 12/4022, S. 12.
[195] Amtl. Begründung, BT-Drucks 12/4022, S. 12.

Programmkopie erfasst wird. Eine Eingrenzung des Anwendungsbereichs wurde dagegen mit der Änderung des Wortlauts nicht bezweckt.[196]

Im Ergebnis ist somit festzuhalten, dass § 69 d Abs. 1 UrhG dem Schutz des freien Wirtschafts- und Warenverkehrs dient. In diesem Zusammenhang ergänzt § 69 d UrhG den gesetzlich bindenden Erschöpfungsgrundsatz, weshalb für die Berechtigung im Sinne des § 69 d Abs. 1 UrhG allein der Programmerwerb ausreichend ist.

Im Hinblick auf den Erwerb eines Open Source Programms bedeutet dies, dass der Nutzer von Open Source Software, für deren bloße Nutzung keiner weiteren Nutzungsrechtseinräumung durch die Urheber bedarf. [197] § 69 d UrhG ermächtigt die Nutzer der Open Source Software bereits kraft Gesetzes zu deren bestimmungsgemäßen Nutzung, so dass es insoweit keines, über den Erwerb der Programmkopie hinausgehenden, Vertragsschlusses zwischen Nutzern und Urhebern bedarf, um neben der Programmkopie auch die zur bestimmungsgemäßen Nutzung erforderlichen Nutzungsrechte eingeräumt zu bekommen.

Zwischenergebnis:

Insofern sind im Open Source Modell zwei Verträge zu unterscheiden. Auf der einen Seite steht der Vertrag, mit dem der Nutzer die Programmkopie erwirbt und auf der anderen Seite die GPL, mit welcher der Nutzer die Möglichkeit erhält, über die bestimmungsgemäße Nutzung hinaus, zu Entwicklung und Vertrieb der Open Source Software berechtigt zu werden.

Hält man sich diese Vertragskonstellation vor Augen, wird ersichtlich, dass die Nutzung von Open Source Software in zahlreichen Fällen ohne die Vereinbarung der GPL erfolgen wird.[198] Der Abschluss der GPL ist vielmehr nur dann Voraussetzung, wenn der Nutzer, über die zustimmungsfrei zulässige bestimmungsgemäße Nutzung des Programms hinaus, Handlungen

[196] Amtl. Begründung, BT-Drucks 12/4022, S. 12; *Grützmacher* in: Wandtke / Bullinger, § 69 d Rn. 24; *Sahin / Haines*, CR 2005, 241, 245.
[197] Vgl. *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 471. und wohl auch *Jaeger / Metzger*, Open Source Software, Rn. 174.
[198] Dies trifft zumindest dann zu, wenn nicht bereits der Programmbezug an die Annahme der GPL gekoppelt ist. In der Praxis lassen sich teilweise Programme finden, welche die GPL auf diese Weise einbeziehen. Die GPL setzt eine solche Form der „zwingenden" Einbeziehung hingegen nicht voraus. Für interaktive Programme soll lediglich ein Hinweis vorgesehen werden; vgl. Anhang zur GPL: *"If the program is interactive, make it output a short notice…"*

im Sinne von § 69 c UrhG vornimmt, für die er auf eine Zustimmung des Rechtsinhabers angewiesen ist.[199]

II. Vertragstypologische Einordnung des Erwerbs der Programmkopie

Der Erwerb der Programmkopie kann auf vielfältige Weisen erfolgen. Neben dem Download der Software von der Projektseite findet häufig ein Download von einem sog. Mirror,[200] oder auch direkt vom Distributor statt.[201] Daneben gibt es im Internet zahlreiche Webseiten, die sich mit dem Testen und Bewerten von Open Source Programmen befassen und zudem Downloadmöglichkeiten auf eigenen Servern bereithalten.[202]

Im Hinblick auf die vertragstypologische Einordnung des Erwerbs der Programmkopie lässt sich zunächst danach differenzieren, ob die Programmkopie entgeltlich oder unentgeltlich überlassen wurde.

1. Entgeltlicher Erwerb der Programmkopie

Sofern Open Source Software entgeltlich erworben wird, liegt diesem Erwerb häufig ein Vertrag mit einem Distributor zugrunde, bei dem neben der auf einem Datenträger verkörperten Programmkopie zumeist auch Handbücher und Supportleistungen Vertragsgegenstand sind.

---

[199] *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 690, geht ähnlich davon aus, dass der Abschluss der Open Source Lizenz erst in dem Moment relevant wird, in dem der Erwerber der Software diese nicht allein verwenden, sondern auch in die Vertriebs- und Entwicklungsposition einrücken möchte; von einem konkludenten Vertragsschluss zum Zeitpunkt der Vornahme von zustimmungsbedürftigen Handlungen gehen auch *Jaeger / Metzger*, Open Source Software, Rn. 177, aus; zu den einzelnen zustimmungsbedürftigen Handlungen im Sinne von § 69 c UrhG s.u. Kapitel 5 ab S. 98.

[200] Als einen Mirror bezeichnet man Server, die ein genaues Abbild eines anderen Datenbestands erstellen. Im Internet werden Mirror vor allem dann eingesetzt, wenn die zentralen Server großer Projekte nicht dazu in der Lage sind, ihre Dienste allen Interessierten zur Verfügung zu stellen. Aus diesem Grund werden die Daten auf anderen Servern gespiegelt, um die mit den Nutzerzugriffen einhergehende „Belastungen" der einzelnen Downloadserver verteilen und eine Verfügbarkeit der Dateien damit sicherstellen zu können.

[201] So sind Distributionen wie Debian (http://www.debian.org), Gentoo (http://www.gentoo.org), Ubuntu (http://www.ubuntu.com) aber auch kostenlose Varianten der kommerziellen Distributionen von Novell/Suse (http://de.opensuse.org) und Red Hat (http://fedora.redhat.com) im Internet verfügbar.

[202] Hierbei seien nur Downloadseiten wie http://www.zdnet.com, http://www.download.com oder auch das relativ neue Angebot von T-Online (http://www.softwareload.de) genannt.

Neben dem Vertrieb von Distributionen kommt ein entgeltlicher Vertrieb von Open Source Programmen aber auch in den Fällen vor, in denen die Software zusammen mit Hardware verkauft wird.[203]

Bei diesen Erwerbsvorgängen stellt sich regelmäßig die Frage, ob die verschiedenen Vertragsleistungen, die neben dem reinen Programmerwerb erbracht werden, als unabhängige Vertragsgegenstände angesehen werden können und somit vertragstypologisch getrennt einzuordnen sind. Dies ist im Open Source Bereich insofern von besonderer Bedeutung, als es sowohl haftungs- als auch gewährleistungsrechtlich erhebliche Unterschiede macht, ob sich die Geschäfte in entgeltliche und unentgeltliche trennen lassen, oder einheitlich als (gemischter) entgeltlicher Vertrag zu behandeln sind.[204]

Für die Beantwortung dieser Frage kommt es maßgeblich darauf an, ob die Trennung des Rechtsgeschäfts in entgeltliche und unentgeltliche Teile vom Parteiwillen umfasst wurde.[205] Soweit eine Auslegung der Willenserklärungen gem. §§ 133, 157 BGB ergibt, dass die Parteien von einem einheitlichen Vertragsgegenstand ausgegangen sind, kommt eine Aufspaltung der einzelnen Leistungspositionen nicht in Betracht. Das der Programmüberlassung zugrunde liegende Rechtsgeschäft ist dann einheitlich zu betrachten und als solches auch vertragstypologisch einzuordnen.

Ergibt sich hingegen aus dem Parteiwillen, dass von mehreren Vertragsgegenständen ausgegangen wurde, so ist eine Differenzierung danach, welche Vertragsgegenstände unentgeltlich und welche entgeltlich verschafft werden sollten, möglich.

[203] In der Praxis lassen sich solche Bundels oder auch Embedded Systeme zahlreich finden. Ein gemeinsamer Vertrieb von Hardware und Linux-Distributionen findet sich beispielsweise im Zusammenhang mit der Distribution Suse Linux Enterprise Server, http://www.heise.de/newsticker/meldung/79726. Zu den weiteren Einsatzmöglichkeiten von Linux im Bereich der Embedded Systeme siehe bereits oben S. 15.

[204] Zu den Fragen der Wirksamkeit des in Ziff. 11 und 12 GPL enthaltenen Haftungs- und Gewährleistungsausschluss im Hinblick auf die Regelungen der §§ 305 ff BGB vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 219 ff; *Schiffner*, Open Source Software, S. 241 ff; *Marly*, Softwareüberlassungsverträge, Rn. 440 ff; *Omsels* in: FS Hertin, S. 147 ff; *Spindler*, Rechtsfragen bei Open Source, D. Rn. 14 ff.

[205] Im Hinblick auf den Erwerb von Distributionen, vgl. *Jaeger / Metzger*, Open Source Software, Rn. 256; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 804 f.

a.Erwerb einer Distribution

Legt man den Parteiwillen zugrunde, wird im Rahmen des Vertriebs von Distributionen regelmäßig von einem einheitlichen gemischten Vertrag auszugehen sein.[206] Der Kunde erwirbt die Distribution zu einem einheitlichen Preis. In diesem Preis sind die auf Datenträgern verkörperte Programmkopie, die Handbücher aber auch die Supportleistungen und mithin Kauf- und Dienstvertragliche Leistungen enthalten. Eine Trennung in verschiedene Verträge kommt hierbei im Zweifel nicht in Betracht, da sich die Vertragsgegenstände – wie sich nicht zuletzt aus dem Gesamtpreis ergibt – nicht trennen lassen, sondern als ein Gesamtpaket erbracht werden. Der Kunde wird trotz Kenntnis von der Open Source Eigenschaft der erworbenen Software nicht erkennen können, welche Leistungen unentgeltlich erworben werden.[207] In schuldrechtlicher Sicht liegt dem Erwerb einer Distribution somit regelmäßig ein einheitlicher und nicht weiter aufzuspaltender Vertrag zugrunde.[208]

Der Kunde erwirbt die Distribution, einschließlich der darin enthaltenen Programme, in einem einheitlichen Vertrag, der auf den einmaligen entgeltlichen Leistungsaustausch gerichtet ist und somit vertragstypologisch einem Kaufvertrag im Sinne von § 433 BGB entspricht.[209] Jedenfalls für die vertragstypologische Einordnung als Kaufvertrag ist es seit der Einführung des § 453 BGB und der dadurch erfolgten Gleichstellung von Rechts- und Sachkauf zudem unerheblich,[210] ob man auf Software als Sache qualifiziert, da die kaufrechtlichen Vorschriften zumindest entsprechend Anwendung finden.[211] Ebenso vermag die Form, in

---

[206] Ebenso *Jaeger / Metzger*, Open-source-Software, Rn. 256; sowie *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 357 f.

[207] *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 357.

[208] Vgl. *Jaeger / Metzger*, Open-source-Software, Rn. 256; sowie *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 357 f; aufgrund der Bündelung der verschiedenen Leistungen geht auch *Spindler*, Rechtsfragen bei Open Source, D. Rn. 33, von einem einheitlichen Vertrag aus.

[209] So auch *Jaeger / Metzger*, Open-source-Software, Rn. 256; *Grützmacher*, ITRB 2002, 84, 86; *Schiffner*, Open Source Software, S. 222 f; für den gemeinsamen (körperlichen) Vertrieb von Open Source Dienstleistungen *Spindler*, Rechtsfragen bei Open Source, D. Rn. 33; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 353 ff, 358.

[210] Im Hinblick auf die Anwendung der kaufrechtlichen Vorschriften vgl. *Hoeren* in: Dauner-Lieb / Konzen, u.a., Das neue Schuldrecht in der Praxis, S. 517; *Schlechtriem*, Schuldrecht Besonderer Teil, § 2 Rn. 28; *Spindler*, Rechtsfragen bei Open Source, D. Rn. 6; allerdings ist zu beachten, dass sich die Frage nach der Qualifikation von Software als Sache dadurch nicht erübrigt hat; zu den verbleibenden Problemfragen, vgl. nur *Marly*, Softwareüberlassungsverträge, Rn. 116.

[211] Die Sacheigenschaft von Software ist nach wie vor streitig (für die Qualifizierung als Sache, vgl. u.a. *Hoeren*, Softwareüberlassung als Sachkauf, Rn. 70 ff; *König*, Das Computerprogramm im Recht, Rn. 258 ff; *Marly*, Softwareüberlassungsverträge, Rn. 96 ff; hingegen für die Qualifizierung als immaterielles Gut, vgl. u.a. *Köhler / Fritzsche* in: Lehmann, Herstellung und Überlassung von Software im bürgerlichen Recht, Kap. XIII Rn. 7; *Junker / Benecke*, Computerrecht, Rn. 156) und wurde zuletzt aufgrund der Schuldrechtsmodernisierung erneut in Frage gestellt (vgl. u.a. *Stichtenoth*, K&R 2003, 105, 106 ff); zum Diskussionsstand insgesamt vgl. nur *Marly*, Softwareüberlassungsverträge, Rn. 96 ff m.w.N., sowie zu den Änderungen durch die

welcher der Nutzer die Distribution erhält – also ob sie auf einem Datenträger oder als Bits und Bytes übertragen wurde – nichts an der vertragstypologischen Einordnung zu ändern.[212]

Soweit Open Source Software entgeltlich zur dauerhaften Nutzung von einem Distributor erworben wurde, ist der diesem Rechtsgeschäft zugrunde liegende Vertrag somit einheitlich als Kaufvertrag einzuordnen. Etwas anderes kann sich im Einzelfall nur dann ergeben, wenn der Distributor eine eindeutige Aufspaltung in entgeltliche und unentgeltliche Leistungen vornimmt, aufgrund der ein objektiver Erklärungsempfänger von mehreren Verträgen ausgehen würde.

b.Erwerb von Open Source Software zusammen mit Hardware

Open Source Software wird daneben auch in solchen Fallkonstellationen Gegenstand eines entgeltlichen Geschäfts, in denen sie zusammen mit Hardware vertrieben wird. Dies kommt in der Praxis häufig dann vor, wenn Computer verkauft werden, auf denen GNU/Linux bzw. andere Open Source Software bereits vorinstalliert ist.[213] Ebenfalls sehr verbreitet ist der Vertrieb von Open Source Software auf sog. Embedded Systemen.[214]

Alle dieser Fallgestaltungen zeichnen sich dadurch aus, dass die Hardware zumeist zusammen mit der Software zu einem einheitlichen Preis verkauft wird und mithin auch von einem einheitlichen Kaufvertrag auszugehen ist.[215] Häufig wird sich der Kaufvertrag dabei nicht einmal ausdrücklich auf die Software beziehen, da es dem Käufer typischerweise nur auf das Gerät an sich, nicht aber auf die darin enthaltenen Open Source Programme ankommt. Daher kann dem Parteiwillen regelmäßig keine Aufspaltung der Vertragsgegenstände entnommen werden, so dass auch hier wieder von einem einheitlichen Kaufvertrag im Sinne von § 433 BGB auszugehen ist.

Schuldrechtsmodernisierung Rn. 115 ff; ausführlich zur Sacheigenschaft nach altem Recht *Bydlinski*, AcP 1998, 287, 288 ff.

[212] Der Vertriebsform wurde bereits vor der Schuldrechtsmodernisierung zu Recht keine Bedeutung beigemessen, vgl. BGH NJW 1990, 320, 321; OLG Stuttgart NJW 1989, 2635, 2636; *Hoeren*, Softwareüberlassung als Sachkauf, Rn. 76; *Marly*, Softwareüberlassungsverträge, Rn. 109 ff; *Pres*, Gestaltungsformen urheberrechtlicher Softwarelizenzverträge, S. 170.

[213] Ein gemeinsamer Vertrieb von Hardware und Linux-Distributionen findet sich beispielsweise im Zusammenhang mit der Distribution Suse Linux Enterprise Server, http://www.heise.de/newsticker/meldung/79726.

[214] Zu den Einsatzgebieten von Linux im Bereich der Embedded Systeme vgl. S. 15 f.

[215] Im Hinblick auf den Vertrieb von Embedded Systemen, vgl. *Jaeger / Metzger*, Open-source-Software, Rn. 269; differenzierend anhand der Ausgestaltung des Angebots *Spindler*, Rechtsfragen bei Open Source, D. Rn. 39.

2.Unentgeltlicher Erwerb der Programmkopie

Im Gegensatz zum entgeltlichen Vertrieb von Open Source Software, kommt bei einem unentgeltlichen Erwerb von Programmen vor allem die vertragstypologische Einordnung als Handschenkung im Sinne von § 516 BGB aber auch die als Auftragsverhältnis im Sinne von §§ 662 ff BGB in Betracht. Eine Einordnung als Leihvertrag im Sinne von §§ 598 ff BGB, die aufgrund der Unentgeltlichkeit ebenfalls in Erwägung gezogen werden könnte, scheidet hingegen bereits wegen der fehlenden Rückgabeverpflichtung aus.[216]

Soweit in der Literatur teilweise eine Einordnung als Auftragsverhältnis vertreten wird, wird diese damit begründet, dass eine Vergleichbarkeit mit der Überlassung von Public Domain Software und Freeware bestünde. Ebenso wie dort stehe bei der unentgeltlichen Überlassung von Open Source Software nicht die Verschaffung der Programmkopie im Mittelpunkt, sondern die Kopier-, Bereitstellungs- und Versendetätigkeit.[217]

Unabhängig davon, ob diese Annahmen für die Überlassung von Public Domain Software und Freeware zutreffen, lässt sich jedenfalls der Erwerb von Open Source Software nicht mit diesen Vertriebsformen vergleichen. Im Gegensatz zu den Fällen der Public Domain Software oder auch Freeware, will der Vertriebsberechtigte bei der Überlassung von Open Source Software typischerweise nicht im fremden Interesse tätig werden,[218] wie es für das Auftragsverhältnis kennzeichnend wäre.[219] Er strebt durch sein Angebot regelmäßig nicht die Bereitstellung der Software, sondern deren endgültige Übertragung an, so dass er nicht im Interesse eines Dritten tätig wird.[220]

Der Schwerpunkt des Vertrags ist damit auf die unentgeltliche Übertragung der Open Source Software gerichtet, weshalb sie aus vertragstypologischer Sicht als Handschenkung im Sinne von § 516 BGB einzuordnen sein könnte, was wiederum unabhängig davon möglich ist, ob man Software als Sache qualifiziert, da die Rechtsprechung die Vorschriften über Sache zumindest analog anwendet[221] und auch in der Literatur darüber Einigkeit besteht, dass Open

---

[216] So auch *Koch*, CR 2000, 333, 335; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 360; sowie *Weber* in: Festschrift Honsell, S. 52 f, im Hinblick auf das Schweizer Recht.
[217] Vgl. die Ausführungen zur unentgeltlichen Bereitstellung von Open Source Software auf einem Server zum Download bei *Marly*, Softwareüberlassungsverträge, Rn. 427.
[218] *Hoeren* in: Festschrift Kollhosser, S. 234.
[219] Zur Interessenlage beim Auftragsverhältnis vgl. nur *Sprau* in: Palandt, § 662 Rn. 7 m.w.N.
[220] Vgl. *Metzger / Jaeger*, GRUR Int. 1999, 839, 847; ebenso: *Spindler*, Rechtsfragen bei Open Source, D. Rn. 4; *Hoeren* in: Festschrift Kollhosser, 234; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 831; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 361.
[221] Vgl. nur BGH NJW 1993, 2436, 2437 f; BGH NJW 1990, 320, 321; BGH NJW 1988, 406, 407 f.

Source Programme jedenfalls grundsätzlich dazu geeignet sind, Gegenstand einer Zuwendung im Sinne von § 516 BGB zu sein.[222]

Voraussetzung ist dafür, dass in der Programmüberlassung eine unentgeltliche Zuwendung aus dem Vermögen des Schenkers liegt, die zu einer Bereicherung des Beschenkten führt.

a.Unentgeltlichkeit der Zuwendung

An der Unentgeltlichkeit der Zuwendung könnten im Hinblick auf die Überlassung der Programmkopie allenfalls dann Zweifel bestehen, wenn diese an Bedingungen geknüpft würde, die letztlich als Gegenleistung einzustufen sind.[223]

Im Zusammenhang mit der Überlassung der Programmkopie könnten solche Bedingungen einzig aus der GPL folgen, die jedoch aufgrund der gebotenen Aufspaltung zwischen dem Erwerb der Programmkopie und dem der Nutzungsrechte, bei der vertragstypologischen Einordnung nicht zu berücksichtigen ist.[224]

Weitere Bedingungen, an welche die Übergabe der Programmkopie geknüpft wird, sind nicht ersichtlich, weshalb sie die Voraussetzungen einer unentgeltlichen Zuwendung im Sinne von § 516 BGB erfüllt.[225]

---

[222] Insofern sind sich sämtliche Ansichten in der Literatur – trotz durchaus unterschiedlicher Grundannahmen zur Sachqualität von Software – zumindest im Ausgangspunkt darüber einig, das Open Source Programme Gegenstand einer Zuwendung im Sinne von § 516 BGB sein können, vgl. u.a. *Spindler*, Rechtsfragen bei Open Source, D. Rn. 6; *Jaeger / Metzger*, Open Source Software, Rn. 206; *Schiffner*, Open Source Software, S. 228; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 841 ff; *Hoeren* in: Festschrift Kollhosser, S. 235.

[223] Vgl. nur *Putzo* in: Palandt, § 516 Rn. 8.

[224] Wie hier *Schiffner*, Open Source Software, S. 228; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 828 Rn. 907; Bedenken an der Unentgeltlichkeit der Zuwendung werden von denjenigen geäußert, welche die – nach der hier vertretenen Auffassung – gebotene Aufspaltung der Rechtsgeschäfte in den Erwerb der Programmkopie und den der weitergehenden Nutzungsrechte nicht vornehmen, vgl. *Koch*, CR 2000, 333, 335.

[225] Vgl. *Metzger / Jaeger*, GRUR Int. 1999, 839, 847; *Jaeger / Metzger*, Open Source Software, Rn. 209; ebenso *Deike*, CR 2003, 9, 14 f; *Schiffner*, Open Source Software, S. 228; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 828; a.A. *Koch*, CR 2000, 333, 335, der wohl bereits die Unentgeltlichkeit der Zuwendung aufgrund der in den Open Source Lizenzen enthaltenen Bedingungen ablehnt, im Übrigen zudem darauf verweist, dass in den Lizenzbedingungen eine Schenkungsauflage im Sinne von § 518 BGB liegen würde, die mangels notarieller Beurkundung regelmäßig nichtig ist; ähnlich auch *Hoeren* in: Festschrift Kollhosser, 237 f.

b.Bereicherung

Die für eine Handschenkung gem. § 516 BGB weiterhin erforderliche Bereicherung des Beschenkten tritt im Rahmen der Überlassung der Programmkopie dadurch ein, dass der Nutzer diese erhält und sie gem. § 69 d UrhG in bestimmungsgemäßer Weise nutzen darf.[226]

c.Zuwendung aus dem Vermögen des Schenkers

Problematisch an der vertragstypologischen Einordnung der Programmüberlassung als Handschenkung könnte jedoch das Erfordernis der Zuwendung aus dem Vermögen des Schenkers sein.

Bedenken am Bestehen der hierfür erforderlichen Entreicherung ergeben sich dabei aus dem Umstand, dass Software beliebig vervielfältigt werden kann. Soweit der Erwerber aber – wie etwa beim Download – lediglich eine Kopie des Originals herstellt, kommt es hierdurch zu keinem Verlust des Originals.[227] Insofern scheinen die Zweifel an dem Fehlen der von § 516 BGB vorgesehenen Entreicherung des Schenkers gerechtfertigt zu sein. Fraglich ist allerdings, ob hieraus auch zwangsläufig der Schluss zu ziehen ist, dass Software generell nicht Gegenstand einer Schenkung sein kann.

Die wohl herrschende Ansicht ist sich – mit unterschiedlichen Begründungen – im Ergebnis darüber einig, dass allein die beliebige Vervielfältigungsmöglichkeit von Software nicht per se gegen die Anwendung der schenkungsrechtlichen Vorschriften spricht.[228]

aa)Entreicherung aufgrund der Herstellungskosten

Soweit in der Literatur am Erfordernis der Entreicherung festgehalten wird, wird die Anwendbarkeit des Schenkungsrechts teilweise damit begründet, dass mit jeder Herstellung

[226] *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 857.
[227] Aufgrund der hierdurch gegebenen ungehinderten weiteren Nutzungsmöglichkeiten wird die Einordnung als Handschenkung von *Sester*, CR 2000, 797, 799 f, abgelehnt; *Koch*, CR 2000, 333, 335, stellt vor allem darauf ab, dass lediglich Kopien nicht aber das Original des Programms weitergegeben werden.
[228] Vgl. *Schiffner*, Open Source Software, 227 f; *Jaeger / Metzger*, Open Source Software, Rn. 207; *Spindler*, Rechtsfragen bei Open Source, D. Rn. 7; a.A. *Hoeren* in: Festschrift Kollhosser, 235 f; *Sester*, CR 2000, 797, 799 f; *Koch*, CR 2000, 333, 334.

der Programmkopie zugleich auch Kopier-, zumindest aber Energiekosten verbunden wären,[229] in denen die erforderliche Entreicherung liegen würde.

Hiergegen spricht indes, dass solche Kostenfaktoren letztlich bei jeder Übertragungs-, Dienst- oder sonstigen Leistung auszumachen sind, weshalb jede unentgeltliche Tätigkeit vertragstypologisch als Schenkung anzusehen wäre. Die Unterscheidungskraft, die dem Kriterium der Entreicherung zukommen soll, wäre bei Berücksichtigung von Kostenteilen wie dem Energieaufwand oder auch der verwendeten Arbeitszeit, nahezu völlig aufgehoben.[230] Für die Entreicherung kann daher nicht auf die, mit der Erstellung der Programmkopie verbundenen, Herstellungskosten abgestellt werden.

bb)Entreicherung im Hinblick auf die Nutzungsrechte

Weiterhin wird in der Literatur die Ansicht vertreten, dass es für die Beurteilung der Entreicherung nicht auf die Programmkopie an sich, sondern auf die hiermit verbundenen Nutzungsrechte ankommen würde.[231] Eben diese Rechte seien der eigentliche Vermögensgegenstand, der im Rahmen der Handschenkung zugewendet werde. Der Erhalt der Sachsubstanz soll für den Erwerber hingegen nachrangig sein, weshalb auf der Seite des Schenkers ebenfalls nicht auf die physische Entreicherung abgestellt werden könne.[232] Für die erforderliche Entreicherung soll es vielmehr ausreichen, dass der Schenker nach dem Vollzug der Schenkung in seinem von § 69 c UrhG vermittelten Ausschließlichkeitsrecht eingeschränkt werde.[233] Aus diesem Grund wäre auch bei der Überlassung von Open Source Programmen regelmäßig die erforderliche Entreicherung gegeben.[234]

Obwohl dieser Ansatz im Grundsatz zutrifft, vermag gerade die Letzte Schlussfolgerung im Open Source Bereich nicht zu überzeugen, da die Programmkopie hier eben oftmals nicht direkt vom Urheber, sondern von einem Dritten überlassen wird, bei dem nach vorgenannten Gesichtspunkten mangels entsprechender Rechtsinhaberschaft aber gerade keine Entreicherung festzustellen ist. Geht man demnach davon aus, dass die Programmkopie nicht direkt vom Urheber überlassen wurde, muss man richtigerweise zu dem Ergebnis kommen,

[229] So *Schiffner*, Open Source Software, S. 228.
[230] Als „wenig hilfreich“ bezeichnet daher *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 839 Fn. 919, den Ansatz von Schiffner.
[231] *Spindler*, Rechtsfragen bei Open Source, D. Rn. 7; *Bartosz*, MR 2005, 40, 41.
[232] *Spindler*, Rechtsfragen bei Open Source, D. Rn. 7; *Bartosz*, MR 2005, 40, 41.
[233] *Spindler*, Rechtsfragen bei Open Source, D. Rn. 7.
[234] *Spindler*, Rechtsfragen bei Open Source, D. Rn. 7.

dass eine Entreicherung beim Schenker nicht gegeben ist, weshalb zumindest eine direkte Anwendung des § 516 BGB nicht in Betracht kommt.

cc)Analoge Anwendung von § 516 BGB

Allerdings könnte eine analoge Anwendung der schenkungsrechtlichen Vorschriften gem. § 516 BGB geboten sein.

Neben der Ähnlichkeit, welche die unentgeltliche Überlassung von Programmkopien zum Wesen der Handschenkung aufweist, lässt sich für die Begründung einer analogen Anwendung von § 516 BGB auch ein Vergleich mit der unentgeltlichen Überlassung von proprietärer Software heranführen. Bei dieser würde nämlich im Zweifel auch bei einem Download der Software von einer Schenkung ausgegangen werden, sofern dem Schenker lediglich mengenmäßig beschränkte Vertriebsrechte zustehen.[235] Die für eine Anwendung von § 516 BGB erforderliche Entreicherung läge in diesen Fällen darin, dass sich die künftigen Verfügungsmöglichkeiten des Schenkers mengenmäßig um eben diese Programmkopie reduziert haben.[236]

Aus vertragstypologischer Sicht läge der maßgebliche Unterschied zum Open Source Modell somit allein in dem Umfang der rechtlichen Verfügungsmöglichkeiten des Schenkers, nicht jedoch in dem eigentlichen Schenkungsakt, der sich im Hinblick auf die Programmkopie ebenfalls durch deren beliebige Vervielfältigungsmöglichkeit auszeichnet. Der Umfang der rechtlichen Verfügungsbefugnisse ist jedoch als Kriterium für die vertragstypologische Einordnung ungeeignet,[237] weshalb der äußerlich identische Schenkungsakt im Open Source Bereich ebenfalls als Handschenkung im Sinne von § 516 BGB zu bewerten ist.

Dies wird auch durch die Entstehungsgeschichte der Vorschrift bestätigt. Aus den Motiven des Gesetzgebers geht hervor, dass das Kriterium der Entreicherung letztlich eine eindeutige Abgrenzung zu den Fällen des § 517 BGB ermöglichen sollte,[238] also für solche

[235] *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 851 ff, weist insofern zu Recht darauf hin, dass das Problem der Zuwendung einzig darauf zurückzuführen ist, dass der Verfügende im Open Source Bereich ausnahmsweise nicht mengenmäßig in seinen Verfügungsmöglichkeiten beschränkt ist.
[236] *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 851.
[237] Ähnlich *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 853, der darauf abstellt, dass für die Frage nach der für die Zuwendung erforderlichen Vermögensverschiebung stärker auf die Seite des Leistungsempfängers abzustellen ist und es auf Seiten des Zuwendenden ausreichen muss, dass über ein in Geld bewertbares Gut aus dem Vermögen des Zuwendenden verfügt wurde.
[238] Motive zum BGB Band 2 S. 286 f, 290.

Fallgestaltungen gedacht war, in denen zum Vorteil eines anderen auf einen künftigen Vermögenserwerb verzichtet wird. Um das praktische Verständnis des Schenkungsbegriffs zu verdeutlichen, hat der damalige Gesetzgeber künftige Vermögensänderungen unberücksichtigt lassen wollen, indem er den Schenkungsbegriff auf eine Verminderung des gegenwärtigen Vermögen begrenzt hat.[239] Diese Begrenzung widerspricht aber nicht der vertragstypologischen Einordnung der Programmüberlassung als Handschenkung, da es bei dieser nicht um die Einbeziehung künftiger Vermögensänderungen geht. Vielmehr stellt sich einzig die Frage danach, wie in den Fällen zu verfahren ist, in denen zwar strukturell das gegenwärtige Vermögen betroffen ist, eine Vermögensminderung jedoch aufgrund der beliebigen Reproduzierbarkeit des Vertragsgegenstandes nicht eintritt.

Um für die vertragstypologische Einordnung der Programmüberlassung nicht auf das ungeeignete Kriterium der rechtlichen Verfügungsmöglichkeiten des Schenkers abstellen zu müssen, kann es vorliegend nicht auf dessen tatsächliche Entreicherung ankommen. Es reicht vielmehr aus, dass der Beschenkte um einen Vermögensgegenstand bereichert wird, der aus dem Bereich der Verfügungsmöglichkeiten des Schenkers stammt[240] und somit zu dessen gegenwärtigen Vermögen zu rechnen ist.

Ergebnis:

Im Ergebnis lässt sich somit festhalten, dass dem entgeltlichen Erweb der Programmkopie regelmäßig ein Kauf- und der unentgeltlichen Überlassung ein Schenkungsvertrag zugrunde liegt, unabhängig davon, in welcher Form die Programmkopie überlassen wurde.

[239] Motive zum BGB Band 2 S. 286 f, 290.
[240] So auch *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 853 f.

III. Vertragstypologische Einordnung des Erwerbs der weitergehenden Nutzungsrechte

Neben dem Vertrag, mit dem der Nutzer die Programmkopie erwirbt, wird er oftmals auch die GPL annehmen. Dies ist zunächst immer dann der Fall, wenn der Nutzer bereits vor dem Programmbezug zur Annahme der GPL verpflichtet wird.[241] Eine konkludente Annahme der GPL liegt aber auch dann vor, wenn der Nutzer Entwicklungs- oder Vertriebshandlungen vornimmt, für die er gem. § 69 c UrhG auf die Zustimmung des Urhebers angewiesen ist.[242]

In den Fällen schließt der Nutzer, neben dem Vertrag der dem Erwerb der Programmkopie zugrunde liegt, auch die GPL ab und erwirbt hierdurch das Recht die Programmkopie zu verändern und zu vertreiben.

Zur vertragstypologischen Einordnung dieses Vertrags werden in der Literatur zum Teil sehr verschiedene Ansätze vertreten. Diese reichen von einer Einordnung als Gesellschaftsvertrag[243] bis hin zu der als Schenkungsvertrag.[244] Daneben wird die GPL teilweise auch als Vertrag sui generis eingeordnet,[245] ohne dass diese Feststellung alleine jedoch im Hinblick auf die rechtlichen Konsequenzen weiterhelfen würde.

1. Leistungspflichten der GPL

Um eine vertragstypologische Einordnung der GPL in das System der bürgerlich-rechtlichen Vertragstypen zu ermöglichen, ist zunächst ein Blick auf die zugrunde liegenden

[241] Diese Form der Einbeziehung lässt sich teilweise in der Praxis finden, auch wenn sie durch die GPL nicht gefordert wird. Dem Anhang der GPL lässt sich vielmehr entnehmen, dass lediglich ein kurzen Vermerk auf die GPL vorgesehen ist, ohne dass deren Abschluss zur Voraussetzung für die Nutzung oder den Bezugs der Programmkopie gemacht würde. Vgl. Anhang der GPL: *"If the program is interactive, make it output a short notice (...) when it starts in an interactive mode…"*

[242] Einzelheiten hierzu unter Kapitel 5 S. 98 ff.

[243] So bei *Sester*, CR 2000, 797, S. 801 f, der jedoch ausdrücklich darauf hinweist, dass er zwar von einer Vergleichbarkeit zur BGB-Gesellschaft ausgehe, aber bewusst die Aussage vermeide, dass eine solche auch vorläge; beschränkt auf den „Kern der Entwickler eines Open Source Programms" wohl auch *Schiffner*, Open Source Software, S. 234 ff.

[244] So bei *Spindler*, Rechtsfragen bei Open Source, D. Rn. 4 ff, ohne dass jedoch eine deutliche Trennung zwischen dem Erwerb der Programmkopie und dem der Nutzungsrechte vorgenommen würde; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 379 ff, der zwar die für die Nutzungsrechtseinräumung zwischen dem Vertrag zur Einräumung der Basisnutzung und dem zur Einräumung der qualifizierten Nutzung unterscheidet, allerdings hinsichtlich beider Verträge zu einer Anwendung der schenkungsrechtlichen Vorschriften kommt; ähnlich auch *Jaeger / Metzger*, Open-source-Software, Rn. 217, die von einem Lizenzvertrag mit schenkungsrechtlichen Elementen ausgehen.

[245] Vgl. *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 948 ff; *Koch*, CR 2000, 333, 335; ähnlich *Schiffner*, Open Source Software, S. 237 f, der diese Einordnung als Alternative zum Gesellschaftsvertrag ansieht.

Leistungspflichten zu werfen. Hierbei ist zu klären, ob die in der GPL enthaltenen Verpflichtungen auf einen einmaligen Leistungsaustausch oder auf dauerhafte Leistungsverpflichtungen ausgerichtet sind.

Bei konstitutiven Nutzungsrechtseinräumungen, wie sie im Rahmen der GPL gegeben sind, wird in der Literatur teilweise generell von dauerhaften Leistungspflichten und somit von Dauerschuldverhältnissen ausgegangen.[246] Zur Begründung wird dabei darauf verwiesen, dass eine dingliche Vollrechtsübertragung im Immaterialgüterrechtssystem nicht vorgesehen sei, so dass den Lizenzgeber fortlaufend die Pflicht zur Duldung der Nutzung treffe. Insofern sei ein einmaliger Leistungsaustausch von vorneherein ausgeschlossen, so dass aus vertragstypologischer Sicht ausschließlich Dauerschludverhältnisse vorliegen könnten.

Demgegenüber geht die wohl herrschende Ansicht zu Recht davon aus, dass auch bei konstitutiven Nutzungsrechtseinräumungen reine Austauschverhältnisse vorliegen können.[247] Selbst wenn die Übertragung des immaterialgüterrechtlichen Ausschließlichkeitsrechts nicht möglich ist, können die im Lizenzvertrag enthaltenen Verpflichtungen gleichwohl auf eine einmalige Rechtseinräumung gerichtet sein und sich hierin erschöpfen.[248] Darum kann nicht generell bei allen Lizenzverträgen von einem Dauerschuldverhältnis ausgegangen werden. Für eine Einordnung als Dauerschuldverhältnis ist vielmehr zu fordern, dass neben die Pflicht zur Rechtsverschaffung weitere Verpflichtungen der Vertragsparteien treten.[249] Liegt hingegen eine Vereinbarung vor, deren Hauptleistung ausschließlich auf eine einmalige und endgültige Rechtseinräumung gerichtet ist, und lassen sich andauernde und wiederkehrende

[246] *Lütje* in: Möhring / Nicolini, § 112 UrhG Rn. 13; im Hinblick auf den Softwarevertrag *Hilty*, MMR 2003, 3, 14 f; Übersicht m.w.N. bei *Kreuzer / Reber* in: Loewenheim, Handbuch des Urheberrechts § 95 Rn. 71 ff.

[247] Vgl. Insbesondere *Wallner*, ZIP 2004, 2073, 2074 ff; *Wallner*, Die Insolvenz des Urhebers, S. 156 ff; *Grützmacher* Anm. zu LG Mannheim CR 2004, 811, 815; *Paulus*, CR 2003, 237, 240 Fn. 9 für den Verkauf von Software; im Hinblick auf die Einräumung der Nutzungsrechte im Rahmen der GPL *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 381 f, mit Verweis auf die Ausführungen ab S. 135 ff.

[248] Für Softwarekauf *Grützmacher* Anm. zu LG Mannheim CR 2004, 811, 815; *Paulus*, CR 2003, 237, 240 Fn. 9.

[249] Solche weiteren Verpflichtungen können beispielsweise in einer vereinbarten Vertragslaufzeit, wiederkehrenden Zahlungsverpflichtungen oder auch Auswertungspflichten des Lizenznehmers liegen; weitere „Indizien“ die für eine Einordnung von Nutzungsverträgen als Dauerschuldverhältnisse sprechen vgl. *Kreuzer / Reber* in: *Loewenheim*, Handbuch des Urheberrechts, § 95 Rn. 82. Solche weiteren Verpflichtungen lagen letztlich auch bei den Lizenzverträgen vor, bei denen die Rechtsprechung eine Einordnung als Dauerschuldverhältnis vorgenommen hat, vgl. beim Musikverlagsvertrag BGH GRUR 1982, 41, 43 - Musikverleger III; BGH GRUR 1990, 443, 444 - Musikverleger IV, beim Stoffrechtevertrag OLG Schleswig ZUM 1995, 867, 873 – „Werner“-Serie; sowie beim Softwarelizenzvertrag LG Mannheim CR 2004, 811, 813, wobei es dem LG bereits genügte, dass noch Nebenleistungspflichten ausstanden.

Pflichten auch nicht aus den Nebenpflichten entnehmen,[250] so vermag die vertragstypologische Einordnung als Dauerschuldverhältnis nicht zu überzeugen.[251]

Dies wirft die Frage auf, wie die in der GPL enthaltenen Verpflichtungen zu beurteilen sind. Neben die Pflicht zur Rechtsverschaffung, deren zeitliche Wirkung eher auf eine einmalige Leistung gerichtet ist, sind dabei vor allem die den Nutzer treffenden Pflichten zur Einhaltung gewisser Vertriebsmodalitäten, sowie zur Offenlegung des Quellcodes zu berücksichtigen. Betrachtet man diese Vertriebsmodalitäten im Hinblick auf ihre zeitliche Geltung, so ist festzustellen, dass der Nutzer hierdurch nicht nur einmalig, sondern für die gesamte Nutzungszeit der Software gebunden werden soll, weshalb eine Einordnung als Dauerschuldverhältnis geboten ist.[252]

Berücksichtigt man weiterhin, dass die GPL die Einräumung umfassender Vertriebs- und Entwicklungsrechte zum Gegenstand hat, liegt die Einordnung als urheberrechtlicher Lizenzvertrag nahe, ohne dass damit jedoch bereits die Anwendung spezifischer gesetzlicher Regelungen vorgegeben wäre.[253] Aufgrund der Unentgeltlichkeit der Rechtseinräumung passen aber sowohl die Vorschriften über Pacht- als auch die über Mietverträge nicht.[254]

Auch eine Anwendung der gesetzlichen Regelungen in §§ 705 ff. BGB kommt letztlich nicht in Betracht.[255] Zwar könnte für einen gesellschaftsähnlichen Vertrag sprechen, dass der Open

[250] Dies hat dem LG Mannheim CR 2004, 811, 813 genügt, um von einer Anwendung von § 103 InsO ausgehen zu können.

[251] Str., wie hier *Wallner*, Die Insolvenz des Urhebers, S. 156 ff; für den Verkauf von Standardsoftware *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 139 ff; ebenso *Grützmacher* Anm. zu LG Mannheim CR 2004, 811, 815; *Paulus*, CR 2003, 237, 240 Fn. 9; wohl auch *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 942, der die Annahme der für Dauerschuldverhältnisse erforderlichen Abhängigkeit von Zeitmoment und Pflichtenumfang dort für sehr zweifelhaft hält, wo von den Vertragsparteien ausschließlich eine endgültige Aufgabe der wirtschaftlichen Position durch den Rechtsinhaber und eine endgültige Neuzuordnung des Vermögenswertes beabsichtigt wird; a.A. *Lütje* in: Möhring / Nicolini, § 112 UrhG Rn. 13.

[252] *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 943 ff; *Jaeger / Metzger*, Open Source Software, Rn. 217; a.A. *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 381, der zwar zunächst (S. 139 ff) zu Recht zu dem Ergebnis gelangt, dass beim Verkauf von Standardsoftware zumeist ein einmaliger Leistungsaustausch beabsichtigt sei, der eine vertragstypologische Einordnung als Dauerschuldverhältnis nicht zu rechtfertigen vermag, dann aber fälschlicher Wfälschlicherweiseis undifferenziert auch auf die GPL anwenden will und dabei übersieht, dass die Leistungspflichten Rahmen der GPL nicht denen beim Verkauf von Standardsoftware entsprechen.

[253] *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 948 f;

[254] Für urheberrechtliche Lizenzverträge wird in der Literatur oftmals auf diese gesetzlichen Regelungen zurückgegriffen. Vorliegend passen sie jedoch nicht, da die GPL gerade keine entgeltliche Gegenleistung vorsieht. Die Vertriebspflichten des Nutzers stehen in keinem Gegenseitigkeitsverhältnis zur Einräumung der Rechte, was nicht zuletzt daraus ersichtlich wird, dass der Nutzer GPL-lizenzierte Software ohne weiteres modifizieren und nutzen kann, ohne hierfür an Pflichten gebunden zu sein; a.A. *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 952.

[255] Für eine Nähe zur BGB-Gesellschaft spricht sich *Sester*, CR 2000, 797, 801 f, aus; ähnlich auch *Schiffner*, Open Source Software, S. 234 ff, im Hinblick auf den Kern der Entwickler, wobei nicht ganz ersichtlich wird, ob Schiffner die vertragstypologische Einordnung hierbei allein auf die Regelungen der GPL oder nicht viel eher auf vermeintliche (konkludente) Zusatzregelungen zwischen dem Kern der Entwickler zurückführt.

Source Bereich durch den Willen zur gemeinsamen Softwareentwicklung und mithin durch einen gemeinsamen Zweck gekennzeichnet ist, allerdings fehlt es, aufgrund der im Hinblick auf die Rechtseinräumung im Vordergrund stehenden Ausrichtung auf eine einmalige Leistung, an der für eine BGB-Gesellschaft erforderlichen Förderungspflicht der Mitglieder.[256]

Die in der GPL enthaltenen Verpflichtungen treffen den Nutzer nicht unmittelbar und unbedingt, sondern sind von dessen weiteren Absichten abhängig. Es obliegt somit allein dem Nutzer, ob er die durch die GPL statuierten Pflichten, auslösen will. Für eine gesellschaftsvertragliche Förderungspflicht reicht dies nicht aus, weshalb eine Anwendung der §§ 705 ff. BGB ausscheiden muss.

Im Hinblick darauf, dass zumindest die Rechtseinräumung auf eine einmalige Leistung gerichtet ist, die im Übrigen auch in keinem Gegenseitigkeitsverhältnis zu den dauerhaften Vertriebspflichten des Nutzers steht,[257] könnte diesbezüglich eine Anwendung der Vorschriften über die Handschenkung gem. § 516 BGB angebracht sein.

2.Bereicherung

Die für eine Einordnung als Handschenkung erforderliche Bereicherung liegt mit der Rechtsverschaffung vor, um die das Vermögen des Nutzers vermehrt wird.

3.Unentgeltlichkeit der Zuwendung

Problematisch könnte aber die Voraussetzung der Unentgeltlichkeit der Zuwendung sein. Aufgrund der in den Ziff. 1, 2 GPL enthaltenen Verpflichtungen werden in der Literatur teilweise Bedenken daran geäußert, ob die für eine Handschenkung erforderliche

[256] *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 382 ff; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 921 ff; *Spindler* in: Festg. Schricker, S. 554 f; im Grundsatz übereinstimmend auch *Jaeger / Metzger*, Open Source Software, Rn. 199; *Bartosz*, MR 2005, 40, 40 f.

[257] So auch *Spindler*, Rechtsfragen bei Open Source, D. Rn. 8; *Deike*, CR 2003, 9, 14; a.A. *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 905, der den Begriff der Gegenseitigkeit weit auslegt und davon ausgeht, dass die Einhaltung der Vertriebspflichten in einem Gegenseitigkeitsverhältnis zur Einräumung der Rechte stehe.

Unentgeltlichkeit gegeben ist.[258] Berücksichtigt man in diesem Zusammenhang, dass von einer Unentgeltlichkeit in der Rechtsprechung bereits dann nicht mehr ausgegangen wird, wenn eine konditionale[259] oder kausale[260] Verknüpfung zwischen Leistungspflicht und Übertragung der Zuwendung besteht, so erscheinen die Bedenken sogar berechtigt zu sein. Immerhin muss der Erwerber eigene Bearbeitungen unter den Voraussetzungen der Ziff. 2 GPL wieder herausgeben und auch während des Vertriebs von Open Source Software trifft ihn die Pflicht, zur Offenlegung des Quellcodes und zum Open Source Angebot an jedermann.[261] Dies könnte dafür sprechen eine direkte Verknüpfung zwischen Rechtsverschaffung und Einhaltung der GPL anzunehmen.

Zweifel hieran bestehen jedoch, da die in der GPL enthaltenen Verpflichtungen den Erwerber nicht generell, sondern nur in den Fällen treffen, in denen er Bearbeitungen der Open Source Software vornimmt und diese auch vertreiben will. Fehlt es dem Erwerber hingegen an solchen Vertriebsabsichten, betrifft ihn die Pflicht zur Offenlegung des Quellcodes nicht. In diesen Fällen kann daher weder von einer Gegenleistung, noch von einer Bedingung ausgegangen werden, unter der die Zuwendung erfolgt.

An der Unentgeltlichkeit der Zuwendung ändert sich zudem auch dann nichts, wenn mit einem Open Source Projekt letztlich auch wirtschaftliche Ziele verfolgt werden, weil etwa Dienstleistungen im Zusammenhang mit der Software angeboten, oder durch die kostenlose Verbreitung eine Etablierung eines neuen Produkts angestrebt wird.[262] Solche Bestrebungen werden sich in etlichen Open Source Projekten feststellen lassen, allerdings ist die Einräumung der Nutzungsrechte in aller Regel nicht unmittelbar an diese Zwecke geknüpft, weshalb es sich lediglich um „Fernziele“ handelt,[263] welche die Open Source Entwicklungen im Hintergrund begleiten, aber nicht dazu führen, dass der Unentgeltlichkeitscharakter der einzelnen Vertragsbeziehungen entfällt.[264]

---

[258] Vgl. *Hoeren* in: Festschrift Kollhosser, 237; *Schiffner*, Open Source Software, S. 233 f; *Sester*, CR 2000, 797, 800; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 952.
[259] Vgl. nur *Kollhosser* in: Müko, § 516 Rn. 18 m.w.N.
[260] Vgl. nur *Kollhosser* in: Müko, § 516 Rn. 19 m.w.N.
[261] Aus diesem Grund nimmt *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 952, ein Gegenseitigkeitsverhältnis zwischen den Leistungen an und lehnt die Unentgeltlichkeit ab.
[262] *Sester*, CR 2000, 797, 800, geht im Hinblick auf die Entwicklung von Linux davon aus, dass hiermit einzig das Ziel der „Entwicklung und Verbreitung eines Konkurrenzprodukts zu Windows“ verfolgt wird. Dieses Ziel soll derart bestimmend für den kostenlosen Vertrieb von Open Source Software sein, dass es nicht um einen kurzfristig abgewickelten Leistungsaustausch, sondern um eine dauerhafte Verwirklichung eines gemeinsamen Projekts gehe.
[263] Ähnlich auch *Spindler*, Rechtsfragen bei Open Source, D. Rn. 9, der von einer nur durch ein übergeordnetes, allgemeines Interesse definierten Gemeinschaft ausgeht.
[264] Vgl. *Spindler*, Rechtsfragen bei Open Source, D. Rn. 9.

Aus dem gleichen Grund entfällt die Unentgeltlichkeit der Zuwendung auch nicht dadurch, dass die Einräumung der Nutzungsrechte letztlich in der Hoffnung erfolgt, künftig am Wissen Anderer zu partizipieren.[265] Auch dieses Ziel stellt lediglich eine von mehreren Motivationen dar, ohne jedoch derart bestimmend zu sein, dass sich hierdurch an der Unentgeltlichkeit der Zuwendung etwas ändern würde.[266]

Zum Zeitpunkt des Vertragsschlusses treffen den Nutzer somit keine Verpflichtungen, die gegen eine Unentgeltlichkeit der Rechtsverschaffung sprechen würden. Vielmehr erhält er durch den Abschluss der GPL einen Vermögensgegenstand, der zwar auf gewisse Verwertungsrechte beschränkt ist,[267] innerhalb dieses Rahmens aber bedingungslos genutzt werden kann. Es fehlt an einer direkten Verknüpfung zwischen der Rechtsverschaffung und der Einhaltung der GPL.[268]

Die Pflichten, die aus der GPL folgen, knüpfen nicht unmittelbar an die Rechtsverschaffung an, sondern sind von den weiteren Nutzungsabsichten des Erwerbers abhängig. In der Literatur werden sie deshalb auch teilweise als „nachhängende Verpflichtungen“ bezeichnet,[269] ohne dass sich hierdurch etwas an der Unentgeltlichkeit der Zuwendung ändern würde.

4.Entreicherung

Die für die Handschenkung wesensmäßige Entreicherung liegt im Rahmen der GPL darin, dass der Schenker nach dem Vollzug der Schenkung in seinem von § 69 c UrhG vermittelten Ausschließlichkeitsrecht eingeschränkt wird.[270]

[265] *Sester*, CR 2000, 797, 800.
[266] *Spindler*, Rechtsfragen bei Open Source, D. Rn. 9.
[267] *Metzger / Jaeger*, GRUR Int. 1999, 839, 847, sprechen von "nachhängenden Verpflichtungen"; *Spindler*, Rechtsfragen bei Open Source, D. Rn. 8.
[268] *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 386.
[269] So die Bezeichnung bei *Metzger / Jaeger*, GRUR Int. 1999, 839, 847; ähnlich auch *Spindler*, Rechtsfragen bei Open Source, D Rn. 8, der von einer von vorneherein bestehenden Beschränkung der Verwertungsrechte ausgeht.
[270] Vgl. *Spindler*, Rechtsfragen bei Open Source, D. Rn. 7, der jedoch die Aufspaltung zwischen Programmüberlassung und Rechtsverschaffung unberücksichtigt lässt; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 380 f.

Ergebnis

Insgesamt lässt sich somit festhalten, dass die Hauptleistung der GPL in einer einmaligen unentgeltlichen Zuwendung besteht, wie sie für eine Handschenkung typisch ist. Da es der GPL zudem an Regelungen fehlt, die als vollziehbare Auflage im Sinne des § 525 BGB einzuordnen wären,[271] lässt sie sich vertragstypologisch als Lizenzvertrag einordnen, auf den im Wesentlichen die Regelungen der Handschenkung im Sinne von § 516 BGB Anwendung finden.

Zusammenfassung

Die Untersuchung der Erwerbsvorgänge im Rahmen des Open Source Modells hat gezeigt, dass hierbei eine Differenzierung zwischen dem Erwerb der Programmkopie und dem der weitergehenden Nutzungsrechte zu erfolgen hat.
Wird die Programmkopie eines Open Source Programms entgeltlich überlassen, was in der Praxis häufig mit der Übergabe eines Datenträgers einhergeht, so ist das zugrunde liegende Rechtsgeschäft als Kaufvertrag im Sinne von § 433 BGB zu qualifizieren. Erfolgt die Überlassung der Programmkopie hingegen unentgeltlich, so hat sich herausgestellt, dass das zugrunde liegende Rechtsgeschäft als Handschenkung im Sinne von § 516 BGB einzuordnen ist, obwohl es aufgrund der beliebigen Reproduzierbarkeit der Programmkopie eigentlich an der von § 516 BGB vorgesehenen Entreicherung des Schenkers fehlt.

Ebenfalls als Handschenkung im Sinne von § 516 BGB ist auch die GPL einzuordnen. Die Untersuchung hat jedoch gezeigt, dass der Abschluss der GPL nicht zwangsläufig mit jedem Erwerb der Programmkopie einhergeht. Vielmehr ist der Nutzer erst dann auf den Abschluss der GPL angewiesen, wenn er die erworbene Programmkopie nicht nur bestimmungsgemäß nutzen, sondern urheberrechtlich zustimmungsbedürftige Handlungen im Sinne von § 69 c UrhG vornehmen will und hierfür auf die Einräumung weitergehender Nutzungsrechte angewiesen ist.

[271] Als vollziehbarer Auflage, wie sie von § 525 BGB vorausgesetzt wird, käme innerhalb der Regelungen der GPL einzig Ziff. 4 GPL in Betracht, die allerdings als Konsequenz aus einem Lizenzverstoß lediglich den automatischen Wegfall der Nutzungsrechte vorsieht. Im Gegensatz zu einer vollziehbaren Auflage räumt Ziff. 4 GPL dem Urheber aber keine Möglichkeit ein, die in der GPL enthaltenen Entwicklungs- und Vertriebsbedingungen, tatsächlich durchzusetzen, weshalb die Annahme einer Auflagenschenkung aus vertragstypologischer Sicht ausscheidet, vgl. auch *Spindler*, Rechtsfragen bei Open Source, D. Rn. 10; *Jaeger / Metzger*, Open-source-Software, Rn. 216; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 387 f.

Kapitel 4

E. Open Source Software im Kontext des Internationalen Privatrechts

Die Weiterentwicklung und der Vertrieb von Open Source Software erfolgen heutzutage überwiegend über das Internet. Aufgrund der durch Entwicklungsplattformen[272] gegebenen Möglichkeit Software nicht nur kollaborativ, sondern auch dezentral, zu entwickeln, setzt die Mitarbeit an Open Source Projekten nur noch eine Internetverbindung voraus. Der Aufenthaltsort des jeweiligen Entwicklers spielt – jedenfalls aus tatsächlicher Sicht – keine Rolle mehr.

Neben der international verteilten Entwicklung von Programmen vollzieht sich aber auch deren Vertrieb nahezu weltweit. Wird ein Programm im Internet zum Download bereitgehalten, kann von jedem beliebigen Ort aus hierauf zugegriffen werden. Die Staatsgrenzen setzten den tatsächlichen Erwerbsmöglichkeiten keine Hindernisse mehr entgegen und scheinen daher für den Open Source Bereich an Bedeutung verloren zu haben.

Zur Verdeutlichung der internationalen Belange im Open Source Bereich sei der Untersuchung folgendes Beispiel vorangestellt:

Ein deutscher Verbraucher sucht im Internet ein Programm, mit dem er seine Fotos besser sortieren kann. Nach einer Suche bei Google landet er auf der in Amerika gehosteten Internetseite von t-mobile.com, auf der ein passendes Open Source Programm getestet und für gut befunden wurde. Das Programm wird von einem Projekt entwickelt, an dem 10 Entwickler beteiligt sind, die allesamt auch schöpferische Beiträge geleistet haben. Die Entwickler kommen sowohl aus Frankreich, der Schweiz und Russland.

Der deutsche Verbraucher will das Programm nun herunterladen und bekommt hierfür 10 Downloadmöglichkeiten (auf Mirrors) angeboten.[273] Er entscheidet sich für einen Download vom australischen Server, da dieser in der alphabetischen Liste weit oben aufgeführt ist.

---

[272] Verbreitete Entwicklungsplattformen sind bspw. Sourceforge Enterprise (http://www.sourceforge.net), CollabNet Enterprise Edition (http://www.collab.net), Rational Rose Enterprise (http://www.rational.com) und Codebeamer (http://www.intland.com).

[273] Beispielhaft sei der Datei-Download von Projekten angeführt, die bei Sourceforge (http://www.sourceforge.net) gehostet sind. Hier stehen letztlich Mirror in mehr als 10 unterschiedlichen

Kommt es nun durch die Verwendung des Programms zu einem völligen Verlust sämtlicher Fotos, stellt sich die Frage, nach welcher Rechtsordnung sich die Rechte des Verbrauchers beurteilen.

Solche und ähnliche Erwerbskonstellationen kommen im Internet alltäglich vor. Aufgrund des Beispiels wird ersichtlich, wie komplex die Frage nach der anwendbaren Rechtsordnung im Open Source Bereich sein kann. Internationale Belange wird es bei jedem umfangreicheren Programm geben, sei es, weil die Urheber nicht alle aus ein und demselben Land kommen, oder weil das Programm in verschiedene Länder heruntergeladen wurde. Daher ist es für die Beurteilung der Rechtsverhältnisse unerlässlich die Rechtsordnung zu klären, die für die Beurteilung der daraus erwachsenden Rechtsbeziehungen maßgeblich sein soll. In Deutschland richtet sich diese grundsätzlich nach den Regelungen des Internationalen Privatrechts (IPR) und hierbei im Besonderen – wenn auch nicht abschließend – nach den Vorschriften der Art. 3 ff und 27 ff EGBGB.

I. Urheberrechtsstatut

Eine erste Ausnahme von den Vorschriften der Art. 3 ff und 27 EGBGB hat im Zusammenhang mit den Immaterialgüterrechten zu erfolgen, da es im EGBGB an einer Regelung hierzu fehlt. Allerdings enthält auch das UrhG keine Vorschriften, weshalb sich das IPR des Immaterialgüterrechts im Wesentlichen nach ungeschriebenen Regeln richtet.[274] Nach Auffassung des BGH, der auch die überwiegende Ansicht der Literatur folgt, ist für urheberrechtliche Fragestellungen an das Recht des Schutzlandes anzuknüpfen (sog. Schutzlandsprinzip).[275] Danach sind Immaterialgüterrechte in Wirkung und Geltung auf das Gebiet des Staates begrenzt, der sie individuell verleiht oder unter bestimmten Bedingungen generell anerkennt.[276] Ein Urheber hat daher auch kein einheitliches internationales Urheberrecht, sondern vielmehr ein „Bündel“ nationaler Urheberrechte, die sich je nach

---

Ländern (u.a. Taiwan, Irland, Schweiz, Russland, Amerika, Japan, Brasilien und Australien) zur Verfügung, von denen der Download erfolgen kann.

[274] Vgl. nur *Dreier* in: Dreier / Schulze, Vor §§ 120 ff Rn. 27.

[275] Vgl. nur BGH GRUR 1999, 152, 153 – Spielbankaffaire; GRUR 1992, 697, 698 – ALF; *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 28 m.w.N.

[276] Sog. Territorialitätsprinzip, vgl. *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 28; BGH GRUR 1999, 152, 153 – Spielbankaffaire; BGH GRUR 1994, 798 – Folgerecht bei Auslandsbezug; BVerfG GRUR 199, 438, 441 – Bob Dylan.

Staatsgebiet unterscheiden.[277] Deutsches Recht ist hiernach immer dann zugrunde zu legen, wenn es um Entstehung, erste Inhaberschaft, sowie den Inhalt und Umfang des Urheberrechts geht.[278] Daher muss die Urheberschaft für jedes Land gesondert geprüft werden.

Ein angestellter amerikanischer Programmierer könnte z.B. ein deutsches Gericht um Schutz gegen Urheberrechtsverletzungen anrufen, während er diese Möglichkeit in den USA mangels eigener Urheberschaft[279] nicht hätte. Der amerikanische Programmierer würde daher lediglich für Urheberrechtsverletzungen in Deutschland als Urheber angesehen werden, während sein Arbeitgeber dies für Verletzungen wäre, die in den USA stattgefunden haben.

Ebenso richten sich die Übertragbarkeit und Schutzdauer des Urheberrechts sowie die Rechtsfolgen bei dessen Verletzung nach deutschem Recht.[280]

Hieraus folgt, dass deutsches Recht immer dann zur Anwendung kommt, wenn Schutz vor Urheberrechtsverletzungen begehrt wird, die in Deutschland stattgefunden haben.[281] Damit ist jedoch noch nicht geklärt, welches Recht in den Fällen einer grenzüberschreitenden Verletzungshandlung anzuwenden ist, bei denen also entweder die eigentliche Verletzung nur zum Teil im Schutzland erfüllt wurde oder der tatbestandliche Erfolg erst in einem anderen Schutzland eingetreten ist. Hierbei interessieren im Rahmen der vorliegenden Untersuchung vor allem solche Verletzungshandlungen, die sich typischerweise im Zusammenhang mit der Verbreitung von Open Source Software im Internet ergeben.

1. Vervielfältigungsrecht

Das deutsche Vervielfältigungsrecht i.S.d. §§ 15 Abs. 1, 16, 69 c Nr. 1 UrhG ist grundsätzlich immer dann betroffen, wenn eine Vervielfältigungshandlung auf deutschem Boden stattfindet. Sofern es um Werknutzungen im Internet geht, sind im Hinblick auf die Frage nach dem anwendbaren Recht die einzelnen Verletzungshandlungen jeweils gesondert zu beurteilen und insbesondere auch vom Akt des öffentlichen Zugänglichmachens zu unterscheiden.

---

[277] Vgl. nur BGH GRUR 1999, 152, 153 – Spielbankaffaire; *Kegel* in: Soergel, Art. 12 Anh. Rn. 16; *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 121; ebenso *Dreier* in: Dreier / Schulze, Vor §§ 120 ff Rn. 28; als „Flickenteppich“ nationaler Schutzrechte bezeichnet vom BVerfGE 81, 208, 223 – Bob Dylan.

[278] Vgl. nur *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 127; sowie *Dreier* in: Dreier / Schulze, Vor §§ 120 ff Rn. 30 jeweils m.w.N.

[279] Im Rahmen sog. *work made for hire* gem. 17 U.S.C. § 101 ist der Arbeitgeber Inhaber der Urheberrechte, so dass nur dieser den Rechtsweg bestreiten könnte.

[280] Vgl. nur *Dreier* in: Dreier / Schulze, Vor §§ 120 ff Rn. 30 m.w.N.

[281] *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 130.

Wenn Software auf einen Server geladen wird, um sie von dort aus zum Download bereitzuhalten, ist auf diesen Sachverhalt sowohl das Recht des Staates anwendbar, in dem der Upload auf den Server stattgefunden hat, als auch dasjenige, in den die Software heruntergeladen wurde.[282] Zudem ist auch das Recht des Staates anwendbar, in dem der Server steht.[283]

2.Recht der öffentlichen Zugänglichmachung

Neben dem Vervielfältigungsrecht liegt bei der Verbreitung von Software über das Internet ein besonderes Augenmerk auf dem Recht der öffentlichen Zugänglichmachung i.S.v. §§ 19 a, 69 c Nr. 4 UrhG. Dies liegt an den typischen Verbreitungsformen, die das Internet bietet und die sich dadurch auszeichnen, dass Software auf einen Server hochgeladen wird und von dort aus der Allgemeinheit zum Download bereitsteht. Zweifelsfrei wird durch das Downloadangebot das Recht der öffentlichen Zugänglichmachung betroffen. Umstritten ist jedoch, in welchen Ländern dadurch in das Recht auf öffentliche Zugänglichmachung eingegriffen wird. Im Wesentlichen stehen sich zwei Ansätze gegenüber, die sich vor allem in der Beurteilung unterscheiden, worin die maßgebliche Verwertungshandlung liegt.

So wird zum Teil vertreten, dass allein das Bereithalten durch den Serverbetreiber, als Nutzungstatbestand der öffentlichen Zugänglichmachung anzusehen sei,[284] während die wohl h.M. auch die Abrufmöglichkeit hierunter fasst.[285]

Sofern allein im Bereithalten der Daten eine maßgebliche Verwertungshandlung gesehen wird, hätte dies zur Folge, dass das Recht auf öffentliche Zugänglichmachung ausschließlich in dem Land betroffen ist, in dem der jeweilige Server steht.[286] Aufgrund der durch das Internet bestehenden beliebigen Verlegungsmöglichkeiten des Servers und der dadurch

---

[282] Vgl. *Koch*, GRUR 1997, 417, 423 sowie 425 m.w.N.; *Loewenheim* in: Schricker, § 16 Rn. 23.
[283] Vgl. *Hoeren* in: Hoeren / Sieber, Handbuch Multimedia-Recht, 7.10 Rn. 16 ff; *Spindler* in: Leible, Morpheus, Napster & Co., S. 163 ff; *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 33.
[284] Sog. „Country-of-Upload-Regel“, die in unterschiedlichen Ausprägungen vertreten wird; vgl. *Dieselhorst*, ZUM 1998, 293, 299 f; *Gaster*, ZUM 1995, 740, 745; *Koch*, CR 1999, 121, 123; ebenso *Schack*, MMR 2000, 59, 64, der sich für den Serverstandort ausspricht.
[285] LG Hamburg GRUR-RR 2004, 313, 315 – Thumbnails; *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 145; *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 41; sowie *v. Welser* in: Wandtke / Bullinger, Vor §§ 120 ff Rn. 19, jeweils m.w.N.
[286] Dieser Ansatz wurde auch noch von Europäischen Union in ihrem „*Grünbuch über Urheberrechte und verwandte Schutzrechte in der Informationsgesellschaft*“ (KOM (95) 382, S. 38, 41 f) verfolgt, indem sie eine Anknüpfung an das Recht des Mitgliedstaats vorschlug, in dem die Dienstleistung ihren Ursprung hat.

gegebenen Gefahr vor Rechtsumgehungen,[287] wird von Vertretern dieser Ansicht teilweise nicht auf den Serverstandort,[288] sondern auf den gewöhnlichen Aufenthaltsort desjenigen abgestellt, der für den Server verantwortlich ist.[289]

Dieser Lösungsansatz hat zweifellos den Charme, dass sich hierdurch oftmals[290] eine Kumulation der anwendbaren Rechtsordnungen vermeiden ließe und es letztlich zu einer eindeutigen Anwendung von Schrankenregelungen kommt.

Problematisch ist an dieser Herangehensweise jedoch, dass sie Rechtsumgehungen geradezu provoziert.[291] Durch die technischen Möglichkeiten, die das Internet bietet, macht es weder einen Unterschied, in welchem Land der Server steht, noch von wo aus er betrieben wird. Während die Verlegung des Serverstandorts nahezu keine zusätzlichen Kosten verursacht, erscheint zwar das Abstellen auf den gewöhnlichen Aufenthaltsort des Serverbetreibers zunächst dazu geeignet zu sein, ein gewisses Hemmnis aufzubauen, jedoch würde hierdurch letztlich nur bewirkt werden, dass die Provider sich künftig gerade in Ländern mit geringem Urheberschutzniveau niederlassen, um von dort aus Server zu betreiben. Daher hätte das isolierte Abstellen auf diese Kriterien einzig die gezielte Verlagerung der Tätigkeit in Länder mit geringem Urheberschutzniveau zur Folge.[292]

Solche Umgehungsversuche werden sich jedoch so lange nicht vermeiden lassen, bis der Urheberschutz weltweit ein einheitlich hohes Niveau erreicht hat.[293] Solange diese Voraussetzung aber nicht gegeben ist, kann zur Vermeidung der Kumulation der anwendbaren Rechtsordnungen weder auf das Kriterium des Serverstandorts noch auf den Sitz des Betreibers abgestellt werden.

---

[287] Auf diese Gefahr weist *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 41, hin; ebenso *Hartmann* in: Möhring / Nicolini, Vor §§ 120 Rn 34; *v. Welser* in: Wandtke / Bullinger, Vor §§ 120 ff. Rn. 19; *Thum* in: Bartsch / Lutterbeck, Das Territorialitätsprinzip im Zeitalter des Internet, S. 137 f; *Schønning*, ZUM 1997, 34, 38.

[288] Gleichwohl auf den Serverstandort abstellend, vgl. *Gaster*, ZUM 1995, 740, 745; *Koch*, CR 1999, 121, 123.

[289] Mit ausführlicher Begründung *Spindler*, IPrax 2003, 412, 416 ff.

[290] Gerade im Hinblick auf die wachsende Bedeutung dezentraler Verteilungssysteme, bei denen die Daten nicht von einem einheitlichen Server heruntergeladen, sondern auf verschiedenen Servern vorgehalten werden, wird es auch nach dieser Ansicht zu einer Kumulation zahlreicher Rechtsordnungen kommen, vgl. *Koch*, CR 1999, 121, 123. Zunehmend problematisch dürfte auch die Beurteilung distributiven Datenaustauschverfahren in Peer-to-Peer-Netzen sein, die durch das sog. Bittorent-Sharing Protokoll möglich wurden und bei denen jeder Nutzer bereits während des Downloadvorgangs die heruntergeladenen Datenteile wiederum verteilt und somit zugleich zu deren (Server-) Anbieter avanciert.

[291] Vgl. *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 41; *Hartmann* in: Möhring / Nicolini, Vor §§ 120 Rn 34; *v. Welser* in: Wandtke / Bullinger, Vor §§ 120 ff. Rn. 19; *Thum* in: Bartsch / Lutterbeck, Das Territorialitätsprinzip im Zeitalter des Internet, S. 137 f; *Schønning*, ZUM 1997, 34, 38.

[292] Zur Gefahr der Delokalisierung vgl. *Thum* in: Bartsch / Lutterbeck, Das Territorialitätsprinzip im Zeitalter des Internet, S. 137 f; *Gesmann-Nuissl* in: Ensthaler / Bosch, u.a., S. 428 ff; *Schønning*, ZUM 1997, 34, 38.

[293] *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 41.

Vielmehr ist in den Fällen der öffentlichen Zugänglichmachung im Internet nicht nach dem Herkunftslandsprinzip, sondern entsprechend der Bogschen-Theorie vorzugehen.[294] Danach ist jeder Ort als Erfolgsort der Urheberrechtsverletzung anzusehen, in dem die Möglichkeit des Downloads besteht. Es kommt somit zwangsläufig zu einer Kumulation der anwendbaren Rechtsordnungen, da durch das öffentliche Zugänglichmachen der Software auf einem Server und die hierdurch vermittelte weltweite Abrufbarkeit, das Recht sämtlicher Staaten anwendbar ist.[295]

II. Vertragsstatut

Im Gegensatz zu den Immaterialgüterrechten, für welche die ungeschriebenen Regeln des Urheberrechtsstatuts gelten, richten sich Fragen im Zusammenhang mit Urheberrechtsverträgen grundsätzlich nach den allgemeinen Regelungen des internationalen Privatrechts und somit nach den Vorschriften der Art. 3 ff und 27 ff EGBGB. Dies gilt nach allgemeiner Ansicht zumindest für den schuldrechtlichen Teil des Urheberrechtsvertrags.[296]

Soweit es hingegen um eine Beurteilung der in den Verträgen enthaltenen Verfügungen geht, ist umstritten, ob sich diese nach dem Vertrags-[297] oder nach dem Urheberrechtsstatut richten.[298]

---

[294] LG Hamburg GRUR-RR 2004, 313, 315 – Thumbnails; *Katzenberger* in: Schricker, Vor §§ 120 ff. Rn. 145; sowie *v. Welser* in: Wandtke / Bullinger, Vor §§ 120 ff. Rn. 19, jeweils m.w.N.; *Hilty / Peukert*, Interessenausgleich im Urheberrecht, S. 62.

[295] Um diese, von vielen als unpraktikabel empfundene, Kumulation zu vermeiden, wird teilweise eine einschränkende autonome Anknüpfung vorgeschlagen (ausführlich zu den bestehenden Alternativvorschlägen sowie zu der Kritik daran, vgl. *Thum* in: Bartsch / Lutterbeck, Das Territorialitätsprinzip im Zeitalter des Internet, S. 136 ff; ebenso *Gesmann-Nuissl* in: Ensthaler / Bosch, u.a., S. 428 ff). Die Vorschläge in der Literatur gehen dabei zumeist dahin, dass nicht ausschließlich auf die theoretische Abrufbarkeit abgestellt, sondern auch die Intention des Anbietenden (vgl. *Junker*, Anwendbares Recht und internationale Zuständigkeit bei Urheberrechtsverletzungen im Internet, S. 359, der die Intention allerdings analog zum objektiven Erklärungsempfänger aus der Sicht eines objektiven Dritten beurteilt) oder auch die Relevanz des Angebots am Markt berücksichtigt wird (vgl. *Hoeren*, NJW 1998, 2849, 2851, der sich für den Einsatz des Marktortsprinzips als allgemeine Anknüpfungsregel im „Cyberspace“ ausspricht). Teilweise wird aber auch für internationale Urheberrechtsverletzungen die Anwendbarkeit des Schutzlandsprinzips bezweifelt und stattdessen auf das Recht des Ursprungslandes abgestellt (vgl. nur *Schack*, GRUR Int. 1985, 523, 524; *Schack*, MMR 2000, 59, 63 f.)
Gegen diese Lösungsansätze spricht jedoch, dass sie allesamt nicht dazu geeignet zu einer Einschränkung der anwendbaren Rechtsordnungen zu führen, die auch im Hinblick auf die erforderliche Rechtssicherheit angemessen ist (vgl. die kritischen Auseinandersetzung mit den jeweiligen Ansätzen bei *Thum* in: Bartsch / Lutterbeck, Das Territorialitätsprinzip im Zeitalter des Internet, S. 136 ff; ebenso *Gesmann-Nuissl* in: Ensthaler / Bosch, u.a., S. 428 ff).

[296] Vgl. *Obergfell* in: Reithmann / Martiny, Rn. 1783 ff; *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 148 m.w.N.

[297] So die sog. Einheitstheorie, vgl. BGH GRUR 1959, 331, 333 – Dreigroschenroman; OLG Frankfurt GRUR 1998, 141, 142 – Mackintosh-Entwürfe; eingehend *Katzenberger* in: Festg. Schricker, S. 249 ff; *Katzenberger*

Während die sog. Spaltungstheorie den verfügenden Teil vom schuldrechtlichen getrennt nach dem Recht des Schutzlandes beurteilen will,[299] soll der Urheberrechtsvertrag nach der sog. Einheitstheorie generell dem Vertragsstatut unterliegen.[300]

Eine Annäherung zwischen der Spaltungs- und der Einheitstheorie findet sich bei Fragen nach der Entstehung des Urheberrechts, dessen erste Inhaberschaft, der Schutzfähigkeit, der Übertragbarkeit und der Aktivlegitimation bei vertraglichen Rechtseinräumungen sowie bei den Rechtsfolgen von Rechtsverletzungen, der Schutzdauer und des Erlöschens des Urheberrechts. Soweit es also um Fragen der dinglichen Rechtseinräumung geht – worunter sowohl die Einräumung ausschließlicher als auch einfacher Nutzungsrechte zu fassen ist –[301] bestimmt sich auch nach Auffassung der Einheitstheorie die anwendbare Rechtsordnung nach dem Schutzlandsprinzip,[302] weshalb sich das Ergebnis in diesem Bereich nicht von dem unterscheidet, welches bei einer Anwendung der Spaltungstheorie erzielt würde.

Der Bereich, in dem das Vertragsstatut über die anwendbare Rechtsordnung entscheidet, ist somit begrenzt. Gleichwohl werfen auch die verbleibenden schuldrechtlichen Regelungsbereiche Fragen auf, die für die rechtliche Beurteilung von Open Source Software von erheblicher Bedeutung sind. Dabei ist insbesondere an die bereits in der Literatur mehrfach behandelten schuldrechtlichen Fragen der Wirksamkeit des Haftungs- und Gewährleistungsausschlusses[303] aber auch die sonstigen Fragen der Reichweite der GPL zu denken.[304] Bei diesen handelt es sich um schuldrechtliche Regelungsbereiche, so dass sich die anwendbare Rechtsordnung nach dem Vertragsstatut und somit grundsätzlich nach der Rechtswahl der Parteien richtet.

---

in: Schricker, Vor §§ 120 ff Rn. 149 m.w.N.; ebenso *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 50; *Ulmer*, Die Immaterialgüterrechte im internationalen Privatrecht, S. 48 f.

[298] So die Spaltungstheorie, vgl. BGH NJW 1988, 1847, 1848, die sich allerdings wohl weniger in der Rechtsprechung durchgesetzt hat, aber vor allem im Umfeld der IPR-Literatur vertreten wird; vgl. nur *Obergfell* in: Reithmann / Martiny, Rn. 1786; *Kegel / Schurig*, Internationales Privatrecht, § 17 VII, S. 560 f; *v. Welser* in: Wandtke / Bullinger, Vor §§ 120 ff Rn. 22.

[299] Vgl. nur *Obergfell* in: Reithmann / Martiny, Rn. 1786 m.w.N.

[300] Vgl. nur *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 149 m.w.N.; *Ulmer*, Die Immaterialgüterrechte im internationalen Privatrecht, S. 48 f.

[301] Im Hinblick auf die dingliche Qualifikation von einfachen Nutzunsrechten ist dies sehr umstritten, die wohl h.M. geht aber von einer dinglichen Wirkung aus, vgl. nur *Schricker* in: Schricker, Vor §§ 28 Rn. 61 mit Nachweisen zum divergierenden Meinungsstand; als quasi-dingliches Recht wird das einfache Nutzungsrecht von *Rehbinder*, Urheberrecht, Rn. 306, bezeichnet.

[302] BGH GRUR Int. 1998, 427, 429 – Spielbankaffaire; *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 150 m.w.N.

[303] Ausführlich behandelt werden die Fragen der Wirksamkeit der GPL im Hinblick auf die §§ 305 ff BGB vor allem im Zusammenhang mit dem in Ziff. 11, 12 GPL enthaltenen Haftungs- und Gewährleistungsausschluss, vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 219 ff; *Schiffner*, Open Source Software, S. 241 ff; *Marly*, Softwareüberlassungsverträge, Rn. 440 ff; *Omsels* in: FS Hertin, S. 147 ff; *Spindler*, Rechtsfragen bei Open Source, D. Rn. 14 ff.

[304] Zur Frage nach der Reichweite der Lizenzierungspflicht siehe unten Kapitel 5 S. 115 ff.

1.Rechtswahl der Vertragsparteien, Art. 27 EGBGB

Die Bedeutung der Rechtswahl der Vertragsparteien für die anwendbare Rechtsordnung folgt aus der Regelung des Art. 27 EGBGB. Hiernach kommt es für die Bestimmung des anwendbaren Rechts zunächst einmal auf den Willen der Vertragsparteien an, der auf eine darin enthaltene Rechtswahl zu überprüfen ist. Da beim Erwerb von GPL-lizenzierten Programmen zwischen dem Vertrag, mit dem die Programmkopie erworben wird und der GPL, welche die weitergehenden Nutzungsrechte beinhaltet, zu unterscheiden ist,[305] sind diese Verträge getrennt auf eine etwaige Rechtswahl zu untersuchen.

a.Erwerb der Programmkopie

Soweit es um den Erwerb der Programmkopie geht, liegt diesem häufig ein Schenkungs- oder auch ein Kaufvertrag zugrunde.[306]

Unabhängig davon, ob der Nutzer die Programmkopie entgeltlich oder unentgeltlich erworben hat, ist zu berücksichtigen, dass die Regelungen der GPL sich auf eine in diesem Vertrag enthaltene Rechtswahl nicht auswirken. Die GPL entfaltet ihre Wirkung ausschließlich für die Rechtseinräumung,[307] weshalb selbst eine in der GPL enthaltene Rechtswahl keine Auswirkungen auf diesen Vertrag hätte.

Im Hinblick auf Art. 27 EGBGB kommt es somit einzig auf den Vertrag an, mit dem die Programmkopie an den Nutzer überlassen wurde. Aufgrund der Vielzahl an Regelungsmöglichkeiten kann eine Untersuchung, der in der Praxis beispielsweise durch die Distributoren, verwendeten Verträge an dieser Stelle nicht erfolgen.

Im Folgenden wird vielmehr von der „klassischen“ Erwerbskonstellation ausgegangen, in der die Programmkopie ohne ausdrückliche Regelungen zwischen den Parteien mittels Download unentgeltlich überlassen und somit konkludent ein Schenkungsvertrag geschlossen wurde.

Legt man einen solchen Erwerbsvorgang zugrunde, lässt sich im Zusammenhang mit Art. 27 Abs. 1 S. 2 EGBGB zunächst feststellen, dass eine ausdrückliche Rechtswahl mangels expliziter Vertragsregelungen fehlt. Auch aus den Umständen des Falles lässt sich beim gewöhnlichen Download von einem Server bzw. einem Mirror regelmäßig keine Rechtswahl

[305] Vgl. hierzu die Ausführungen oben Kapitel 3 S. 40 ff.
[306] Zur vertragstypologischen Einordnung der Programmüberlassung siehe Kapitel 3 S. 47 ff.
[307] Zu den Gründen für eine getrennte Beurteilung vgl. Kapitel 3 S. 40 ff.

entnehmen, so dass Art. 27 EGBGB keine Anwendung findet. Mangels Rechtswahl bestimmt sich die anwendbare Rechtsordnung daher grundsätzlich nach Art. 28 EGBGB.

b.Erwerb der weitergehenden Nutzungsrechte

Soweit der Nutzer die weitergehenden Nutzungsrechte an dem Programm durch den Abschluss der GPL erworben hat, stellt sich ebenfalls die Frage, ob eine Rechtswahl im Sine von Art. 27 EGBGB gegeben ist.

Im Hinblick auf die Regelungen der GPL lässt sich dabei zunächst festhalten, dass eine ausdrückliche Rechtswahl im Sinne von Art. 27 Abs. 1 S. 2 EGBGB, darin nicht enthalten ist. Allerdings wird in der Literatur vereinzelt vertreten, dass sich die Rechtswahl aus den Umständen des Falles ergeben solle, indem die GPL auf das US-Recht hinweist und insofern die Voraussetzungen von Art. 27 Abs. 1 S. 2 var. 2 EGBGB gegeben seien.[308] Zur Begründung werden sowohl die im GPL-Lizenztext gewählte englische Sprache als auch deren Entwicklung vor dem Hintergrund des amerikanischen Rechts angeführt.[309]

Gegen das Bestehen einer solchen Rechtswahl ist jedoch einzuwenden, dass Ziff. 11 und 12 GPL darauf hindeuten, dass es sich bei der GPL um einen Mustervertrag handelt,[310] der international Anwendung finden sollte, ohne die jeweiligen Verwender an eine bestimmte Rechtsordnung zu binden.

Hierfür spricht sowohl der in Ziff. 11 enthaltene Zusatz *„...soweit dies gesetzlich zulässig ist..."*, als auch der in Ziff. 12 GPL enthaltene Vorbehalt, der gegenüber dem geltenden Recht Einschränkungen der Gewährleistungs- und Haftungsregelungen vorsieht.[311] Im Übrigen ist mit der GPL gerade auch bezweckt, dass diese als Mustervertrag für möglichst viele Entwicklungen verwendet wird. Diesem Zweck würde jedoch eine Rechtswahl entgegenlaufen, da durch sie „nicht-amerikanische" Urheber dazu gezwungen wären ihre Werke einer Rechtsordnung zu unterstellen, die ihnen im Regelfall nicht bekannt ist.

Zusammenfassend ist daher festzuhalten, dass der GPL weder eine ausdrückliche, noch eine sich aus den Umständen des Falls ergebende, Rechtswahl entnehmen lässt. Das anwendbare

[308] Vgl. *Metzger / Jaeger*, GRUR Int. 1999, 839, 842 Fn. 49; neuerdings a.A. *Jaeger / Metzger*, Open Source Software, Rn. 362.
[309] *Metzger / Jaeger*, GRUR Int. 1999, 839, 842 Fn. 49.
[310] Hierauf weist zu Recht *Deike*, CR 2003, 9, 11, hin.
[311] Vgl. *Deike*, CR 2003, 9, 11; *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 296.

Recht kann daher nicht anhand von Art. 27 EGBGB bestimmt werden, weshalb es grundsätzlich auf Art. 28 EGBGB ankommt.

2.Erwerb durch einen Verbraucher

Eine Ausnahme von der Anwendung des Art. 28 EGBGB könnte jedoch gem. Art. 29 EGBGB geboten sein, wenn ein Verbraucher am jeweiligen Erwerbsvorgang beteiligt ist. Art. 29 EGBGB findet Anwendung, sofern die Lieferung beweglicher Sachen oder die Erbringung von Dienstleistungen Gegenstände eines Verbrauchervertrages sind.

a.Programmkopie als bewegliche Sache im Sinne von Art. 29 EGBGB

Art. 29 EGBGB setzt die Lieferung einer beweglichen Sache voraus. Daher stellt sich zwangsläufig die Frage, ob ein Verbraucher, der eine Programmkopie mittels Download – und somit in unkörperlicher Form – erwirbt, vom Schutzbereich der Vorschrift erfasst wird.

Die Sacheigenschaft von Software ist im Deutschen materiellen Recht seit Langem umstritten.[312] Für erneute Diskussionen hat zuletzt die Schuldrechtsmodernisierung gesorgt, die einige Autoren zum Anlass genommen haben, um die bis dahin bestehenden Rechtsprechungstendenzen für die Zukunft erneut in Frage zu stellen.[313]

Unabhängig von diesen Streitigkeiten ist allerdings im Rahmen der Auslegung des Art. 29 EGBGB zu berücksichtigen, dass die Vorschrift auf dem Europäischen Schuldvertragsübereinkommen (EVÜ) beruht und die dort verwendeten Begriffe gem. Art. 36 EGBGB autonom auszulegen sind.[314] Auf die, vor dem Hintergrund der deutschen Rechtsordnung bestehenden Ansichten zur Sacheigenschaft von Software, kommt es insofern nur bedingt an, da diese für eine Auslegung des Art. 29 EGBGB nicht unmittelbar herangezogen werden können.[315] Vielmehr bieten sich im Rahmen des Art. 29 EGBGB sowohl das EG-Recht als auch internationale Übereinkommen als Orientierungshilfe an.[316] In

[312] Zum Diskussionsstand insgesamt siehe nur *Marly*, Softwareüberlassungsverträge, Rn. 96 ff m.w.N., sowie zu den Änderungen durch die Schuldrechtsmodernisierung Rn. 115 ff.
[313] Vgl. nur *Stichtenoth*, K&R 2003, 105, 107; *Diedrich*, CR 2002, 473, 476.
[314] *Martiny* in: Müko, Art. 36 Rn. 9.
[315] *Martiny* in: Müko, Art. 36 Rn. 9; *Klimek / Sieber*, ZUM 1998, 902, 906; *Deike*, CR 2003, 9, 12.
[316] *Heldrich* in: Palandt, Art. 36 Rn. 1.

diesem Kontext ist insbesondere das UN-Kaufrecht konventionsvergleichend heranzuziehen, dass zwar gem. Art. 2 lit a CISG keine Anwendung auf den Kauf von Waren für den persönlichen Gebrauch findet, dessen Warenbegriff jedoch ebenfalls auf bewegliche Gegenstände beschränkt ist und dem des Art. 29 EGBGB entspricht.[317] Im Zusammenhang mit der Auslegung des vom UN-Kaufrecht verwendeten Warenbegriffs ist es anerkannt, dass hierunter auch Software zu fassen ist, unabhängig davon, ob diese online übertragen oder mittels Datenträger verschafft wurde.[318] Vor diesem Hintergrund ist es gerechtfertigt, Software nicht nur als bewegliche Ware im Sinne des UN-Kaufrechts anzusehen, sondern sich auch bei der Auslegung des Art. 29 EGBGB am Warenbegriff des UN-Kaufrechts zu orientieren.

Hierfür spricht zudem der Wortlaut des Art. 29 EGBGB, der neben den beweglichen Sachen ausdrücklich auch Dienstleistungen erfasst und damit der Körperlichkeit des Vertragsgegenstandes keine Bedeutung beimisst.[319]

Insofern ist, unabhängig von den verschiedenen Auffassungen, die zur Sacheigenschaft von Software im Deutschen materiellen Recht bestehen, von Art. 29 EGBGB auch der Erwerb von Software mittels Download erfasst.[320]

Sofern der Vertrag, welcher der Programmüberlassung zugrunde liegt, unter den in Art. 29 Abs. 1 EGBGB bezeichneten Umständen überlassen wurde, ist somit grundsätzlich von einer Anwendbarkeit der deutschen Rechtsordnung auszugehen. Dies gilt jedenfalls dann, wenn die Softwareüberlassung im Rahmen eines Kaufvertrags erfolgte.

b.Unentgeltliche Programmüberlassung erfasst?

Sofern dem Erwerb der Programmkopie hingegen ein Schenkungsvertrag zugrunde lag, könnte die Anwendbarkeit von Art. 29 EGBGB zweifelhaft sein, da der Verbraucher, im Gegensatz zur entgeltlichen Überlassung, keine Gegenleistung zu erbringen hat und seine Schutzbedürftigkeit mit dem von Art. 29 EGBGB bezweckten Schutz der schwächeren Vertragspartei daher zumindest nicht ohne Weiteres zu vereinbaren scheint.

---

[317] *Magnus* in: Staudinger, Art. 29 Rn. 47 ff; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 146.
[318] *Klimek / Sieber*, ZUM 1998, 902, 906 f; zum UN-Kaufrecht *Endler / Daub*, CR 1993, 601, 604 f; *Mehrings* in: Hoeren / Sieber, Handbuch Multimedia-Recht, Kap. 13.1, Rn. 19.
[319] Vgl. *Endler / Daub*, CR 1993, 601, 604 f.
[320] *Mankowski*, RabelsZ 1999, 203, 232 f; *Lurger* in: Leible, Internationaler Verbraucherschutz, S. 44 f; *Magnus* in: Staudinger, Art. 29 Rn. 50; *Martiny* in: Müko, Art. 29 Rn. 15 m.w.N.

Art. 29 EGBGB wurde im Zuge der Angleichung des Verbraucherrechts zwischen den Mitgliedsstaaten der EG eingeführt und entspricht in seinem Wortlaut im Wesentlichen Art. 5 EVÜ.[321] Der Hintergrund für die Entstehung der verbraucherschützenden Vorschriften ist letztlich in Art. 3 Abs. 1 lit. t, Art. 153 EG zu sehen, wonach die Verbesserung des Verbraucherschutzes zu den Tätigkeitsbereichen der Gemeinschaft gehört. Die Regelungen des EVÜ und mithin auch die des Art. 29 EGBGB wurden durch die Zunahme von grenzüberschreitenden Geschäften erforderlich, die gerade für Verbraucher als regelmäßig schwächere Vertragspartei eine Reihe an spezifischen Gefahren hervorgebracht haben. Durch Art. 29 EGBGB sollte also, die durch die Gemeinschaft gewährte Warenverkehrs- und Dienstleistungsfreiheit (Art. 28 ff., 49 ff. EG) zugunsten der schwächeren Vertragspartei ausnahmsweise eingeschränkt werden.[322]

Dass eine solche Einschränkung auch beim Vorliegen eines unentgeltlichen Geschäfts erforderlich ist, erscheint zunächst nicht zwingend zu sein. Allerdings kann dem Wortlaut des Art. 29 EGBGB ebenso wenig entnommen werden, dass der Anwendungsbereich von Art 29 EGBGB auf entgeltliche Geschäfte beschränkt wäre. Es wird lediglich ein Vertrag „*...über die Lieferung beweglicher Sachen…*“ vorausgesetzt, ohne dass es darauf ankäme, ob dieser Lieferung eine Leistung auf Seiten des Verbrauchers gegenübersteht.

Ein Hinweis auf das Erfordernis einer Entgeltlichkeit des Geschäfts könnte einzig in der Alternative des Finanzierungsgeschäfts in Art. 29 Abs. 1 EGBGB enthalten sein, indem der Anwendungsbereich hierin auf solche Verträge erweitert wird, die *„ …zur Finanzierung eines solchen Geschäfts…“* dienen. Wenn aber die Finanzierung des eigentlichen Geschäfts möglich sein soll, könnte dies dafür sprechen, dass der Normgeber von einer Entgeltlichkeit sämtlicher Vertragstypen ausging, da deren Finanzierung ansonsten nicht erforderlich wäre.

Hiergegen lässt sich einwenden, dass eine Entgeltlichkeit in Art. 29 EGBGB nicht ausdrücklich als Voraussetzung enthalten ist. Auch kann allein aus der Möglichkeit, dass neben dem eigentlichen Vertrag zugleich auch ein Finanzierungsgeschäft besteht, nicht zwangsläufig darauf geschlossen werden, dass für sämtliche Verbraucherverträge – zumindest theoretisch – eine Finanzierungsmöglichkeit bestehen muss.

Im Rahmen des Anwendungsbereichs von Art. 29 EGBGB ist vielmehr maßgeblich auf den damit verfolgten Zweck abzustellen, der darin liegt, den Verbraucher, als regelmäßig

[321] Art. 5 Abs. 1 EVÜ: *Dieser Artikel gilt für Verträge über die Lieferung beweglicher Sachen oder die Erbringung von Dienstleistungen an eine Person, den Verbraucher, zu einem Zweck, der nicht der beruflichen oder gewerblichen Tätigkeit des Verbrauchers zugerechnet werden kann, sowie für Verträge zur Finanzierung eines solchen Geschäfts.*
[322] Amtl. Begründung BT-Drucks. 10/ 504, S. 79 f.

schwächere Vertragspartei, vor allem kollisionsrechtlich zu schützen. Dies setzt aber nicht zwangsläufig voraus, dass dem Vertrag mit dem Verbraucher ein entgeltliches Geschäft zugrunde liegt. Hieraus folgt, dass Art. 29 EGBGB auch auf den Vertrag Anwendung findet, mit welchem dem Verbraucher die Programmkopie unentgeltlich eingeräumt wird.[323]

Sofern ein Verbraucher die Programmkopie eines Open Source Programms erwirbt, kommt Deutsches Recht zur Anwendung, unabhängig davon, ob die Programmkopie entgeltlich oder unentgeltlich überlassen wurde.

c.Erwerb der weitergehenden Nutzungsrechte

Im Rahmen der kollisionsrechtlichen Einordnung des Erwerbs von Open Source Software stellt sich nach der hier vertretenen Ansicht – wonach zwischen dem Erwerb der Programmkopie und dem der Nutzungsrechte zu unterscheiden ist[324] – weiterhin die Frage, ob Art. 29 EGBGB auch auf den Vertrag Anwendung findet, mit dem der Nutzer die Nutzungsrechte erwirbt.

Eine Anwendbarkeit von Art 29 EGBGB scheidet allerdings bereits aus dem Grund aus, da die GPL ausschließlich die Einräumung von Rechten beinhaltet, die mangels Sacheigenschaft aber weder als Lieferung beweglicher Sachen noch als Dienstleistung[325] qualifiziert werden können.

Die Anwendbarkeit von Art. 29 EGBGB kann zudem nicht mit dem Sachzusammenhang zwischen Rechtseinräumung und Überlassung der Programmkopie begründet werden,[326] da die Programmkopie und die Nutzungsrechte zumeist von unterschiedlichen Vertragspartnern überlassen werden und es daher an Umständen fehlt, mit denen sich ein solcher Sachzusammenhang begründen ließe.

Die GPL, mit welcher der Nutzer die weitergehenden Nutzungsrechte eingeräumt bekommt, fällt somit nicht in den Anwendungsbereich des Art. 29 EGBGB. Sofern ein Verbraucher Open Source Software herunterlädt und verändert oder weiterverbreitet, richtet sich das

[323] Im Ergebnis ebenso *Spindler*, Rechtsfragen bei Open Source, C. Rn. 145 ff, allerdings ohne eine Problematisierung, ob Art. 29 EGBGB auch Schenkungsverträge erfasst; ebenso *Jaeger / Metzger*, Open Source Software, Rn. 363; sowie *Deike*, CR 2003, 9, 12, der zwar die Frage aufwirft, ob die von Art. 29 EGBGB erfassten Vertragstypen beim Bezug von Open Source Software betroffen sind, sodann aber nur auf die Sacheigenschaft von Software eingeht.
[324] Vgl. hierzu die Ausführungen im Kapitel 3 S. 40 ff.
[325] *Spickhoff* in: Bamberger / Roth, Art. 29 EGBGB, Rn. 6.
[326] So auch *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 298.

anwendbare Recht daher nach den allgemeinen Vorschriften. Mangels Rechtswahl kommt allerdings nicht Art. 27 EGBGB, sondern Art. 28 EGBGB zur Anwendung.

3.Objektive Anknüpfung nach Art. 28 EGBGB

Nach Art. 28 EGBGB bestimmt sich das anwendbare Recht nach dem Prinzip der objektiven Anknüpfung,[327] wonach der Vertrag dem Recht des Staates unterliegt, zu dem er die engste Verbindung aufweist. Entsprechend Art. 28 Abs. 2 S. 1 EGBGB wird dabei vermutet, dass der Vertrag die engste Verbindung mit dem Staat aufweist, in dem die Partei, welche die charakteristische Leistung zu erbringen hat, im Zeitpunkt des Vertragsschlusses ihren gewöhnlichen Aufenthaltsort bzw. ihre Hauptverwaltung hat.

a.Erwerb der Programmkopie

Dem Erwerb der Programmkopie liegt, sofern sie im Wege des Downloads erfolgt ist, ein Schenkungsvertrag zugrunde,[328] dessen charakteristische Leistung in der Zuwendung des Schenkenden besteht.[329]

Soweit die Programmkopie eines Open Source Programms somit unentgeltlich an den Nutzer übergeben wird, ist im Rahmen von Art. 28 Abs. 2 EGBGB auf den Wohnort bzw. Sitz desjenigen abzustellen, der das Downloadangebot bereithält.

Im Open Source Bereich folgt daraus, dass Deutsches Recht nur dann zur Anwendung gelangt, wenn der Anbieter des Programms seinen Sitz in Deutschland hat. Aufgrund der Vielzahl an internationalen Anbietern, von denen Open Source Software heruntergeladen werden kann, führt dies dazu, dass der Schenkungsvertrag häufig nicht nach Deutschem Recht zu beurteilen sein wird.

Immerhin findet der Erwerb von Open Source Software mittels Download – auf dem vorliegend das Augenmerk liegt – regelmäßig in der Art und Weise statt, dass die Programmkopie auf einem oder mehreren Downloadservern öffentlich bereitgestellt wird und

[327] Vgl. nur *Katzenberger* in: Schricker, Vor §§ 120 ff Rn. 154 ff; *Dreier* in: Dreier / Schulze, Vor §§ 120 ff Rn. 52; *Schack*, Urheber- und Urhebervertragsrecht, Rn. 1143 f.
[328] Vgl. Kapitel 3 S. 40 ff.
[329] OLG Köln NJW-RR 1994, 1026 – Brautgeld; OLG Frankfurt/M GRUR 1998, 141, 142; *Martiny* in: Reithmann / Martiny, Rn. 136 m.w.N.

von dort von jedermann uneingeschränkt heruntergeladen werden kann. Den konkreten Downloadlink erhält der Nutzer auf unterschiedliche Weise. Dies hängt davon ab, wie er auf das Programm aufmerksam wurde. Beliebte Open Source Programme werden beispielsweise häufig in Zeitschriften als kostenlose Alternative zu kommerziellen Lösungen getestet und in diesem Zusammenhang auf die entsprechenden Downloadmöglichkeiten hingewiesen.[330] Im Übrigen gibt es viele Betreiber von Webseiten, die sich mit dem Testen von Open Source Software beschäftigen und ebenfalls auf die Downloadserver hinweisen bzw. diese sogar selbst bereitstellen.[331] Eine weitere Möglichkeit besteht darin, die Software direkt von der Projektseite herunter zu laden, wobei auch diese Projektseiten oftmals externe Links für den eigentlichen Programmdownload bereithalten.

Daneben gibt es im Internet eine Reihe von Webangeboten,[332] die sich sowohl an die Entwickler als auch an die Nutzer wenden. Die Entwickler erhalten im Rahmen dieser Angebote eine Entwicklungsplattform sowie die Möglichkeit das eigene Projekt kostenfrei zu hosten. Dem Nutzer werden umfangreiche Suchmöglichkeiten an die Hand gegeben, mit denen er unter der Masse an Open Source Programmen das passende herausfinden kann. Für den eigentlichen Download des Programms erhält der Nutzer zumeist verschiedene Server zur Auswahl, die als Mirror fungieren und die ständige Verfügbarkeit gewährleisten sollen.

Insgesamt stehen dem Nutzer somit eine Vielzahl von Downloadquellen zum Erwerb von Open Source Software zur Auswahl, deren Anbietern ihren Sitz nicht zwangsläufig in Deutschland haben.[333] Als Folge dessen wird sich der Schenkungsvertrag, mit dem die Programmkopie erworben wurde, oftmals nicht nach Deutschem Recht beurteilen.[334]

---

[330] Mittlerweile befinden sich in nahezu allen Computerzeitschriften hinweise auf Open Source Programme. In der Zeitschrift Chip ist sogar monatlich eine Art „best of“ enthalten, in der Open Source Programme mit Empfehlungen und Tests aufgeführt werden.

[331] Hierbei seien nur Downloadseiten wie http://www.zdnet.com, http://www.download.com oder auch das relativ neue Angebot von T-Online (http://www.softwareload.de) genannt.

[332] Da die Open Source Projekte zumeist ohne eigene Finanzmittel starten, verwenden sie häufig kostenlose Hosting-Anbieter, die den Entwicklern darüber hinaus auch auch Entwicklungsplattformen zur Verfügung stellen. Solche Angebote finden sich u.a. bei http://www.sourceforge.net, http://www.berlios.de, http://www.freshmeat.net und http://www.javaforge.com.

[333] Dabei kommt es im Übrigen nicht darauf an, ob der Downloadlink letztlich auf den eigenen Server oder einen externen Mirror verweist, sofern der Download durch das Anklicken des Links unmittelbar in Gang gesetzt wird. Sofern der Download von einem externer Server (Mirror) erfolgt, wird der Serverbetreiber im Hinblick auf das Vertragsangebot als Bote tätig, so dass es auch in diesem Fall darauf ankommt, wem das eigentliche Downloadangebot zuzurechnen ist.

[334] Ebenso in der Einschätzung *Deike*, CR 2003, 9, 13.

b.Erwerb der weitergehenden Nutzungsrechte

Soweit der Nutzer auf weitergehende Nutzungsrechte angewiesen ist, weil er GPL-lizenzierte Programme verändern oder sonstige zustimmungsbedürftige Handlungen im Sinne von § 69 c UrhG vornehmen will, bekommt er diese mit Abschluss der GPL eingeräumt. Die charakteristische Leistung, die diesen Vertrag gem. Art. 28 Abs. 2 EGBGB kennzeichnet, liegt in der Verfügung über die Nutzungsrechte,[335] so dass es für die Bestimmung des anwendbaren Rechts grundsätzlich auf den gewöhnlichen Aufenthalt der verfügenden Vertragspartei ankommt.

Verfügende Vertragspartei ist gem. Ziff. 6 GPL der Urheber des Programms, weshalb sich das anwendbare Recht maßgeblich nach dessen gewöhnlichem Aufenthalt richtet.

aa)Ein Urheber

Sofern nur ein Urheber an einem Programm besteht, kommt es zu einem Vertrag zwischen diesem und dem Nutzer. Die charakteristische Leistung dieses Vertrags liegt in der Einräumung der Nutzungsrechte und wird vom Urheber erbracht, so dass sie an dessen gewöhnlichem Aufenthaltsort zu lokalisieren ist.

Eine Ausnahme wird hiervon in der Literatur teilweise für den Fall vertreten, in dem das Open Source Programm neben zusätzlichen Leistungen erworben wurde.[336] In diesem Fall sollen die Schenkung der Programmkopie, die Nutzungsrechtseinräumung und die Erbringung der Zusatzleistungen einen einheitlichen gemischten Vertrag darstellen, dessen charakteristische Leistung nicht vom Urheber, sondern von demjenigen erbracht wird, der auch die weiteren Leistungen schuldet.[337] Begründet wird dies damit, dass dem Vertrag in solchen Konstellationen, nicht nur die Pflicht zur Überlassung der Programmkopie nebst weiteren Leistungen, sondern auch die schuldrechtliche Pflicht zur Einräumung der urheberrechtlichen Nutzungsrechte zu entnehmen sei.[338]

---

[335] Vgl. *Jaeger / Metzger*, Open Source Software, Rn. 364; *Deike*, CR 2003, 9, 12; *Schiffner*, Open Source Software, S. 179; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 142; allgemein zur charakteristischen Leistung bei Urheberrechtsverträgen *Katzenberger* in: Schricker, Vor §§ 120 Rn. 156 ff, sowie *Dreier* in: Dreier / Schulze, Vor §§ 120 Rn. 52, jeweils m.w.N.

[336] So bei *Spindler*, Rechtsfragen der Open Source Software, S. 86 f, www.vsi.de/inhalte/aktuell/studie_final_safe.pdf.

[337] Vgl. *Spindler*, Rechtsfragen bei Open Source, D. Rn. 44.

[338] *Spindler*, Rechtsfragen bei Open Source, D. Rn. 44.

Dass die urheberrechtlichen Nutzungsrechte dabei nicht vom Schuldner, sondern nur vom Urheber dinglich wirksam eingeräumt werden können, soll sich nach dieser Auffassung nicht auswirken,[339] da die schuldrechtliche Verpflichtung unabhängig von der tatsächlichen dinglichen Erfüllbarkeit eingegangen werden kann. Für die kollisionsrechtliche Beurteilung käme es einzig auf die schuldrechtliche Verpflichtung an, so dass diese auch für die Bestimmung der charakteristischen Leistung maßgeblich sei.[340]

Hiergegen spricht jedoch bereits eine objektive Betrachtung der Willenserklärungen, aufgrund derer der Vertrag zwischen dem Nutzer und dem Leistungsverpflichteten zustande gekommen ist. Die Willenserklärung des Leistungsverpflichteten bezieht sich, neben den sonstigen Leistungen, jedenfalls auf die Überlassung der Programmkopie. Aus Sicht eines objektiven Erklärungsempfängers kann aber in dieser Willenserklärung, sofern sie sich auf die Verschaffung einer GPL-lizenzierten Programmkopie bezieht, kein Angebot auf Verschaffung umfassender Vertriebs- und Entwicklungsrechte gesehen werden.[341] Dies folgt daraus, dass auch aus Sicht des Nutzers derjenige, der die Programmkopie vertreibt, wegen Ziff. 6 GPL nicht zur Einräumung weitergehender Nutzungsrechte befugt ist. Zwar bezieht sich Ziff. 6 GPL grundsätzlich nur auf die dingliche Nutzungsrechtseinräumung, jedoch wird ein objektiver Erklärungsempfänger davon ausgehen, dass sich auch der schuldrechtliche Verpflichtungswille entsprechend den dinglichen Befugnissen verhalten soll.[342] Ohne weitere Hinweise wird sich daher aus der Willenserklärung kein Angebot entnehmen lassen, in dem der Vertragspartner sich zur Nutzungsrechtseinräumung verpflichten will.[343]

Daher bleibt es auch in solchen Fallgestaltungen dabei, dass die charakteristische Leistung vom Urheber erbracht wird, unabhängig davon, auf welche Weise sich der Erwerb der Programmkopie vollzogen hat.

Der gewöhnliche Aufenthalt des Urhebers ist somit für die Bestimmung der engsten Verbindung maßgeblich.[344] Die GPL richtet sich folglich nur dann nach Deutschem Recht, wenn der Urheber seinen gewöhnlichen Aufenthalt in Deutschland hat.

[339] So ausdrücklich *Spindler*, Rechtsfragen bei Open Source, D. Rn. 44. .
[340] Vgl. *Spindler*, Rechtsfragen bei Open Source, D. Rn. 44.
[341] So zutreffend *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 776.
[342] Vgl. *Deike*, CR 2003, 9, 11.
[343] Im Ergebnis ebenso *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 776.
[344] Vgl. *Deike*, CR 2003, 9, 12; *Schiffner*, Open Source Software, S. 179; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 142 f.

bb)Schöpferische Bearbeitung durch einen Entwickler

Sofern ein urheberrechtlich geschütztes Programm von einem einzelnen Entwickler in schöpferischer Weise bearbeitet wurde, stellt sich diese Fallkonstellation in der rechtlichen Konsequenz ähnlich dar.

Der Nutzer, der auf die weitergehenden Nutzungsrechte an der bearbeiteten Programmversion angewiesen ist, erhält diese vom Bearbeiter durch den Abschluss der GPL eingeräumt. Diesem Ergebnis scheint zwar zunächst Ziff. 6 GPL zu widersprechen, indem hierin geregelt ist, dass die Rechtseinräumung stets durch die Rechtsinhaber des ursprünglichen Programms erfolgen soll.[345] Der Vertragsschluss zwischen Bearbeiter und Nutzer folgt aber letztlich aus § 23 UrhG, wonach dem Bearbeiter ein selbstständiges Urheberrecht an der Bearbeitung zusteht.[346] Die für die Verwertung und Veröffentlichung erforderliche Einwilligung der Urheber der vorangegangenen Programmversionen erhält der Bearbeiter seinerseits durch das in der GPL enthaltene Angebot auf Einräumung des Nutzungsrechts.[347]

Es kommt daher ausschließlich zu einem Vertrag zwischen dem Nutzer und dem Bearbeiter, so dass es für die Frage nach dem anwendbaren Recht auf dessen gewöhnlichen Aufenthaltsort ankommt. Deutsches Recht findet somit auch in diesen Fallgestaltungen nur dann Anwendung, wenn der verfügende Bearbeiter seinen gewöhnlichen Aufenthalt in Deutschland hat.

cc)Mehrere Urheber

Im Gegensatz zu den vorangegangenen Fallgruppen sind die Entwicklungsstrukturen im Open Source Bereich regelmäßig dadurch gekennzeichnet, dass sich nicht nur einzelne Urheber an den Entwicklungen beteiligen. Vielmehr werden Open Source Programme häufig über Jahre hinweg sukzessiv entwickelt. Dabei kommt es mitunter zu zahlreichen Versionierungen des Programms, abhängig davon, ob es sich bei dem aktuellen Zwischenstand um eine End-,

[345] Insofern wird zum Teil auch davon ausgegangen, dass es sowohl zu einem Vertragsschluss zwischen dem Bearbeiter und dem Nutzer sowie daneben zu zahlreichen weiteren Vertragsschlüssen mit den jeweiligen Urhebern kommt. Die Rechtseinräumung würde demnach immer nur vom jeweiligen Urheber in einem gesonderten Vertrag erfolgen, vgl. *Spindler*, Rechtsfragen der Open Source Software, S. 33, www.vsi.de/inhalte/aktuell/studie_final_safe.pdf; *Deike*, CR 2003, 9, 16.

[346] Vgl. auch *Schiffner*, Open Source Software, S. 182; daran anknüpfend auch *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 310.

[347] Vgl. *Schiffner*, Open Source Software, S. 182 Fn. 691; ebenso *Lenhard*, Vertragstypologie von Softwareüberlassungsverträgen, S. 310.

Beta- oder auch nur Deltaversion handelt. Der Entwicklungsform nach, ist der Open Source Bereich oftmals durch gemeinsame Entwicklungsbemühungen gekennzeichnet, die einem gemeinsamen Ziel dienen und somit als Miturheberschaft im Sinne von § 8 UrhG zu beurteilen sind.[348]

Daneben kommt im Rahmen der Entwicklung von Open Source Software aber auch dem Bearbeiterurheberrecht besondere Bedeutung zu.[349] Allerdings findet auch die schöpferische Weiterentwicklung einer Programmversion in der Regel wiederum durch gemeinsame Bemühungen mehrerer Entwickler statt,[350] weshalb die daraus folgenden Bearbeiterurheberrechte nicht einer Person allein, sondern mehreren Personen in Miturheberschaft zustehen.

Aus vertragsrechtlicher Sicht folgt daraus für den Nutzer, dass er die durch die GPL vermittelten Nutzungsrechte zumeist in einem Vertrag eingeräumt bekommt, an dem, aufgrund der aus § 8 Abs. 2 UrhG folgenden gesamthänderischen Bindung der Verwertungsrechte, mehrere Urheber beteiligt sind.

Im Hinblick auf die Regelung in Art. 28 Abs. 2 EGBGB stellt sich somit die Frage, wie die charakteristische Leistung, die auch hier in der Einräumung der Nutzungsrechte besteht, zu lokalisieren ist.

(1)Gesellschaftlicher Zusammenschluss zwischen den Urhebern

Eindeutig kann die charakteristische Leistung zunächst in solchen Fallgestaltungen lokalisiert werden, in denen sich die beteiligten Miturheber zu einer höheren Organisationsform zusammengeschlossen haben. Aufgrund der steigenden wirtschaftlichen Bedeutung von Open Source Software lassen sich in der Praxis zunehmend Fälle finden, in denen die Gesamtentwicklung eines Programms in den Händen eines Unternehmens liegt.

Dies kommt zunächst einmal in denjenigen Fällen vor, in denen sich Unternehmen dazu entschließen ihre ehemals proprietären Entwicklungen unter einer Open Source Lizenz zu veröffentlichen.[351] Zudem sind in der Praxis gerade in jüngerer Zeit Bestrebungen einzelner Unternehmen zu erkennen, die Verwertungsrechte trotz eines öffentlichen

[348] Einzelheiten hierzu siehe bereits Kapitel 2 S. 25 ff.
[349] Vgl. Kapitel 2 S. 36 ff.
[350] Ähnlich *Küng*, MR 2004, 21, 26.
[351] Bekanntestes Beispiel dürfte der Browser Netscape sein, dessen Quellcode 1998 veröffentlicht wurde, vgl. http://www.mozilla.org/mission.html. Ein aktuelleres Beispiel stellt die Firma Sun Microsystems dar, die damit begonnen hat, die gesamte Java Plattform unter der GPL zu lizenzieren, vgl. http://www.sun.com/software/opensource/java/faq.jsp#a.

Entwicklungsmodells dauerhaft in der eigenen Hand zu behalten. Aus diesem Grund werden Open Source Entwicklungen zum Teil erst nach deren unternehmensinterner Vollendung dem Open Source Bereich zugeführt,[352] oder aber auch die beteiligten „unternehmensfremden" Entwickler dazu verpflichtet, die Unternehmen an ihren Verwertungsrechten zu beteiligen.[353]
Im Hinblick auf die Frage nach dem anwendbaren Recht folgt aus dem Umstand, dass sämtliche Verwertungsrechte einem Unternehmen zustehen, dass für die Bestimmung des anwendbaren Rechts gem. Art. 28 Abs. 2 EGBGB auf dessen Hauptverwaltungssitz abzustellen ist. Insofern ist sowohl die Bestimmung der charakteristischen Leistung des Vertrags, als auch deren Lokalisierung unproblematisch möglich.
Deutsches Recht findet auf einen solchen Vertrag nur dann Anwendung, wenn das Unternehmen seinen Hauptverwaltungssitz in Deutschland hat.

(2)Fehlen von ausdrücklichen Verwertungsregelungen

Probleme mit der Lokalisierung der charakteristischen Leistung könnten allerdings dann bestehen, wenn es zwischen den Miturhebern zu keinen Absprachen gekommen ist, die über die in der GPL enthaltenen Regelungen hinausgehen. Zwar liegt auch in diesen Fällen, aufgrund von § 8 Abs. 2 UrhG eine gesamthänderische Bindung zwischen den Miturhebern vor, zweifelhaft ist aber bereits, ob sich diese als höhere Organisationsform qualifizieren lässt[354] und ob eine solche gesellschaftsrechtliche Einordnung – sofern sie denn angemessen wäre – überhaupt für die Lokalisierung der charakteristische Leistung aussagekräftig ist.
Zweifel an der Aussagekraft bestehen vor allem vor dem Hintergrund, dass es zwischen den Open Source Entwicklern typischerweise sowohl an Regelungen zur Vertretung der Gesellschaft als auch zur Bildung von Gesellschaftsvermögen fehlt.[355]
Weiterhin zeichnet sich die Mitarbeit an Open Source Entwicklungen gerade durch deren Freiwilligkeit aus, weshalb es zumeist weder Mitwirkungspflichten noch sonstige zwingende Aufgabenzuweisungen zwischen den Urhebern gibt. Auch lassen sich mangels tatsächlicher Gesellschaftsgründung keine Tatsachen erkennen, die auf einen Unternehmenssitz oder

[352] Solche Entwicklungspraxen ließen sich jüngst bei Novell und Red Hat beobachten, die hinsichtlich Gnome Oberflächen zunächst ein völlige Eigenentwicklung vollzogen haben und erst danach diese zur Weiterbearbeitung freigegeben haben, vgl. http://www.heise.de/newsticker/meldung/69447Kde.
[353] In diese Richtung geht beispielsweise das Copyright Assignment von Sun, http://www.openoffice.org/licenses/jca.pdf, das ein Joint ownership vorsieht.
[354] Vgl. *Sester*, CR 2000, 797, 801, der jedoch ausdrücklich darauf hinweist, dass er aufgrund der bestehenden Unterschiede nur von einer Nähe zum Paradigma der BGB-Gesellschaft ausgeht.
[355] *Sester*, CR 2000, 797, 801; *Weber* in: Festschrift Honsell, S. 55.

Gründungsort hinweisen, weshalb eine Bestimmung des Gesellschafts- bzw. Verwaltungssitzes für solche Organisationsformen weder nach der „Sitz-“ noch nach der „Gründungstheorie“ möglich erscheint.[356]

Die auftretenden Schwierigkeiten ähneln denen, die in der Literatur unlängst im Zusammenhang mit der Sitzbestimmung virtueller Unternehmen ohne eigenen Verwaltungssitz und ohne leitende Partei erörtert wurden.[357] Für virtuelle Unternehmen wird dabei vorgeschlagen, dass sich deren Hauptsitz nach der engsten Verbindung des Gesellschaftsvertrages richten soll.[358] Gerade hieran fehlt es aber häufig im Rahmen der lockeren Zusammenschlüsse zwischen Entwicklern im Open Source Bereich, weshalb sich die Vorschläge zu virtuellen Unternehmen kaum auf den Open Source Bereich übertragen lassen.

Im Übrigen wäre selbst bei der Annahme, dass ein Gesellschaftsvertrag auch im Rahmen solcher lockeren Entwicklerverbünde konkludent abgeschlossen würde, nur wenig im Zusammenhang mit der Frage nach dem anwendbaren Recht hinzugewonnen. Letztlich käme es – mangels Rechtswahl im Sinne von Art. 27 EGBGB – auch im Rahmen der Begutachtung des Gesellschaftsvertrags wiederum auf Art. 28 EGBGB an, ohne dass weitere Umstände vorhanden sind, an die für die Bestimmung des anwendbaren Rechts angeknüpft werden könnte.

Aufgrund dieser Schwierigkeiten bemüht sich die Literatur seit geraumer Zeit um neue Kriterien, an die im Rahmen des Art. 28 Abs. 2 EGBGB für die Lokalisierung der charakteristischen Leistung angeknüpft werden könnte.

(3)Serverstandort

Der Serverstandort, von dem der Nutzer die Programmkopie heruntergeladen hat, könnte dabei zunächst aussagekräftig sein. Richtigerweise kann dem Serverstandort jedoch kein Gewicht beigemessen werden, da er nahezu beliebig verlagert werden kann und zudem allzu oft allein aus finanziellen Erwägungen gewählt wird.[359] Insofern entspricht es der allgemeinen

[356] Vgl. *Spindler*, Rechtsfragen bei Open Source, C. Rn. 138; allgemein zur Bestimmung des anwendbaren Rechts bei Gesellschaften ohne eigene Organisation *Grossfeld* in: Staudinger, Int. GesR Rn. 772 f, der zur Anwendung des Vertragsstatuts gelangt; *Heldrich* in: Palandt, Art. 28 EGBGB Rn. 19.
[357] Eingehend *Lange*, Virtuelle Unternehmen, Rn. 531 ff.
[358] Vgl. *Lange*, Virtuelle Unternehmen, Rn. 531 ff. und 574.
[359] Da Open Source Projekte zumeist ohne ein eigenes Vermögen mit den Entwicklungen starten, nutzen sie häufig die kostenlosen Angebote von Hosting-Anbietern, die zudem auch die Nutzung von Entwicklungsumgebungen beinhalten; vgl. http://www.sourceforge.net, http://www.berlios.de, http://www.freshmeat.net und http://www.javaforge.com.

Ansicht,[360] dass der Serverstandort für die Bestimmung des Hauptverwaltungssitzes nicht maßgeblich sein kann.

(4)Aufenthaltsort des Serverbetreibers

Um der Gefahr entgegenzuwirken, dass sich eine Verlagerung des Serverstandorts auf den Hauptverwaltungssitz auswirkt, wird in der Literatur teilweise vorgeschlagen, anstatt dessen für Lokalisierung der charakteristischen Leistung auf den Aufenthaltsort des Serverbetreibers abzustellen.[361]

Hiergegen spricht jedoch, dass dieser oftmals weder eine Verbindung zu den Urhebern noch zu dem Projekt aufweist. Insofern stellt sich offensichtlich die Frage, warum es auf dessen gewöhnlichen Aufenthalt ankommen soll, unabhängig davon, welche Rolle ihm ansonsten im Rahmen der Entwicklungen zukommt.

Gegen eine solche Anknüpfung ist auch anzuführen, dass der Serverbetreiber oftmals lediglich Dienstleister ist, der ohne Kenntnis von den einzelnen Entwicklungen des Projekts einzig den Speicherplatz und die Entwicklungsumgebung (kostenlos) zur Verfügung stellt.[362]

Es kann daher nicht überzeugen, dem Sitz des Serverbetreibers gegenüber den Urhebern ein solch hohes Gewicht einzuräumen. Im Übrigen versagt das Kriterium des Aufenthaltsorts auch in den Fällen, in denen die Software nicht zentral in einem Repository gespeichert ist, sondern dezentral verwaltet und entwickelt wird. Diese Möglichkeit wird aufgrund neuer Entwicklungsplattformen zunehmend von Projekten genutzt,[363] wodurch letztlich mehrere Serverbetreiber vorhanden sind, auf deren Aufenthaltsorte abgestellt werden könnte.

Hieraus wird ersichtlich, dass das Kriterium des Aufenthaltsortes des Serverbetreibers, ebenso wie das des Serverstandorts, für die Lokalisierung der charakteristischen Leistung ungeeignet ist.

---

[360] *Mankowski*, RabelsZ 1999, 203, 226 ff; *Waldenberger*, BB 1996, 2365, 2371; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 138.

[361] *Spindler*, Rechtsfragen bei Open Source, C. Rn. 138.

[362] Vgl. hierzu beispielhaft die Entwicklungsportale von http://www.sourceforge.net, http://www.berlios.de, http://www.freshmeat.net und http://www.javaforge.com.

[363] Als Beispiele für solche Entwicklungsprogramme seien hier nur Git, Arch, Darcs und das kommerzielle Bitkeeper genannt, bei denen der Source Code dezentral entwickelt wird und der Source Code mithin auf viele Server verteilt sein kann.

(5)Art der Projektbeteiligung

Statt dessen könnte für die Lokalisierung der charakteristischen Leistung auf den gewöhnlichen Aufenthalt desjenigen abzustellen sein, der das Projekt leitet bzw. den größten Einfluss auf die Projektentwicklung hat. So ließe sich der Hauptverwaltungssitz z.B. an dem gewöhnlichen Aufenthaltsort des Maintainers oder an eine ähnlich einflussreiche Projektstellung knüpfen.

Gegen eine solche Vorgehensweise sprechen jedoch gewichtige Gründe, die ihre Ursache im Wesentlichen in den besonderen Entwicklungsstrukturen von Open Source Projekten haben. So fehlt es Open Source Projekten, aufgrund der freiwilligen Mitarbeit, regelmäßig an zwingenden Aufgabenzuweisungen.

Selbst die Position des „Projektleiters" enthält kein statisches Moment, dass eine Prognose über die künftigen Projektstrukturen zulassen würde. Ein Wandel der Projektmitarbeit, der jederzeit und ohne weiteres vollzogen werden kann, hätte u.U. die Verschiebung der „Projektleitung" und mithin auch die des Gesellschaftssitzes zur Folge. Aufgrund der flexiblen Aufgabenbereiche lässt sich im Übrigen allein aus der Bezeichnung als Maintainer weder ein zwingender Rückschluss auf dessen Einfluss im Rahmen des Projekts noch auf dessen eigene Urheberschaft ziehen.[364] Daher müsste der Gesellschaftssitz in jedem Einzelfall anhand der tatsächlichen Einflussmöglichkeiten bestimmt werden. Außenstehende können allein aufgrund Bezeichnung der Projektstellung keine Rückschlüsse auf den Gesellschaftssitz ziehen, was vor dem Hintergrund der Rechtssicherheit erheblichen Bedenken begegnet.

Durch einen Wandel der Projektmitarbeit könnte sich der Gesellschaftssitz unvorhersehbar ändern, so dass mitunter für ein und dieselbe Version der Software die charakteristische Leistung in verschiedenen Staaten zu lokalisieren wäre und sich als Folge davon nach unterschiedlichen Rechtsordnungen richten würde.

Für den Nutzer bestünde kaum Gewissheit darüber, nach welcher Rechtsordnung sich die GPL richtet. Dies wird zudem dadurch verschlimmert, dass es für die Bestimmung der anwendbaren Rechtsordnung nicht auf den Zeitpunkt des Downloads, sondern auf Zeitpunkt

[364] Die Gruppe der Projektbeteiligten wird nur in Ausnahmefällen mit derjenigen der Urheber übereinstimmen. Dies liegt daran, dass die Entwicklung an einer Open Source Software nicht zugleich auch die Projektmitgliedschaft voraussetzt. Gerade in größeren Projekten beschränkt sich die Projektarbeit häufig im Wesentlichen darauf, die Entwicklung zu koordinieren und auf ein stimmiges Gesamtkonzept auszurichten. Die eigentliche Entwicklungsleistung, an die auch die Urheberschaft geknüpft ist (vgl. hierzu bereits Kapitel 2 S. 17 ff), wird in erheblichem Umfang durch die Einsendung von Patches und ausgestalteten Erweiterungsmöglichkeiten erbracht. Die eigentliche Projektarbeit zeichnet sich dann dadurch aus, dass diese Patches in den Quellcode eingearbeitet und von den Erweiterungsmöglichkeiten solche ausgesucht werden, die stabil laufen und mit den Entwicklungszielen übereinstimmen. Hierdurch wird jedoch keine Urheberschaft begründet, so dass die Gruppe der Miturheber wohl nur in sehr kleinen Projekten tatsächlich mit derjenigen der Projektbeteiligten übereinstimmt; diese Arbeitsteilung wird von *Jaeger / Metzger*, Open Source Software, Rn. 196, als klassich für Community-Projekte bezeichnet.

ankommt, zu dem die GPL abgeschlossen wird. Dieser erfolgt aber häufig erst dann, wenn der Nutzer weitergehende Nutzungsrechte benötigt. Selbst wenn der Nutzer folglich Kenntnis von der Projektstruktur zum Zeitpunkt des Downloads hatte, besagt dies noch nichts für Lokalisierung der charakteristischen Leistung des zeitlich nachfolgenden Abschlusses der GPL.
Insgesamt ist das Kriterium der Projektmitarbeit folglich nicht dazu geeignet, eine zwingende Lokalisierung der charakteristischen Leistung zu rechtfertigen.

c.Zwischenergebnis zu Art. 28 Abs. 2 EGBGB

Insofern ist festzuhalten, dass eine eindeutige Lokalisierung der charakteristischen Leistung bei Open Source Projekten häufig selbst dann nicht möglich sein wird, wenn man, aufgrund der von § 8 Abs. 2 UrhG vermittelten gesamthänderischen Bindung zwischen den Miturhebern, von einer höheren Organisationsform ausgehen wollte. Bei lockeren Bündnissen zwischen Miturhebern fehlt es im Open Source Bereich letztlich an tatsächlichen Anknüpfungspunkten, die eine eindeutige Sitzbestimmung der Hauptverwaltung zulassen würden.

Vor diesem Hintergrund erscheint es zweifelhaft, ob das anwendbare Recht bei den im Open Source Bereich vorherrschenden dezentralisierten Entwicklungsformen tatsächlich anhand von Art. 28 Abs. 2 EGBGB bestimmt werden kann. Immerhin lässt sich der mit Art. 28 Abs. 2 EGBGB verfolgte Zweck einer größeren Einfachheit, Klarheit und Voraussehbarkeit der Ergebnisse,[365] im Open Source Bereich nicht verwirklichen.
In der Literatur wird in vergleichbaren Sachverhaltskonstellationen zum Teil von einer Anwendung des Art. 28 Abs. 2 EGBGB abgesehen, wenn hierdurch zwar die charakteristische Leistung eindeutig bestimmt, sich letztlich aber nicht lokalisieren lässt.[366] Diese Voraussetzungen sollen beispielsweise vorliegen, wenn mehrere Schuldner der charakteristischen Leistung vorhanden sind und eine strikte Anwendung des Art. 28 Abs. 2 EGBGB zu einer Aufspaltung des Vertrages führen würde.[367] Zur Begründung der Nichtanwendbarkeit von Art. 28 Abs. 2 S. 1 EGBGB wird dabei auf die Regelung in Art. 28

[365] *Kropholler*, Internationales Privatrecht, § 52 III, S. 461.
[366] *Martiny* in: Reithmann / Martiny, Rn. 138; *Martiny* in: Müko, Art. 28 EGBGB Rn. 60, *Dörner*, Anm. zu BGH JR 1987, 201.
[367] *Martiny* in: Müko, Art. 28 EGBGB Rn. 60, *Dörner*, Anm. zu BGH JR 1987, 201; *Stoll* in: FS Müller-Freienfels, S. 646; *Geisler*, Die engste Verbindung im internationalen Privatrecht, S. 200 f.

Abs. 2 S. 3 EGBGB zurückgegriffen und deren Anwendungsbereich auch auf solche Fälle erweitert, in denen keine *einheitliche* charakteristische Leistung auszumachen ist.[368]

Von der rechtlichen Ausgangslage her ähnlich sind auch die hier untersuchten Open Source Projekte, bei denen die charakteristische Leistung zwar ebenfalls eindeutig zu ermitteln, jedoch mangels tatsächlicher Anknüpfungspunkte nicht einheitlich zu lokalisieren ist. Eine Anwendung von Art. 28 Abs. 2 S. 1 EGBGB ist daher abzulehnen.[369]

Im Ergebnis findet somit die in Art. 28 Abs. 2 S. 1 EGBGB enthaltene Vermutungsregelung keine Anwendung, weshalb für die Frage nach dem anwendbaren Recht auf die Generalklausel in Art. 28 Abs. 1 EGBGB zurückzugreifen ist.[370]

4.Engste Verbindung der GPL, Art. 28 Abs. 1 EGBGB

Nach Art. 28 Abs. 1 EGBGB ist für die Bestimmung des anwendbaren Rechts maßgeblich darauf abzustellen, zu welchem Staat die GPL, die zwischen den Miturhebern und dem Nutzer abgeschlossen wird, die engste Verbindung aufweist.

Die Kriterien, die für die Beurteilung der engsten Verbindung dabei maßgeblich sind, lassen sich dem Gesetz nicht entnehmen. Augenscheinlich haben aber die Eigenheiten des Vertrages und die Verhältnisse der Parteien zueinander Auswirkungen auf die Beurteilung.[371]

Daher sind, wie bereits nach altem Recht, die Interessen der Parteien zu ermitteln und gegeneinander abzuwägen.[372] Hierbei kommt jedoch nur solchen Interessen eine Bedeutung zu, die international-privatrechtlicher Natur sind und sich mithin auf die Anwendung einer bestimmten Rechtsordnung beziehen.[373] Im Gegensatz dazu bleibt es unberücksichtigt, wenn eine der Vertragsparteien ein besonderes Interesse an der Anwendung eines bestimmten materiellen Rechts hat.

Daneben lassen sich der Rechtsprechung weitere Indizien entnehmen, die für die Bestimmung der engsten Verbindung bedeutsam sind. Besonderes Gewicht wird dabei Gerichtsstands- oder Schiedsklauseln sowie der Vereinbarung eines gemeinsamen Erfüllungsortes, beigemessen.[374]

---

[368] So wohl *Martiny* in: Müko, Art. 28 EGBGB Rn. 60; *Dörner*, Anm. zu BGH JR 1987, 201.

[369] Ob dafür eine entsprechenden Anwendung von Art. 28 Abs. 2 S. 3 EGBGB erforderlich ist, oder ob solche Fallgestaltungen bereits vom Wortlaut mit umfasst werden – wie sich dies wohl mit Hinweis auf das weite Verständnis in der Literatur vertreten ließe – kann hier letztlich dahinstehen, da auch die Analogievoraussetzungen aufgrund der vergleichbaren Umstände gegeben wären.

[370] *Magnus* in: Staudinger, Art. 28 EGBGB Rn. 32.

[371] *Ferid*, Internationales Privatrecht, Rn. 6-52.

[372] *Hohloch* in: Erman, Art. 28 EGBGB Rz. 1.

[373] *Martiny* in: Müko, Art. 28 EGBGB Rn. 11.

[374] Vgl. *Martiny* in: Müko, Art. 28 EGBGB Rn. 89 ff m.w.N.

Den weiteren Umständen, wie Abschlussort, Vertragssprache und -währung sowie gemeinsamer Staatsangehörigkeit der Vertragsparteien, wurde von der Rechtsprechung lediglich minder großes Gewicht zuerkannt.[375] Für die Begründung der engsten Verbindung soll das Vorliegen eines solchen Umstands nur ausreichen, wenn dieser durch weitere Tatsachen bestätigt wurde.[376]

Deuten mehrere Umstände – sich widersprechend – auf verschiedene Rechtsordnungen hin, so müssen diese einander gegenübergestellt und anhand ihrer Bedeutung gegeneinander abgewogen werden.[377]

Unter Zugrundelegung vorgenannter Kriterien wird ersichtlich, dass diese nur sehr begrenzt für die Beurteilung der GPL verwertbar sind. Die GPL enthält weder eine ausdrückliche Rechtswahl,[378] noch Gerichtsstands- oder Schiedsklauseln.

Der Erfüllungsort, an dem die Nutzungsrechte eingeräumt werden, liegt zwar regelmäßig beim Lizenznehmer. Allerdings ist dies nicht auf eine Vereinbarung zurückzuführen, sondern darauf, dass sich das der GPL innewohnende Angebot auf Übertragung der Nutzungsrechte an dem Programm auf deren Einräumung am Sitz des Lizenznehmers bezieht. Insofern lässt sich anhand des Erfüllungsorts keine engere Verbindung zum Staat des Lizenznehmers begründen.

Ebenso verhält es sich mit dem Abschlussort. Dieser wird zwar typischerweise beim Nutzer liegen, was jedoch nicht an einer Vereinbarung, sondern daran liegt, dass der Lizenznehmer das in der GPL enthaltene Angebot zum Abschluss eines Lizenzvertrages erst in dem Zeitpunkt konkludent annimmt, in dem er die Software verändern oder verbreiten will.[379] Diese Annahme wird in aller Regel am Sitz des Unternehmens erfolgen, weshalb hier auch der Abschlussort zu lokalisieren ist.

Der in englischer Sprache verfasste Vertragstext könnte im Zusammenhang mit der Entstehungsgeschichte der GPL auf eine enge Verbindung zum US-Recht hinweisen.[380] Hiergegen spricht indes, dass es sich bei der GPL um einen Mustervertrag handelt, der möglichst weltweit zur Anwendung kommen sollte. Insofern wurde Englisch nur deshalb als Vertragssprache gewählt, um einem möglichst großen Nutzerkreis die Verwendung zu

[375] Zu den einzelnen Kriterien vgl. die eingehende Darstellung bei *Martiny* in: Reithmann / Martiny, Rn. 144 ff; sowie *Martiny* in: Müko, Art. 28 EGBGB Rn. 89 ff, jeweils m.w.N. aus der Rechtsprechung.
[376] *Martiny* in: Reithmann / Martiny, Rn. 143.
[377] *Martiny* in: Reithmann / Martiny, Rn. 143.
[378] Vgl. bereits S. 72 f.
[379] Vgl. Kapitel 3 S. 42 ff.
[380] Vgl. *Metzger / Jaeger*, GRUR Int. 1999, 839, 842 Fn. 49; neuerdings a.A. *Jaeger / Metzger*, Open Source Software, Rn. 362.

ermöglichen, ohne dass daraus jedoch auch zwangsläufig eine Bindung zum US-Recht beabsichtigt war.[381]

Als weiteres Kriterium käme die Staatsangehörigkeit der Vertragsparteien in Betracht. Aufgrund der internationalen Beteiligung an Open Source Entwicklungen und der weltweiten Verbreitung der Software wird die GPL regelmäßig mit Miturhebern aus verschiedenen Ländern geschlossen. Sofern jedoch keine einheitliche Staatsangehörigkeit aller Miturheber gegeben ist,[382] lässt sich aus der Staatsangehörigkeit kein Indiz für eine engste Verbindung zu einem bestimmten Staat entnehmen.

Zwischenergebnis

Festzuhalten bleibt, dass sich aus den bislang erörterten Kriterien keine eindeutigen Schlüsse auf eine engste Verbindung zu einem bestimmten Staat ziehen lassen. Während Erfüllungs- und Abschlussort eher auf den Ort der Niederlassung des Lizenznehmers hindeuten könnten, sprechen hiergegen die Vertragssprache und der Umstand, dass die charakteristische Leistung – wenn sie auch sonst nicht einheitlich zu lokalisieren ist – häufig nicht aus dem Staat kommt, in dem der Lizenznehmer seine Niederlassung hat.

5.Interessenabwägung

Für die engste Verbindung kommt es daher auf eine Abwägung der Parteiinteressen an, wobei seit der Reform nicht mehr auf den hypothetischen Parteiwillen,[383] sondern auf die objektiv zu gewichtenden gegenseitigen kollisionsrechtlichen Interessen abzustellen ist.[384] Dies führt typischerweise dazu, dass jede Vertragspartei ein besonderes Interesse an der Anwendung der eigenen Rechtsordnung haben wird.

Im Open Source Bereich könnten hieran indes Zweifel bestehen, die aus den Besonderheiten herrühren, durch die das Vertragsverhältnis zwischen Nutzern und Lizenzgebern geprägt ist.

Eine dieser Besonderheiten liegt darin, dass für den Nutzer regelmäßig weder der Vertragspartner noch dessen gewöhnlicher Aufenthalt erkennbar sind. Dies liegt daran, dass

[381] Vgl. oben S. 72 f.
[382] Aufgrund der dezentralen Entwicklung von Open Source Software ist dies in der Praxis eher selten, weshalb solche Besonderheiten vorliegend unberücksichtigt bleiben.
[383] *Martiny* in: Reithmann / Martiny, Rn. 155; *Hohloch* in: Erman Art. 27 EGBGB Rn. 11.
[384] *Kegel / Schurig*, Internationales Privatrecht, § 18 I S. 658 ff; *Martiny* in: Müko, Art. 28 EGBGB Rn. 12.

die Miturheber nicht auch zugleich Projektmitglieder sein müssen[385] und sich deren Urheberschaft somit für den Nutzer nicht ermitteln lässt. Selbst wenn sämtliche Urheber anhand einer vollständigen Protokollierung des Versionsverwaltungssystems namentlich benannt wären, würde dies hieran nichts ändern. In einem Versionsverwaltungssystem werden typischerweise nur der sog. Realname[386] und eine E-Mail-Adresse der Entwickler protokolliert, die jedoch weder zwingend Rückschlüsse auf deren tatsächliche Identität noch Herkunft zulassen. Dies könnte dafür sprechen, dass dem Nutzer ein besonders schutzwürdiges Interesse an der Geltung der eigenen Rechtsordnung zuzuerkennen ist. Bei der Bestimmung der engsten Verbindung würde als Folge davon, ausnahmsweise nicht auf den leistenden Vertragsteil, sondern auf den Leistungsempfänger abgestellt werden.[387]

Eine vergleichbare Ausnahme wird in der Literatur teilweise im Rahmen von Art. 28 Abs. 5 EGBGB für solche Fälle vertreten, in denen der gewöhnliche Aufenthalt der leistenden Partei bei Vertragsabschluss nicht erkennbar ist.[388] Liegen diese Voraussetzungen vor, soll dem Leistungsempfänger ein besonders schutzwürdiges Interesse an der Geltung der eigenen Rechtsordnung zustehen, sofern dem Vertragsschluss nicht offensichtlich ein unseriöses Geschäft zugrunde lag.[389]
Bei der GPL, die zwischen Nutzern und Urhebern zustande kommt, folgt die Unkenntnis vom Vertragspartner nicht daraus, dass es sich um ein unseriöses Geschäft gehandelt hat. Vielmehr folgt sie aus den Besonderheiten, durch die das Open Source Umfeld im Entwicklung- und Vertriebsbereich gekennzeichnet ist. Daher ist es gerechtfertigt, den Nutzern auch hier ein besonders schutzwürdiges Interesse an der Geltung der eigenen Rechtsordnung zuzugestehen.
Dies gilt insbesondere auch vor dem Hintergrund, dass der Nutzer nur selten die Möglichkeit haben wird, mit sämtlichen Urhebern in direkten Kontakt zu treten, um eine bestimmte Rechtswahl zu vereinbaren.[390] Unter Zugrundelegung dieser Umstände könnte eine engere Verbindung zum gewöhnlichen Aufenthalt des Nutzers bestehen.

---

[385] Zu den Entwicklungsstrukturen in Open Source Projekten und den Folgen für die Urheberschaft der jeweiligen Beteiligten, vgl. Kapitel 2 S. 17.
[386] Hierbei handelt es sich jedoch zumeist nicht um den tatsächlichen Namen der Entwickler. In der Praxis verwenden die Entwickler vielmehr zumeist Pseudonyme.
[387] Eine Anknüpfung am gewöhnlichen Aufenthalt des Leistungsempfängers wird für Lizenzverträge grundsätzlich nur dann vertreten, wenn diesem ein ausschließliches Nutzungsrecht eingeräumt wurde bzw. ihn eine Verwertungspflicht trifft, vgl. *Hoffmann*, RabelsZ 1976, 208, 214; *Schack*, Urheber- und Urhebervertragsrecht, Rn. 1144; *Schack* in: FS Heldrich, S. 998 f; a.A. *Hiestand* in: Reithmann / Martiny, Rn. 1737.
[388] *Spickhoff* in: Bamberger / Roth, Art. 28 EGBGB Rn. 24.
[389] *Spickhoff* in: Bamberger / Roth, Art. 28 EGBGB Rn. 24; vgl. auch *Mankowski*, RabelsZ 1999, 203, 224 ff.
[390] Hierauf weist zu Recht *Spindler*, Rechtsfragen bei Open Source, C. Rn. 144, hin.

Hierfür ließe sich weiterhin auch ein mangelndes kollisionsrechtliches Interesse auf Seiten der Urheber anführen. Zweifel am Vorliegen eines schutzwürdigen eigenen Interesses ergeben sich dabei aus dem Umstand, dass die Urheber eines Programms oftmals weltweit verstreut sitzen und regelmäßig voneinander nur insoweit Kenntnis haben, als dass der Realname und die E-Mail-Adresse der Mitentwickler – nicht aber deren gewöhnlicher Aufenthalt – bekannt sind.[391] Selbst wenn aber sämtliche Miturheber Kenntnis voneinander, sowie vom gewöhnlichen Aufenthalt der jeweils anderen hätten, ist allein diese Kenntnis nicht dazu geeignet, ein schutzwürdiges Interesse an der Anwendung einer bestimmten Rechtsordnung zu rechtfertigen.[392]

Für ein schutzwürdiges kollisionsrechtliches Interesse der Urheber bedürfte es, aufgrund der lockeren Bündnisse zwischen den Entwicklern, vielmehr einer ausdrücklichen Regelung über das anzuwendende Recht. Fehlt es hieran, sind auf Seiten der Urheber keine Umstände ersichtlich, aufgrund derer diese darauf Vertrauen können, dass eine bestimmten Rechtsordnung zur Anwendung gelangt.

Infolgedessen wird im Rahmen einer objektiven Gewichtung der gegenseitigen kollisionsrechtlichen Interessen ersichtlich, dass die Miturheber kein schutzwürdiges Interesse an der Anwendung einer – ihren Interessen entsprechenden – Rechtsordnung geltend machen können.

Ergebnis:

Eine objektive Gewichtung der gegenseitigen kollisionsrechtlichen Interessen führt somit dazu, dass den Nutzern, im Gegensatz zu den Urhebern, ein stärkeres Interesse an der Geltung der eigenen Rechtsordnung zuzuerkennen ist. Die GPL, mit welcher die Nutzungsrechte an Open Source Software erworben werden, ist daher bei gemeinschaftlichen Entwicklungen stets nach der Rechtsordnung zu beurteilen, in welcher der Nutzer seinen gewöhnlichen Aufenthalt oder seine Niederlassung hat.

Im Ergebnis kommt es daher, aufgrund der Besonderheiten im Open Source Bereich ausnahmsweise dazu, dass für die Bestimmung des anwendbaren Rechts nicht die Rechtsordnung der leistenden Vertragspartei maßgeblich ist.

[391] Ähnlich auch *Plaß*, GRUR 2002, 670, 672, die davon ausgeht, dass die Mitentwickler zumeist nicht einmal wechselseitig die Namen kennen würden.

[392] Ausgenommen sind hiervon solche Konstellationen, in denen alle Urheber aus ein und demselben Land kommen und insofern davon ausgehen dürfen, dass ihre eigene Rechtsordnung zur Anwendung gelangt.

Im Gegensatz zu „klassischen“ Vertragsverhältnissen spricht hiergegen auch nicht das sog. Uniformitätsinteresse,[393] welches früher in den Fällen von „Massenverträgen“ herangezogen wurde, um für die Rechtswahl eine Anknüpfung am Ort der leistenden Partei zu rechtfertigen.[394] Begründet wurde dieses Interesse nämlich damit, dass die leistende Partei aufgrund der Vielzahl von Vertragsabschlüssen ein besonderes Interesse an der Gewissheit hätte, dass auf sämtliche Verträge einheitlich ihre eigene Rechtsordnung zur Anwendung kommt.[395] Im Gegensatz zu der Konstellation von solchen „Massenverträgen“, fehlt es im Open Source Bereich aber gerade an der Möglichkeit die Vertragsleistung einheitlich zu lokalisieren. Insofern ist die Vertragskonstellation im Hinblick auf Miturheber im Open Source Bereich auch nicht mit derjenigen bei „Massenverträgen“ zu vergleichen. Das Bestehen eines schützenswerten Uniformitätsinteresses auf Seiten der Urheber würde hingegen voraussetzen, dass diese untereinander eine Vereinbarung zur Geltung einer bestimmten Rechtsordnung getroffen hätten. Solange es an einer solchen Vereinbarung fehlt, kann ein Uniformitätsinteresse nicht entstehen. Auf die GPL, mit welcher der Nutzer die Nutzungsrechte von mehreren Urhebern eingeräumt bekommt, findet daher Deutsches Recht Anwendung.

Zusammenfassung

Der dezentrale Entwicklungsprozess von Open Source Software führt ebenso wie deren nahezu weltweit erfolgender Vertrieb zu zahlreichen Schwierigkeiten im Hinblick auf Fragestellungen im Kontext des Internationalen Privatrechts.

Die Anwendung Deutschen Rechts ist aufgrund des Schutzlandsprinzips zunächst immer dann geboten, wenn Schutz vor Urheberrechtsverletzungen begehrt wird, die in Deutschland stattgefunden haben.

Daneben spielen im Umfeld des Open Source Modells aber auch grenzüberschreitende Verletzungshandlungen eine bedeutende Rolle. Dabei hat sich herausgestellt, dass beim Upload von Software auf einen Downloadserver, sowohl das Recht des Staates anwendbar ist, in dem der Uploadvorgang stattgefunden hat, als auch das Recht des Staates, in den die Software heruntergeladen wurde. Weiterhin ist auch das Recht des Staates anwendbar, in dem der Server steht.

[393] Zum Begriff *Weitnauer*, Der Vertragsschwerpunkt, S. 165 ff.
[394] Vgl. nur *Martiny* in: Reithmann / Martiny, Rn. 122 m.w.N.; sowie *Schnitzer* in: Festg. Schönenberger, 392 f.
[395] Vgl. *Schnitzer* in: Festg. Schönenberger, 392 f.

Liegt hingegen eine öffentliche Zugänglichmachung von Software vor, ist durch die hierdurch vermittelte weltweite Abrufbarkeit der Software, das Recht sämtlicher Staaten anwendbar. Es kommt somit zu einer Kumulation der anwendbaren Rechtsordnungen.

Weiterhin haben die Untersuchungen gezeigt, dass sich die anwendbare Rechtsordnung nach dem Schutzlandsprinzip richtet, soweit es um Fragen nach der Entstehung des Urheberrechts, dessen erste Inhaberschaft, der Schutzfähigkeit, der Übertragbarkeit und der Aktivlegitimation bei vertraglichen Rechtseinräumungen, sowie bei den Rechtsfolgen von Rechtsverletzungen, der Schutzdauer und des Erlöschens des Urheberrechts geht.
Stehen hingegen schuldrechtlichen Fragen der Wirksamkeit des Haftungs- und Gewährleistungsausschlusses sowie sonstige Fragen der Reichweite der GPL im Raum, richtet sich die anwendbare Rechtsordnung nach dem Vertragsstatut und somit grundsätzlich nach der Rechtswahl der Parteien.

Eine Rechtswahl ist allerdings weder der GPL, noch dem Schenkungsvertrag zu entnehmen, weshalb sich das anwendbare Recht nach Art. 28 EGBGB bestimmt.

Eine Ausnahme ist hiervon einzig für den Fall zu machen, in dem ein Verbraucher die Programmkopie eines Open Source Programms erwirbt und Deutsches Recht aufgrund von Art. 29 Abs. 1 EGBGB zur Anwendung gelangt.

Im Rahmen des Art. 28 Abs. 2 EGBGB ließ sich zwar die charakteristische Leistung sowohl im Zusammenhang mit dem Erwerb der Programmkopie als auch mit dem der weitergehenden Nutzungsrechte eindeutig bestimmen, allerdings nicht immer auch einheitlich lokalisieren.

Deutsches Recht findet im Rahmen des Erwerbs der Programmkopie immer dann Anwendung, wenn deren Anbieter seinen Sitz in Deutschland hat.

Der gewöhnliche Aufenthalt bzw. Sitz der Urheber an Open Source Programmen enthält hingegen nur dann eine Aussagekraft im Hinblick auf das anwendbare Recht, wenn entweder nur einzelne Urheber vorhanden sind, oder mehrere Urheber sich zu einem festen Entwicklungsverbund zusammengeschlossen haben und sich dadurch ihr Sitz bestimmen lässt.
Erfolgt der Zusammenschluss der Entwickler hingegen auf rein freiwilliger Basis und bestehen neben den Regelungen der GPL keine weitergehenden Absprachen zwischen den beteiligten Entwicklern, ist die von Art. 28 Abs. 2 EGBGB vorgesehene eindeutige Lokalisierung der charakteristischen Leistung mangels tatsächlicher Anknüpfungspunkte nicht möglich.

Es hat daher eine objektive Gewichtung der gegenseitigen kollisionsrechtlichen Interessen im Sinne von Art. 28 Abs. 1 EGBGB zu erfolgen, die letztlich zu einem stärkeren Interesse auf Seiten der Nutzer führt.
Die GPL ist daher bei gemeinschaftlichen Entwicklungen stets nach der Rechtsordnung zu beurteilen, in welcher der Nutzer seinen gewöhnlichen Aufenthalt oder seine Niederlassung hat.
Deutsches Recht ist somit immer im Rahmen der Auslegung der GPL maßgeblich, wenn der Nutzer seinen gewöhnlichen Aufenthalt in Deutschland hat.

Kapitel 5

F. Der virale Effekt

Im Hintergrund der bislang erfolgten rechtlichen Streitigkeiten im Open Source Bereich steht zumeist die in Ziff. 2 GPL enthaltene Pflicht zur Quellcodeoffenlegung.[396] In den bislang verhandelten gerichtlichen Verfahren[397] sind die betroffenen Unternehmen dieser Pflicht nicht nachkommen. Dies liegt häufig daran, dass mit der Offenlegung des Quellcodes auch immer ein Know-how Verlust einhergeht und von den Unternehmen Unsicherheiten im Hinblick auf die Reichweite dieser Offenlegungspflichten bestehen.

Aufgrund dieser Unsicherheiten im Hinblick auf Reichweite der in Ziff. 2 GPL enthaltenen Pflicht zur Offenlegung des Quellcodes entstand die Begrifflichkeit des „viralen Effekts", mit welcher die Angst vor einer „Ansteckung" der eigenen proprietären Entwicklungen durch deren Zusammenwirken mit GPL-lizenzierter Open Source Software umschrieben wird.[398] Viele Unternehmen sind unsicher, in welchem Ausmaß sie Open Source Software im Rahmen ihrer proprietären Geschäftsmodelle einsetzen können, ohne von den in der GPL enthaltenen Offenlegungspflichten betroffen zu sein.

Diese Unsicherheit wird zudem dadurch bestärkt, dass weder die Rechtsprechung noch die juristische Literatur bislang eindeutige Abgrenzungskriterien dafür geliefert hat, wie eine Programmentwicklung gestaltet sein muss, um eine Verletzung der in der GPL enthaltenen Verpflichtungen gänzlich auszuschließen.

Der Grund hierfür liegt zunächst einmal beim Wortlaut der GPL und hier insbesondere bei deren Regelung in Ziff. 2. Bereits bei einem ersten Blick auf den Wortlaut von Ziff. 2 GPL werden die Probleme mit der Bestimmung konkreter Kriterien ersichtlich.

---

[396] Diese folgt mittelbar aus Ziff 2 b GPL: *"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.",* indem die Verbreitung nur unter den Bedingungen GPL erfolgen darf.

[397] LG Frankfurt/M CR 2006, 729; LG Berlin CR 2006, 735; LG München I CR 2004, 774.

[398] Diese Angst wurde nicht zuletzt auch von Microsoft geschürt, indem der GPL stets eine „virale" Wirkung bescheinigt wurde, vgl. die Rede von *Craig Mundie* (Microsoft Senior Vice President) vor der New York University Stern School of Business, http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.mspx; mittlerweile hat sich die Bezeichnung auch in der Literatur etabliert, vgl. ; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 112; *Determann*, GRUR Int. 2006, 645, 649; *Andréewitch*, MR 2005, 240, 241.

So folgt aus Ziff. 2 Abs. 1 b GPL eine Lizenzierungspflicht, wenn die zu verbreitende oder zu veröffentlichende Arbeit „*ganz oder teilweise von dem Programm (...) abgeleitet ist*“,[399] dass seinerseits unter der GPL steht.
Bereits bei einem ersten Blick auf diese Voraussetzungen der GPL wird ersichtlich, dass eine genaue Bestimmbarkeit ihres Wirkungsbereichs aufgrund der verwendeten allgemeinen Begriffe problematisch ist.

Einer Konkretisierung des Wirkungsbereichs wird letztlich aber auch nur bedingt durch Ziff. 2 Abs. 2 GPL erreicht, indem dort drei Voraussetzungen genannt werden, bei deren kumulativen Vorliegen die Lizenzierungsverpflichtung entfällt.

So muss die Eigenentwicklung:

- über selbstständige identifizierbare Teile verfügen, die nicht von dem Programm, welches unter der GPL lizenziert ist, abgeleitet sind,
- diese identifizierbaren Teile müssen vernünftigerweise als unabhängige und eigenständige Datenwerke für sich selbst zu betrachten sein und
- müssen als eigenständiges Datenwerk weitergegeben werden.

Zusammenfassend lassen sich daher aus Ziff. 2 GPL drei Fallgruppen entnehmen, bei deren Vorliegen die Lizenzierungspflicht eingreifen soll.

Die GPL verpflichtet zur Lizenzierung wenn:

- es sich bei der Eigenentwicklung um eine abgeleitete Arbeit handelt (Ziff. 2 Abs. 1 b GPL)
- die Eigenentwicklung zwar identifizierbare Teile aufweist, die auch nicht abgeleitet sind, aber vernünftigerweise nicht als unabhängig und eigenständig betrachtet werden können (Ziff. 2 Abs. 2 GPL).
- die Eigenentwicklung zwar identifizierbare und zugleich nicht abgeleitete Teile aufweist, die auch vernünftigerweise als unabhängig und eigenständige

[399] So die Übersetzung in der inoffiziellen Deutschen Fassung der GPL (vgl. http://www.gnu.de/gpl-ger.html) In der englischen Originalfassung beinhaltet Ziff. 2 Abs. 1 b GPL folgende Formulierung: *“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.”*

> Datenwerke zu betrachten sind, aber nicht als eigenständiges Datenwerk weitergegeben werden (Ziff. 2 Abs. 2 GPL).

Was aber ist unter einer abgeleiteten Arbeit zu verstehen, ab wann hat man vernünftigerweise von unabhängigen und eigenständigen Teilen auszugehen und unter welchen Voraussetzungen werden diese nicht als eigenständige Werke weitergegeben?

Die Suche nach konkreten Abgrenzungskriterien gestaltet sich aufgrund dieser allgemeinen Begriffe erwartungsgemäß schwierig. Hinzu kommt, dass eine saubere Abgrenzung von Open Source Software und proprietärer Eigenentwicklung wohl nur unter Berücksichtigung der technischen Gegebenheiten möglich sein dürfte. Hiervon geht zumindest die FSF aus, indem sie für eine Abgrenzung auf die Zusammenschau von technischer Kommunikationsart und Inhalt der ausgetauschten Informationen abstellen will.[400]

Bevor nachfolgend auf die einzelnen Voraussetzungen der Ziff. 2 GPL näher eingegangen wird, sind zunächst diejenigen Fallgestaltungen herauszustellen, in denen sich die Pflicht zur Offenlegung des Quellcodes offensichtlich nicht auswirken kann.

I. Kein Abschluss der GPL erforderlich

Von der rechtlichen Konsequenz her eindeutig ist dies zunächst in solchen Fällen, in denen es bereits am Abschluss der GPL fehlt. Obwohl dieses Ergebnis in rechtlicher Hinsicht offensichtlich ist, gestaltet sich die Frage, in welchen Fällen es überhaupt zu einem Abschluss der GPL gekommen ist, mitunter schwierig. Dies liegt daran, dass die GPL nur in wenigen Fällen durch eine ausdrückliche Willenserklärung zustande kommt.[401] Vielmehr erhält jeder

[400] Vgl. Faq-Seite der FSF, http://www.fsf.org/licensing/licenses/gpl-faq.html: *„We believe that a proper criterion depends both on themechanism of communication (exec, pipes, rpc, function calls within a shared address space,etc.) and the semantics of the communication (what kinds of information are interchanged).”*

[401] Dies liegt zunächst einmal daran, dass die GPL häufig nicht wirksam in den Programmüberlassungsvertrag einbezogen wird. Für eine wirksame Einbeziehung müsste – jedenfalls im Verbraucherverkehr – bereits bei Abschluss des Programmüberlassungsvertrags auf die Lizenzbedingungen hingewiesen oder die Möglichkeit verschafft werden von ihnen in zumutbarer weise Kenntnis zu nehmen. Diese Voraussetzungen werden in der Praxis oftmals nicht erfüllt (ebenso in der Einschätzung *Jaeger / Metzger*, Open Source Software, Rn. 174; im Ergebnis auch *Spindler*, Rechtsfragen bei Open Source, C. Rn. 47 f), insbesondere kann eine Einbeziehung in den Überlassungsvertrag auch nicht dadurch „nachgeholt“ werden, dass der Nutzer erst während der Programminstallation zur Annahme der Lizenzbedingungen verpflichtet wird (vgl. nur *Marly*, Softwareüberlassungsverträge, 473; *Schneider*, Handbuch des EDV-Rechts, J. Rn. 8; *Schuhmacher*, CR 2000, 641, 642 f). Hinzu kommt, dass beim Download zahlreicher Programme überhaupt keine ausdrückliche Annahme der GPL vorgesehen ist, sondern die Lizenzbedingungen lediglich – entsprechend der Vorgaben im

Nutzer von GPL-lizenzierter Software beim Erwerb der Programmkopie mit dieser zusammen auch das dauerhafte Angebot auf Abschluss der GPL übermittelt,[402] dass zumeist nur konkludent angenommen wird.[403]

Für die konkludente Annahmeerklärung reicht dabei ein Verhalten des Nutzers, das vom Standpunkt eines unbeteiligten Dritten unzweideutig auf den Annahmewillen schließen lässt.[404]

Im Zusammenhang mit der GPL wird eine solche Annahme immer dann vorliegen, wenn Handlungen vorgenommen werden, für die der Nutzer gem. § 69 c UrhG auf eine Zustimmung des Urhebers angewiesen ist.[405] Dabei ist im Hinblick auf § 69 c Nr. 2 UrhG zum Einen an die Vornahme von Entwicklungshandlungen zu denken, die ihrerseits von der "freien Benutzung" im Sinne von § 24 UrhG abzugrenzen sind.[406] Zum Anderen können aber auch Vertriebshandlungen vor dem Hintergrund von § 69 c Nr. 3 und 4 UrhG relevant werden.[407]

Anhang der GPL – als beigefügter License.txt enthalten sind, auf die jeweils zu Beginn der Quelltexte Bezug genommen wird (so z. B. bei den Linux-Distributionen von Suse, Red hat, Ubuntu, Debian und Gentoo).

[402] Vgl. *Plaß*, GRUR 2002, 670, 676 f; die GPL verpflichtet den Veräußerer allerdings ausschließlich dazu den Lizenztext mitzuliefern, insofern missverständlich *Spindler*, Rechtsfragen bei Open Source, C. Rn. 51, wenn er davon spricht, dass die GPL den Veräußerer dazu verpflichten würde *„die GPL bei Weiterverbreitung der Software dem Erwerber aufzuerlegen“*.

[403] Dies folgt jedoch nicht bereits aus Ziff. 5 S. 3 GPL, wonach jedes Ändern oder Verbreiten als Annahmeerklärung behandelt wird, da hierin – jedenfalls gegenüber Verbrauchern – eine unwirksame Tatsachenfiktion im Sinne von § 308 Nr. 5 BGB zu sehen ist (ähnlich auch *Schiffner*, Open Source Software, S. 185, der die Regelung als bedenklich einstuft; von einer Unwirksamkeit der Regelung geht auch *Koch*, CR 2000, 333, 339, aus, allerdings noch im Hinblick auf § 11 Nr. 15 AGBG), die darüber hinaus erst *nach* – nicht aber bereits *vor* – dem Vertragsschluss Wirkung entfalten könnte. Zum konkludenten Abschluss der GPL unter gleichzeitigem Verzicht auf den Zugang der Annahmeerklärung (§ 151 BGB), vgl. LG Frankfurt/M CR 2006, 729, 731; *Metzger / Jaeger*, GRUR Int. 1999, 839, 846; *Jaeger / Metzger*, Open Source Software, Rn. 177; *Koch*, CR 2000, 333, 340; *Spindler / Wiebe*, CR 2003, 873, 874; *Lejeune*, ITRB 2003, 10, 11; *Schiffner*, Open Source Software, S. 185; sowie *Bartosz*, MR 2005, 40, 45.

[404] *Kramer* in: Müko, Vor §§ 116 Rn. 23 m.w.N.

[405] Vgl. LG Frankfurt/M CR 2006, 729, 731.

[406] Zur freien Benutzung sogleich ab S. 101.

[407] Vgl. hierzu S. 103 ff.

1.Zustimmungsbedürftige Entwicklungshandlungen

a.Bearbeitung, § 69 c Nr. 2 UrhG

Sofern es im Umfeld von Open Source Software zu Softwareentwicklungen kommt, sind diese regelmäßig daraufhin zu untersuchen, ob eine eigenständige Entwicklung oder aber eine zustimmungsbedürftige Bearbeitung im Sinne von § 69 c Nr. 2 UrhG vorliegt. Nach dem Wortlaut des § 69 c Nr. 2 UrhG sind sowohl die Übersetzung, die Bearbeitung, das Arrangement und andere Umarbeitungen eines Computerprogramms zustimmungsbedürftig. Als Oberbegriff wird im Rahmen des § 69 c Nr. 2 UrhG die Umarbeitung angeführt, was letztlich auf die wörtliche Übernahme aus Art. 2 Abs. 3 RBÜ zurückzuführen ist.[408] Während unter Übersetzungen vor allem die Übertragung in eine andere Programmiersprache gefasst wird,[409] sollen unter Arrangements die Veränderung der Programmmodule durch Neuanordnung derselben zu verstehen sein.[410] Insgesamt sind die einzelnen Tatbestandsmerkmale jedoch nur schwer auseinander zu halten, weshalb im Hinblick auf die einheitliche Rechtsfolge zumeist eine Differenzierung zwischen diesen für entbehrlich erachtet wird.[411] Insgesamt besteht Einigkeit darüber, dass es sich bei § 69 c Nr. 2 UrhG um ein weit gefasstes Recht handelt, dem alle Abänderungen eines geschützten Computerprogramms unterfallen.[412]

Als zustimmungsbedürftige Handlungen werden demnach insbesondere die Ergänzung des Quell- oder auch Objektcodes, die Veränderung von Kommentierungen innerhalb des Quellcodes, sowie die Fehlerbeseitigung – unabhängig ob diese mittels neuer Releases, Updates, Upgrades oder Wartungsarbeiten erfolgt – die Erreichung von Mehrplatzfähigkeit oder auch die Portierung auf andere Hardware- und/oder Betriebssysteme verstanden. [413]

Neben dem Hinzufügen von Code wird von der Rechtsprechung eine zustimmungsbedürftige Bearbeitung aber auch dann angenommen, wenn dem Programm lediglich Funktionen

[408] Der deutsche Gesetzgeber entschied sich zu einer wörtlichen Übernahme der Regelung, um eine einheitliche europäische Regelung zu ermöglichen, vgl. amtl. Begründung BT-Drucks. 12/4022, S. 11.
[409] *Marly*, Softwareüberlassungsverträge, Rn. 1190 f; *Grützmacher* in: Wandtke / Bullinger, § 69 c Rn. 18.
[410] *Grützmacher* in: Wandtke / Bullinger, § 69 c, Rn. 19; weitere Beispiele für Arrangements bei *Koch*, NJW-CoR 1994, 293, 300 Fn. 24.
[411] Vgl. *Loewenheim* in: Schricker, § 69 c Rn. 12 a.E.; *Dreier* in: Dreier / Schulze, § 69 c Rn. 15.
[412] *Dreier* in: Dreier / Schulze, § 69 c Rn. 15; *Hoeren* in: Möhring / Nicolini, § 69 c Rn. 8; ebenso *Loewenheim* in: Schricker, § 69 c Rn. 13.
[413] Eingehend hierzu *Grützmacher* in: Wandtke / Bullinger, § 69 c Rn. 20 m.w.N.

genommen werden, also der Quellcode ausschließlich durch das Löschen einzelner Teile verändert wird.[414]

Zusammenfassend ist daher festzuhalten, dass in jeder Veränderung von Open Source Software, die nicht bloß der bestimmungsgemäßen Benutzung des Programms dient,[415] eine zustimmungsbedürftige Handlung liegt, bei deren Vornahme von einem konkludenten Vertragsschluss durch den Bearbeiter ausgegangen würde.

b.Freie Benutzung, § 24 UrhG

Liegt hingegen ausschließlich eine freie Benutzung im Sinne von § 24 UrhG vor, ist diese von keiner Zustimmung des Urhebers abhängig, so dass der Nutzer nicht auf den Abschluss der GPL angewiesen wäre. Aus der Vornahme solcher Nutzungshandlungen folgt daher keine konkludente Annahme des GPL Lizenzvertrags.

Unsicherheiten bestehen in diesem Bereich gleichwohl aus dem Grund, da die Grenzziehung zwischen der freien Benutzung im Sinne von § 24 UrhG und zustimmungsbedürftiger Bearbeitung im Sinne von § 69 c Nr. 2 UrhG nicht trennscharf verläuft.

Von einer „freien Benutzung“ im Sinne von § 24 UrhG geht die Rechtsprechung aus, wenn angesichts der Eigenart des neu geschaffenen Werks die entlehnten eigenpersönlichen Züge

[414] Für die Beseitigung der sog. Dongle-Abfrage in Programmen, vgl. OLG Düsseldorf CR 1997, 337, 339; OLG Karlsruhe CR 1996, 341, 342.

[415] Vor dem Hintergrund der bestimmungsgemäßen Benutzung nimmt *Grützmacher* in: Wandtke / Bullinger, § 69 Rn. 20, Bezug auf das Urteil des OLG Hamburg CR 1998, 332, 333 f, und kommt zu dem Ergebnis, dass das sog. Customizing keine Bearbeitung im urheberrechtlichen Sinne sei. Dem kann allerdings nur nach vorheriger Eingrenzung des Begriffs „Customizing“ gefolgt werden, da hierunter grundsätzlich sämtliche Anpassungen an Standardsoftware verstanden werden, also sowohl das einfache Setzen von Parametern – welches wohl der Vorstellung von *Grützmacher* zugrunde lag – als auch Programmänderungen durch Quellcodeänderungen, die offensichtlich die Grenze zur zustimmungsbedürftigen Bearbeitung überschreiten; zum Begriff des Customizing vgl. Geoinformatik-Lexikon, http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=373; unter Wikipedia lässt sich ebenfalls der der Eintrag finden, dass mit „Customizing“ die Anpassung durch Programmänderungen oder durch das Setzen von Parametern verstanden wird, http://de.wikipedia.org/wiki/Customizing. Letztlich wird auch in der Literatur auch davon ausgegangen, dass vom Customizing im Einzelfall „auch Codeanpassungen und/ -ergänzungen (…) umfasst sein“ können, vgl. *Koch*, ITRB 2005, 140, 141.
Während bei der Einstellung der Parameter lediglich solche Codeteile verändert oder erstmals eingetragen werden, die gerade zu diesem Zweck vorhanden sind, erfolgt die Programmänderung regelmäßig durch Umprogrammierung von Codeteilen und hat somit eine Quellcodeänderung zur Folge, weshalb das Customizing in diesen Fällen als Bearbeitung im Sinne von § 69 c Nr. 2 UrhG einzustufen ist. Sofern die Veränderungen im Quellcode allerdings ausschließlich an den hierfür vorgesehenen Stellen durch das Eintragen entsprechender Parameter und Umgebungsvariablen erfolgt, stellen diese Eingriffe zwar im Grundsatz tatbestandliche Bearbeitungen im Sinne des § 69 c Nr. 2 UrhG dar, werden allerdings zumeist der bestimmungsgemäßen Benutzung des Programms dienen und dürfen mithin vom berechtigten Benutzer gem. § 69 d Abs. 1 UrhG zustimmungsfrei vorgenommen werden; vgl. *Schulze* in: Dreier / Schulze, § 69 d Rn. 8.

des geschützten älteren Werkes verblassen.[416] Insofern muss sich die Eigenart des neuen Werkes gegenüber dem älteren Werk in besonderem Maß hervortun.[417] Es ist daher danach abzugrenzen, ob die Eigenentwicklung einen so weiten Abstand zum GPL-lizenzierten Werk hat, dass noch eine erlaubnisfrei zulässige freie Benutzung vorliegt, oder ob die Entwicklung in derart vielen Teilen Übereinstimmungen aufweist, das von einer erlaubnispflichtigen Bearbeitung auszugehen ist.

Offensichtlich ausgeschlossen ist eine freie Benutzung allerdings, wenn das vorbestehende Werk nicht nur zum Anlass, sondern auch in einzelnen Teilen für die Eigenentwicklung verwendet wurde. Zwar besteht auch in solchen Fällen die Möglichkeit, dass der übernommene Teil hinter den Eigenentwicklungen deutlich verblasst, allerdings hat sich der Benutzer hierbei dann nicht nur von dem GPL-lizenzierten Werk anregen lassen, sondern auch Teile desselben auf technischem Weg übernommen und somit eigene persönliche Leistungen erspart. Dies ist aber nicht vom Schutzumfang des § 24 UrhG erfasst,[418] weshalb es in diesen Fällen nicht gerechtfertigt ist, von einer freien Benutzung auszugehen.

Eine Ausnahme ist hiervon wiederum für solche Programmteile zu machen, die dem allgemeinen Programmierwissen zuzurechnen sind und sich mangels urheberrechtlicher Schutzfähigkeit mithin grundsätzlich verwendet lassen.[419]

Insgesamt ist somit festzuhalten, dass GPL-lizenzierte Programme von Unternehmen unter den benannten (unscharfen) Voraussetzungen als Anregung für eigene Entwicklungsleistungen verwendet werden können. Die Grenze zwischen zulässiger Anregung und erlaubnispflichtiger Umarbeitung lässt sich dabei jedoch nicht allgemeingültig festlegen, sondern ist in jedem Einzelfall gesondert zu bestimmen. Aufgrund dieser Ungewissheit sind Entwicklungen, die sich an bestehenden GPL-lizenzierten Programmen orientieren stets mit einem gewissen Investitionsrisiko verbunden. Es besteht letztlich immer die Gefahr, dass die Eigenentwicklungen dem Original zu ähnlich wird und somit eine zustimmungsbedürftige Bearbeitung darstellt. Für diese wäre das Unternehmen aber auf den Abschluss der GPL angewiesen, weshalb die entsprechende Entwicklungstätigkeit zugleich als konkludente Annahmeerklärung zu werten wäre. Im Rahmen des Programmvertriebs hätte

[416] BGH GRUR 1994, 191, 193 – *Asterix-Persiflagen*, m.w.N.; *Ulmer*, Urheber- und Verlagsrecht, S. 275; *Dreier* in: Dreier / Schulze, § 24 Rn. 8.
[417] *Ahlberg* in: Möhring / Nicolini, § 24 Rn. 7.
[418] BGH GRUR 1981, 267, 269 – Dirlada; BGH GRUR 1978, 305, 306 – Schneewalzer; im Hinblick auf die die Verwendung von Ausschnitten aus digitalen Fotos *Schulze* in: Dreier / Schulze, § 24 Rn. 10; noch zum § 13 Abs. 1 LUG BGH in GRUR 1965, 45, 47 – Stadtplan; sowie BGH GRUR 1958, 500, 502 – Mecki-Igel.
[419] Vgl. nur *Loewenheim* in: Schricker, § 69 c Rn. 16; *Dreier* in: Dreier / Schulze, § 69 c Rn. 17.

das Unternehmen mithin die Regelungen der GPL und insbesondere die Pflicht zur Offenlegung des Quellcodes zu berücksichtigen.

2.Zustimmungsbedürftige Vertriebstätigkeiten

Neben der konkludenten Annahme des GPL Lizenzvertrags durch die Vornahme von zustimmungsbedürftigen Bearbeitungen im Sinne von § 69 c Nr. 2 UrhG könnte eine solche auch in den Fällen gegeben sein, in denen GPL-lizenzierte Software durch die Unternehmen vertrieben wird. Der Vertrieb von GPL-lizenzierter Software kann dabei vor allem unter den Voraussetzungen der §§ 69 c Nr. 3 und 4 UrhG zustimmungsbedürftig sein.

a.Vertrieb körperlicher Werkexemplare, § 69 c Nr. 3 UrhG

Nach § 69 c Nr. 3 UrhG sind dessen Voraussetzungen bei jeder Form der Verbreitung eines Computerprogramms erfüllt, unabhängig davon, ob es sich um das Original oder ein Vervielfältigungsstück davon handelt. Vom Begriff der Verbreitung im Sinne von § 69 c Nr. 3 UrhG werden sowohl das Inverkehrbringen als auch das Anbieten gegenüber der Öffentlichkeit umfasst.[420]

Von einem Inverkehrbringen ist dabei auszugehen, wenn das Computerprogramm aus der internen Betriebssphäre in die Öffentlichkeit gebracht wird.[421] Das Anbieten stellt hingegen eine Vorstufe hierzu dar. Es liegt bereits vor, wenn eine abstrakte und unbestimmte Aufforderung zum Eigentums- oder Besitzerwerb gegenüber der Öffentlichkeit gegeben ist.[422]

Der Begriff der Öffentlichkeit entspricht dem von § 17 Abs. 1 UrhG,[423] weshalb es ausreicht, wenn ein Computerprogramm einer Mehrzahl von nicht persönlich verbundenen Personen zugänglich gemacht wird.[424] Hieraus folgt, dass sowohl das Anbieten als auch das Inverkehrbringen von Computerprogrammen zustimmungsfrei zulässig sind, sofern sie

[420] Der Begriff der Verbreitung entspricht nach der Gesetzesbegründung dem aus § 17 UrhG, vgl. amtl. Begründung BT-Drucks. 12/4022, S. 11.
[421] Die Zahl der Personen, an die sich das Angebot richtet, ist hingegen unerheblich, vgl. BGH GRUR 1991, 316, 317 – Einzelangebot.
[422] *Loewenheim* in: Schricker, § 69 c Rn. 22; a.A. im Hinblick auf die Bestimmtheit des Angebots KG GRUR 1983, 174 – Videoraubkassetten.
[423] Vgl. amtl. Begründung BT-Drucks. 12/4022, S. 11.
[424] Zum Begriff der Öffentlichkeit, vgl. *Loewenheim* in: Schricker, § 17 Rn. 10 f m.w.N.; im Hinblick auf den zugrunde liegenden § 15 Abs. 3 UrhG, vgl. die amtl. Begründung BT-Drucks. 684/02, S. 37.

innerhalb des persönlichen Freundeskreises stattfinden. Ob darüber hinaus auch die Weitergabe innerhalb eines Unternehmens noch zustimmungsfrei möglich ist, erscheint zweifelhaft. Jedenfalls in größeren Unternehmen und Behörden wird man wohl davon ausgehen müssen, dass eine Weitergabe des Programms gegenüber der Öffentlichkeit erfolgt und somit der Zustimmung des Urhebers bedarf.[425]

b.Vertrieb unkörperlicher Werkexemplare, § 69 c Nr. 4 UrhG

Ebenso wie bei der körperlichen Weitergabe eines Computerprogramms wird auch bei dessen unkörperlicher Wiedergabe im Sinne von § 69 c Nr. 4 UrhG maßgeblich darauf abgestellt, ob es einer Vielzahl von nicht persönlich verbundenen Nutzern gleichzeitig oder sukzessive wahrnehmbar oder zugänglich gemacht wird.[426]

Wie bereits im Rahmen der körperlichen Verbreitung kommt es daher auch für die Zustimmungsbedürftigkeit von Vertriebshandlungen darauf an, ob diese gegenüber der Öffentlichkeit oder lediglich innerhalb eines persönlich verbundenen Personenkreises erfolgt. Insofern wird man auch hier davon auszugehen haben, dass jedenfalls in größeren Unternehmen und Behörden die Verbreitung eines Computerprogramms im firmeneigenen Netz gem. § 69 c Nr. 4 UrhG einer Zustimmung des Urhebers bedarf.

Eindeutig zustimmungsbedürftig ist zudem das Bereithalten eines Computerprogramms zum Download für eine unbestimmte Zahl an Nutzern. Zu berücksichtigen ist allerdings auch, dass selbst im Downloadvorgang unter Umständen eine zustimmungsbedürftige Verbreitungshandlung liegen kann, sofern dieser über ein Peer-to-Peer Netzwerks erfolgt und deshalb zugleich auch eine öffentliche Zugänglichmachung beinhaltet.[427]

[425] Str., wie hier wohl auch BGH GRUR 1985, 924 f. – Schallplattenimport; BGHZ 17, 376, 380 f. – Betriebsfeier; *Kotthoff*, GRUR 1997, 597, 600; *Spindler*, GRUR 2002, 105, 108 f; *Grützmacher* in: Wandtke / Bullinger, § 69 c Rn. 27; *Andréewitch*, MR 2005, 36, 39; a.A. OLG Hamburg GRUR Int. 1970, 377, 379 – Polydor.

[426] Vgl. nur *Grützmacher* in: Wandtke / Bullinger, § 69 c Rn. 50.

[427] Dies liegt daran, dass beispielsweise während des Downloads eines Torrent-Files zwangsläufig alle Datenpakete, die der Nutzer heruntergeladen hat auch Beliebigen anderen zum Download zur Verfügung stehen. Insofern liegt während des Downloadvorgangs zugleich auch eine öffentliche Zugänglichmachung vor, die einer Zustimmung durch den Urheber gem. § 69 c UrhG bedarf.

Zwischenergebnis

Insgesamt kann daher festgehalten werden, dass grundsätzlich sowohl der körperliche als auch der unkörperliche Vertrieb von GPL-lizenzierter Software einer Zustimmung bedarf, sofern er gegenüber der Öffentlichkeit erfolgt. Insofern liegt es nahe, bei einer entsprechenden Vertriebstätigkeit stets auch vom Vorliegen einer konkludente Annahme der GPL auszugehen.

c.Ausnahme aufgrund des Erschöpfungsgrundsatzes?

Etwas anderes könnte allerdings in den Fällen gelten, in denen Unternehmen GPL-lizenzierte Programmkopien vertreiben, die sie zuvor selbst in körperlicher Form erworben haben. Für den Vertrieb solcher Vervielfältigungsstücke könnte die Zustimmung des Urhebers aufgrund des in §§ 69 c Nr. 3 S. 2, 17 Abs. 2 UrhG enthaltenen Erschöpfungsgrundsatzes entbehrlich sein.

Soweit der in §§ 69 c Nr. 3 S. 2, 17 Abs. 2 UrhG verankerte Erschöpfungsgrundsatz anwendbar ist, erschöpft sich das Verbreitungsrecht am Vervielfältigungsstück eines Computerprogramms, sofern dieses mit Zustimmung des Rechtsinhabers im Gebiet der Europäischen Union oder eines anderen Vertragsstaates des Abkommens über den Europäischen Wirtschaftsraum im Wege der Veräußerung in Verkehr gebracht wurde. Auf die Entgeltlichkeit des Verbreitungsvorgangs kommt es dabei nicht an, da mit der in § 69 c Nr. 3 S. 2 UrhG enthaltenen Veräußerung nach allgemeiner Ansicht nicht nur die käufliche, sondern jede dauerhafte, insbesondere auch die schenkungsweise Übereignung oder Entäußerung des Eigentums erfasst wird.[428] Der Begriff „Veräußerung“ im Sinne von § 69 c Nr. 3 S. 2 UrhG lässt demnach keine Rückschlüsse auf den Vertragstyp des zugrunde liegenden Veräußerungsgeschäfts zu. Maßgeblich ist vielmehr, dass sich der Eigentümer seiner Verfügungsmöglichkeit über das Vervielfältigungsstück dauerhaft begibt.[429]

Demzufolge stellt, neben der käuflichen, insbesondere auch die schenkungsweise Überlassung GPL-lizenzierter Software eine Veräußerung im Sinne von § 69 Nr. 3 S. 2 UrhG dar. Auch hier erwirbt der Nutzer die Programmkopie auf unbestimmte Zeit, so dass sich der Eigentümer seiner Verfügungsmöglichkeiten begibt.

[428] BGH GRUR 1995, 673, 675 f. – Mauer-Bilder; vgl. auch *Heerma* in: Wandtke / Bullinger, § 17, Rn. 13; *Kroitzsch* in: Möhring / Nicolini, §17, Rn. 41; *Vinck* in: Nordemann / Fromm, § 17, Rn. 9.
[429] Vgl. *Vinck* in: Nordemann / Fromm, § 17, Rn. 9; *Loewenheim* in: Schricker, § 17, Rn. 39 m.w.N.

Insofern wäre eigentlich davon auszugehen, dass sich auch das Verbreitungsrecht an GPL-lizenzierten Programmkopien erschöpfen kann und für Unternehmen somit die Möglichkeit besteht diese weiter zu verbreiten, ohne hierfür auf die Zustimmung des Urhebers angewiesen zu sein. Mangels Zustimmungserfordernis wäre die Vertriebstätigkeit in diesen Fällen nicht dazu geeignet, als konkludente Annahmeerklärung der GPL gewertet zu werden, weshalb die Unternehmen nicht in den Wirkungsbereich der GPL kommen könnten. Eine Pflicht zur Quellcodeoffenlegung wäre somit ausgeschlossen.

d.Open Source spezifische Ausnahme vom Erschöpfungsgrundsatz?

Obwohl die Voraussetzungen von § 69 c Nr. 3 S. 2 UrhG nach dessen Wortlaut auch im Rahmen der Verbreitung GPL-lizenzierter Software erfüllt sind, geht ein Teil der Literatur davon aus, dass der Erschöpfungsgrundsatz bei der Verbreitung von Open Source Software keine Anwendung finden könne.

aa)Beschränkte Einräumung des Nutzungsrechts

Zur Begründung wird dabei darauf verwiesen, dass sich der Open Source Bereich dadurch auszeichne, dass das Verbreitungsrecht lediglich beschränkt auf die Nutzung als Open Source Software eingeräumt werde und sich mithin auch nur in dieser Form erschöpfen könne.[430] Dieser Ansicht liegt die Annahme zugrunde, dass es sich bei der Nutzung als Open Source Software zum Einen um eine eigenständige Nutzungsart handelt,[431] und sich die inhaltliche Beschränkung der Nutzungsrechtseinräumung zum Anderen unmittelbar und in gleicher Weise auch auf die Erschöpfung auswirkt.[432] Beide Annahmen treffen aber – unabhängig voneinander – nicht zu.

Selbst wenn man noch das Bestehen einer eigenständigen Open Source Nutzungsart anerkennen wollte,[433] kann spätestens seit der OEM-Rechtsprechung des BGH[434] nicht mehr

[430] *Koch*, CR 2000, 333, 335 f.
[431] *Koch*, CR 2000, 333, 336, will zumindest im Hinblick auf Linux und linuxbasierte Applikationsprogramme von einer eigenständigen Nutzungsart ausgehen.
[432] *Loewenheim* in: Schricker, § 17, Rn. 49 m.w.N., der dies allerdings nur für die Stufe der Erstverbreitung vertritt, vgl. § 69 c Rn. 30; *Vinck* in: Nordemann / Fromm, § 69 c Rn. 6; *Koch*, CR 2000, 333, 335.
[433] Ausdrücklich gegen eine solche Annahme *Deike*, CR 2003, 9, 16.

davon ausgegangen werden, dass die inhaltlich beschränkte Einräumung von Nutzungsrechten zwangsläufig auch eine beschränkte Erschöpfung zur Folge hat.[435] Gerade das Gegenteil nahm der BGH zu Recht an. Danach tritt, selbst bei der inhaltlich beschränkten Einräumung von Nutzungsrechten, die bei Anerkennung einer eigenständigen Nutzungsart dinglich wirksam wäre, eine vollständige Erschöpfung des Verbreitungsrechts ein, sofern die Voraussetzungen des § 69 c Nr. 3 S. 2 UrhG gegeben sind. Insofern ist auch für den Open Source Bereich davon auszugehen, dass kein Zusammenhang zwischen der beschränkten Einräumung des Verbreitungsrechts und einer dadurch limitierten Erschöpfung besteht.[436]

bb)Faktischer Ersterwerb

Dieses Ergebnis wird teilweise auch von Befürwortern einer begrenzten Erschöpfungswirkung im Open Source Bereich geteilt.[437] Um gleichwohl eine Ausnahme von der, durch §§ 69 c Nr. 3 S. 2 UrhG angeordneten, Erschöpfung des Verbreitungsrechts zu erreichen, stellen sie sich auf den Standpunkt, dass im Rahmen des Open Source Modells jeder Zweiterwerb rechtlich und faktisch einen Ersterwerb darstellen würde.[438] Begründet wird dies damit, dass der Nutzer, unabhängig von der Erwerbsstufe, die Nutzungsrechte jeweils vom ursprünglichen Rechtsinhaber eingeräumt bekommt. Dies stelle eine Besonderheit dar, die es rechtfertige, den jeweiligen Erwerbsvorgang als Ersterwerb zu behandeln.

Hiergegen bestehen jedoch erhebliche Bedenken, die sich nicht zuletzt vor dem Hintergrund der dadurch gegebenen Umgehung des zwingenden[439] Erschöpfungsgrundsatzes ergeben. Mit dem Argument des „mehrfachen“ Ersterwerbs auf unterschiedlichen Erwerbsstufen ließe sich letztlich nahezu jedes Vertriebsmodell in einer Art und Weise gestalten, in der keine Erschöpfung eintritt. Hierdurch würde aber der mit dem Erschöpfungsgrundsatz angestrebte Verkehrsschutz, der insbesondere auch von klaren und übersichtlichen Verhältnissen im

[434] BGH ZUM 2000, 1079, 1081 - OEM-Version.
[435] Sehr str., wie hier *Jaeger*, ZUM 2000, 1070, 1074; *Bartsch*, K&R 2000, 612; *Loewenheim* in: Schricker, § 69 c Rn. 30; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 485 ff; ausführlich zur Gegenansicht m.w.N. *Grützmacher* in: Wandtke / Bullinger, § 69 c, Rn. 81 ff.
[436] So auch *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 486 ff.
[437] *Schiffner*, Open Source Software, S. 140.
[438] So *Koch*, CR 2000, 333, 339*Schiffner*, Open Source Software, S. 141; *Heussen*, MMR 2004, 445, 449 f; ähnlich wohl auch noch *Jaeger / Metzger*, Open-source-Software, S. 32 Fn. 126; (1. Aufl.) die in der aktuellen Auflage jedoch von einer vollständigen Erschöpfung ausgehen, vgl. *Jaeger / Metzger*, Open Source Software, Rn. 131 ff.
[439] Vgl. nur *Loewenheim* in: Schricker, § 69 c Rn. 32.

Waren- und Wirtschaftsverkehr lebt, weitgehend aufgehoben.[440] Aus diesem Grund ist auch im Umfeld von Open Source Software keine Ausnahme vom Erschöpfungsgrundsatz gerechtfertigt.

cc)Teleologische Reduktion

Letztlich ließe sich noch über eine Einschränkung des Erschöpfungsgrundsatzes durch eine teleologische Reduktion der §§ 17 Abs. 2, 69 c Nr. 3 S. 2 UrhG nachdenken.[441] Hierfür könnte sprechen, dass die Rechtfertigung für die von §§ 17 Abs. 2, 69 Nr. 3 S. 2 UrhG angeordnete Erschöpfung nicht nur in einem Schutz des Waren- und Wirtschaftsverkehrs, sondern auch in der Befriedigung der wirtschaftlichen Partizipationsinteressen des Urhebers gesehen werden müsse.[442] Im Gegensatz zum Vertrieb von proprietärer Software bleibt aber gerade dieses wirtschaftliche Partizipationsinteresse des Urhebers im Open Source Bereich unbefriedigt, weshalb eine Einschränkung des Erschöpfungsgrundsatzes zunächst vertretbar erscheint.

Dagegen spricht aber bereits der Wortlaut des § 69 Nr. 3 S. 2 UrhG, wonach es für die darin angeordnete Erschöpfung gerade nicht Voraussetzung ist, dass der Urheber auch wirtschaftlich von der Veräußerung profitiert.[443] Es ist allein ausreichend, dass dem Urheber die Möglichkeit der wirtschaftlichen Partizipation zugestanden hat, was aber auch im Open Source Bereich gegeben ist. Für das Vorliegen einer „Veräußerung“ ist deren Entgeltlichkeit gerade nicht Voraussetzung, weshalb die fehlende wirtschaftliche Partizipation nun nicht im Rahmen der teleologischen Auslegung zur Begründung einer Einschränkung des Erschöpfungsgrundsatzes herangezogen werden kann.

Im Ergebnis ist daher festzuhalten, dass die von §§ 17 Abs. 2, 69 Nr. 3 S. 2 UrhG angeordnete Erschöpfung des Verbreitungsrechts auch im Open Source Bereich uneingeschränkt gilt.[444] Hieraus folgt, dass Unternehmen Open Source Programme dann ohne Rücksicht auf die in der

[440] So zu Recht *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 487.
[441] Dies erörtern *Spindler / Wiebe*, CR 2003, 873, 877; ebenso in *Spindler*, Rechtsfragen bei Open Source, C. Rn. 97.
[442] BGH GRUR 2001, 51, 52 – Parfumflakon; BGH GRUR 1985, 131, 132 – Zeitschriftenauslage beim Friseur; *Loewenheim* in: Schricker, § 17 Rn. 36; *Schack*, Urheber- und Urhebervertragsrecht, Rn. 390 m.w.N.
[443] Es reicht eben gerade die *Möglichkeit* der wirtschaftlichen Partizipation, vgl. nur *Loewenheim* in: Schricker, § 17 Rn. 36.
[444] Eine Erschöpfung des Verbreitungsrechts vertreten: *Spindler / Wiebe*, CR 2003, 873, 876 f; *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 482 ff; *Jaeger / Metzger*, Open Source Software, Rn. 131; *Küng*, MR 2004, 21, 28 ff; *Schack*, Urheber- und Urhebervertragsrecht, Rn. 545 c.

GPL enthaltenen Vertriebsbedingungen vertreiben können, wenn sie diese selbst entsprechend den Voraussetzungen in § 69 Nr. 3 S. 2 UrhG z.B. in körperlicher Form von einem Distributor erworben haben.[445]

e.Anwendbarkeit des Erschöpfungsgrundsatzes auf unkörperlich Werkstücke

Neben der körperlichen Überlassung von Software ist gerade der Open Source Bereich dadurch gekennzeichnet, dass Programmkopien über das Internet online übermittelt werden. Diese Übermittlung von Programmkopien unterscheidet sich einzig dadurch von den „klassischen" Verbreitungsformen, dass statt des körperlichen ein unkörperliches Werkstück überlassen wird. Insofern stellt sich auch im Zusammenhang mit der unkörperlichen Überlassung von Werkstücken die Frage, ob hierdurch eine Erschöpfung des Verbreitungsrechts eintreten kann. Sofern diese Voraussetzungen gegeben wären, bestünde nämlich die Möglichkeit, dass sich das Verbreitungsrecht an GPL-lizenzierten Programmkopien bereits durch deren Download erschöpft und sich die heruntergeladenen Programmkopien somit weiterverbreiten lassen, ohne hierfür auf die Zustimmung des Urhebers angewiesen zu sein.

Die Unterscheidung zwischen der Verwertung eines Werks in körperlicher und in unkörperlicher Form lässt sich im Urheberrechtsgesetz an verschiedenen Stellen finden.[446] Im Hinblick auf das von § 17 UrhG geregelte Verbreitungsrecht besteht in der Rechtsprechung Einigkeit darüber, dass sich die hierin angeordnete Erschöpfung in Bezug auf Vervielfältigungsstücke ausschließlich auf körperliche Werkexemplare bezieht.[447] Insofern

[445] Die oftmals angeführte Gefahr für das gesamte Entwicklungs- und Vermarktungsmodell von Open Source Software (vgl. *Koch*, CR 2000, 333, 335 f; *Plaß*, GRUR 2002, 670, 680; *Schiffner*, Open Source Software, S. 142) folgt daraus jedoch nicht. Selbst wenn Zweiterwerber, entgegen den Bestimmungen der GPL, Open Source Software ohne Quellcode vertreiben, dürfte sich dies kaum in der befürchteten Art auswirken. Immerhin haben die Rechtsinhaber die Möglichkeit den Quellcode für jedermann zugänglich auf öffentlichen Servern zu hinterlegen, so dass interessierte Nutzer eigentlich nie in die Verlegenheit kommen sollten sich den dazugehörigen Quellcode zu beschaffen. Im Hinblick auf die ebenfalls immer wieder angeführte Gefahr, das Open Source Software nicht mehr kostenlos weitergegeben werden würde, kann nur festgestellt werden, dass die Versuche Programmkopien gegen Entgelt zu vertreiben aufgrund des auf dem Mark bestehenden Angebots zum scheitern verurteilt sind und mithin keine ernsthafte Gefahr zu vermitteln vermögen. Eingehend hierzu *Schulz*, Dezentrale Softwareentwicklungs- und Softwarevermarktungskonzepte, Rn. 483 Fn. 557; sowie *Jaeger / Metzger*, Open Source Software, Rn. 132 ff.

[446] Ausdrücklich ist dies in § 15 UrhG normiert, der in Abs. 1 die körperliche und in Abs. 2 die unkörperliche Verwertung betrifft.

[447] BGH GRUR 1995, 673, 676 – Mauerbilder; BGH GRUR 1986, 742, 743 – Videofilmvorführung; BGH GRUR 1954, 216, 219 – Schallplatten-Lautsprecherübertragung.

geht die h.M. davon aus, dass die in § 69 c Nr. 3 S. 2 UrhG geregelte Erschöpfungswirkung, durch die Bezugnahme auf den Begriff der „Vervielfältigungs*stücke*“, ebenfalls ausschließlich körperliche Werkstücke betrifft.[448] Die im Rahmen der Online-Übermittlung erfolgende unkörperliche Verbreitung wäre mithin nicht vom Erschöpfungsgrundsatz erfasst.

Diese Auslegung soll letztlich auch durch Art. 3 Abs. 3 und den Erwägungsgrund 29 der Richtlinie 2001/29/EG bestätigt werden.[449] Zwar betreffen die Bestimmungen der Richtlinie den § 69 c UrhG nicht unmittelbar,[450] jedoch lässt sich aus ihr das allgemeine Verständnis des europäischen Gesetzgebers im Hinblick auf die Reichweite des Erschöpfungsgrundsatzes erkennen, welches auch bereits der Richtlinie 91/250/EWG über den Rechtsschutz von Computerprogrammen[451] zugrunde lag. Der deutsche Gesetzgeber wollte bei der Umsetzung der Richtlinie 91/250/EWG von deren Bedeutungsgehalt nicht abweichen,[452] weshalb darauf geschlossen werden kann, dass die in § 69 c Nr. 3 S. 2 UrhG geregelte Erschöpfungswirkung, sich ausschließlich auf körperliche Werkstücke bezieht.

Probleme bereitet dieses Ergebnis vor dem Hintergrund, dass es technisch und wirtschaftlich keinen Unterschied macht, ob der Erwerber die Programmkopie offline auf einem Datenträger, oder online durch einen Download von einem Server übermittelt erhält.[453] Auch vor dem Hintergrund der Verwertungsinteressen des Urhebers lässt sich eine Differenzierung zwischen online und offline Vertrieb nicht rechtfertigen. Immerhin ist kein Grund dafür zu erkennen, dem Urheber der sein Werk online übermittelt und dessen Partizipationsinteressen – wie auch beim offline Vertrieb – im Rahmen dieser Veräußerung berücksichtigt wurden, weitergehende Verwertungsrechte zuzugestehen als demjenigen, der das gleiche Werk in körperlicher Form überlässt und dessen Verbreitungsrecht sich hierdurch erschöpft.[454]

---

[448] Vgl. insbesondere *Loewenheim* in: Schricker, § 69 c Rn. 25 sowie § 17 Rn. 4; *Koch*, CR 2002, 629, 631; *Bechtold*, ZUM 1997, 427, 431; *Waldenberger*, ZUM 1997, 176, 178; *Zahrnt*, CR 1994, 455, 457; eingehend zur Erschöpfung im Hinblick auf Open Source Software vgl. nur *Spindler / Wiebe*, CR 2003, 873, 876 ff.

[449] Vgl. Richtlinie 2001/29/EG zur Harmonisierung bestimmter Aspekte des Urheberrechts und der verwandten Schutzrechte in der Informationsgesellschaft, Abl. EG Nr. L 167, 12: *„Die Frage der Erschöpfung stellt sich weder bei Dienstleistungen allgemein noch bei Online-Diensten im Besonderen. Dies gilt auch für materielle Vervielfältigungsstücke eines Werks oder eines sonstigen Schutzgegenstands, die durch den Nutzer eines solchen Dienstes mit Zustimmung des Rechtsinhabers hergestellt worden sind. (...).“*.

[450] Vgl. Art. 1 Abs. 2 a und die Erwägungsgründe 20 und 50 der Richtlinie 2001/29/EG zur Harmonisierung bestimmter Aspekte des Urheberrechts und der verwandten Schutzrechte in der Informationsgesellschaft, Abl. EG Nr. L 167, 10 ff.

[451] Vgl. hierzu Abl. EG Nr. L 122 vom 17/05/1991 S. 42 – 46.

[452] Vgl. amtl. Begründung BT-Drucks. 12/4022, S. 8, wonach bewusst eine wörtliche Übernahme der Richtlinie erfolgte.

[453] Die identische Interessenlage ist maßgebliches Argument für eine entsprechende Anwendung der Erschöpfungswirkung auf Online-Übermittlungen bei *Hoeren* in: Möhring / Nicolini, § 69 c Rn. 12; *Marly*, Urheberrechtsschutz für Computersoftware, S. 242 ff; *Koch*, GRUR 1997, 417, 425 f; *Kotthoff*, GRUR 1997, 597, 600; *Mäger*, CR 1996, 522, 526.

[454] *Koch*, GRUR 1997, 417, 426.

Im Ergebnis hätte das von der h.M. geteilte Verständnis vom Erschöpfungsgrundsatz nämlich zur Folge, dass der Online-Vertrieb eine Möglichkeit darstellt, Werkexemplare zu veräußern und gleichwohl an der sonst freien Weiterverbreitung durch den Erwerber zu partizipieren. Ein Ergebnis, dass erheblichen Bedenken begegnet und daher auch von Teilen der Literatur durch eine Ausweitung des in § 69 c Nr. 3 S. 2 UrhG enthaltenen Erschöpfungsgrundsatzes auf unkörperliche Werkübermittlung vermieden wird.

Die hierfür erforderliche Ausweitung des Erschöpfungsgrundsatzes wird dabei unterschiedlich begründet. Zumeist wird mit der vergleichbaren Interessenlage oder auch mit Sinn und Zweck des Erschöpfungsgrundsatzes argumentiert und so eine analoge Anwendung des § 69 c Nr. 3 S. 2 UrhG ermöglicht.[455]

In der Rechtsprechung gab es bislang keine höchstrichterliche Entscheidung, bei der es zwingend auf die Erschöpfung bei Online-Übermittlung angekommen wäre. Allerdings sind in jüngerer Zeit Tendenzen zu erkennen, sich ggf. gegen eine Ausweitung des Erschöpfungsgrundsatzes auszusprechen.[456]

Dies vermag auch aus dogmatischer Sicht zu überzeugen, selbst wenn an der hierdurch erzeugten Rechtsfolge rechtspolitische Bedenken bestehen mögen. Gegen eine Ausweitung des Erschöpfungsgrundsatzes spricht zunächst, dass die Online-Übertragung aus systematischer Sicht keine Verbreitung im Sinne des § 69 c Nr. 3 UrhG darstellt.[457] Der Erwerber erhält bei einer Online-Übermittlung eben keine Programmkopie, sondern stellt sich diese, im Gegensatz zur körperlichen Übermittlung, selber her.[458]

Zudem deutet sowohl die Entstehungsgeschichte, als auch eine richtlinienkonforme Auslegung des § 69 c Nr. 3 S. 2 UrhG darauf hin, den Anwendungsbereich des Erschöpfungsgrundsatzes auf die körperliche Übermittlung zu beschränken. Dem Gesetzgeber waren im Rahmen des Gesetzgebungsverfahrens zu den §§ 69 a ff UrhG, durchaus die unterschiedlichen Übermittlungsmöglichkeiten, insbesondere die Unterscheidung zwischen

[455] Vgl. *Hoeren* in: Möhring / Nicolini, § 69 c Rn. 16; *Grützmacher* in: Wandtke / Bullinger, § 69 c Rn. 31; wohl auch *Dreier* in: Dreier / Schulze, § 69 c Rn. 24; *Royla / Gramer*, CR 2005, 154, 155; *Bartsch*, CR 1987, 8, 11 f; *Küng*, MR 2004, 21, 28 ff; eine erweiternde Auslegung für ausreichend halten hingegen *Koch*, GRUR 1997, 417, 426; *Pres*, Gestaltungsformen urheberrechtlicher Softwarelizenzverträge, S. 116; *Nordemann / Vinck* in: Nordemann / Fromm, § 69 c Rn. 6.

[456] In diese Richtung könnte der Hinweis des LG München MMR 2006, 175, 177 (bestätigt durch OLG München MMR 2006, 748) verstanden werden, in dem von diesem abgelehnt wird „den Erschöpfungsgrundsatz über seinen Eigentlichen Anwendungsbereich beim Vertrieb von körperlichen Vervielfältigungsstücken hinaus auf Handlungen auszudehnen, mit denen eine Vervielfältigung verbunden ist“. A.A. LG Hamburg Urt. v. 29.06.2006, Geschäfts-Nr. 315 O 343/06.

[457] Vgl. nur *Loewenheim* in: Schricker, § 69 c Rn. 25, 33 m.w.N.

[458] Insofern unterfällt die Online-Übermittlung dem § 69 c Nr. 4 UrhG, vgl. *Loewenheim* in: Schricker, § 69 c Rn. 30.

körperlicher und unkörperlicher Übermittlung, bekannt. Demzufolge liegt mit § 69 c Nr. 3 UrhG eine bewusste Entscheidung zur Regelung der Erschöpfungswirkung für die Fälle der körperlichen Übermittlung vor, worin aber zugleich auch eine Aussage zur unkörperlichen Übermittlung liegt. Die für eine analoge Auslegung des § 69 c UrhG erforderliche planwidrige Regelungslücke ist daher nicht gegeben.[459]

Im Ergebnis kann die Erschöpfung des Verbreitungsrechts gem. § 69 c Nr. 3 S. 2 UrhG somit nur in den Fällen der körperlichen Überlassung von Programmkopien, nicht jedoch bei deren Online-Übermittlung eintreten. Hieraus folgt, dass Unternehmen, die Open Source Software per Download erworben haben, diese nur nach vorheriger Zustimmung des Urhebers bzw. nach Abschluss der GPL an Kunden weiterveräußern dürfen.

Ergebnis:

Zusammenfassend lassen sich sowohl im Bereich der Entwicklung von Programmkomponenten, als auch bei deren Vertrieb einige typische Fallgestaltungen festhalten, in denen eine Zustimmung des Urhebers gem. § 69 c UrhG und somit auch der Abschluss der GPL erforderlich wird.

Für Unternehmen, die Entwicklungen im Umfeld von Open Source Software vornehmen ist dabei zunächst zu berücksichtigen, dass in jeder Veränderung von Open Source Software eine zustimmungsbedürftige Handlung liegt, bei deren Vornahme von einem konkludenten Vertragsschluss durch den Bearbeiter ausgegangen würde. Eine Pflicht zur Offenlegung des Quellcodes geht damit allerdings nicht zwangsläufig einher, da diese gem. Ziff. 2 GPL erst dann eingreift, wenn solche Veränderungen vervielfältigt oder vertrieben werden sollen.[460] Hierdurch besteht für Unternehmen die Möglichkeit den Copyleft Effekt dadurch zu umgehen, dass der Vertrieb der eigenen Programmkomponenten vor die eigentliche Bearbeitung gesetzt wird und sie somit erst nach dem Vertrieb im Rahmen der Installation erfolgt. Anstatt die beabsichtigten Veränderungen an GPL-lizenzierten Programmen bereits firmenintern durchzuführen und erst das Arbeitsergebnis als (lizenzrechtlich relevante) Koppelung von GPL-Code und eigenem Code zu vertreiben, könnten sich die Unternehmen darauf beschränken, ausschließlich die eigenen Werkteile weiterzugeben und den Kunden im

[459] Zur a.A. vgl. nur *Grützmacher* in: Wandtke / Bullinger, § 69 c Rn. 31 m.w.N.
[460] Ziff. 2 GPL: *"You may modify your copy or copies of the Program ... and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions*

Übrigen darauf zu verweisen sich die GPL-lizenzierten Programme aus öffentlich zugänglichen Quellen zu besorgen.[461] Da der Vertrieb der eigenen Werkteile keiner Beschränkung unterliegt, wäre eine Zustimmung im Sinne von § 69 c Nr. 2 UrhG erst dann erforderlich, wenn die maßgeblichen Veränderungen an den GPL-lizenzierten Programmen zum Installationszeitpunkt vorgenommen werden.[462]

Auf diese Weise könnten selbst solche Werkteile vertrieben werden, die bei einem gemeinsamen Vertrieb zusammen mit GPL-lizenzierten Programmteilen eindeutig als *„derivative work“* anzusehen wären. Indem Unternehmen den Vertrieb GPL-lizenzierter Programmteile vermeiden, besteht für sie die Möglichkeit, Modifikationen und Weiterentwicklungen an GPL-lizenzierter Software durchzuführen, ohne gleichzeitig dazu verpflichtet zu sein, den Quellcode der eigenen Entwicklungen offen zu legen.
Ein solches Vorgehen widerspricht zwar offensichtlich den mit dem Open Source Modell verfolgten Zielen, lässt sich aber wohl anhand der derzeitigen Regelungen nicht vermeiden.[463] Immerhin fehlt es zum Zeitpunkt des Vertriebs der Eigenentwicklung noch an einer

[461] So im Ergebnis auch *Spindler*, Rechtsfragen bei Open Source, C. Rn. 121; *Determann*, GRUR Int. 2006, 645, 649; sowie *Jaeger* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 2 GPL Rn. 18, der hierin allerdings mitunter eine rechtlich angreifbare Umgehung sehen will; a.A. wohl die FSF, die auf ihrer FAQ-Seite darauf verweist, dass Erweiterungsprogramme nur unter der GPL vertrieben werden dürfen, vgl. http://www.fsf.org/licensing/licenses/gpl-faq.html#GPLAndPlugins.
Dass von dieser „*Umgehungs*“-Möglichkeit derzeit kaum Gebrauch gemacht wird, liegt maßgeblich daran, dass die Unternehmen den Kunden zumeist dadurch entgegenkommen wollen, dass sie ihnen sämtliche Programmkomponenten auf einem Datenträger überlassen.

[462] Zu diesem Zeitpunkt knüpft sich sodann die Frage an, wer als Bearbeiter anzusehen und mithin auf die Zustimmung des Urhebers angewiesen ist. Zweifel daran, dass ausschließlich das Unternehmen die Bearbeitung im Sinne von § 69 c Nr. 2 UrhG vornimmt, könnten dadurch geboten sein, dass die zustimmungsbedürftige Bearbeitung am GPL-lizenzierten Programmteil erst im Rahmen der Installation des proprietären Programms erfolgt, welche ihrerseits jedoch im Machtbereich des Kunden liegt und auch ausschließlich von diesem ausgeführt wird. Dadurch könnte die maßgebliche Bearbeitungshandlung nicht dem Unternehmen, sondern dem Kunden zuzurechnen sein. Hiergegen lässt sich indes einwenden, dass dem Kunden keinerlei Handlungsspielraum zusteht, sondern er für eine bestimmungsgemäße Programmbenutzung zwingend auf die im proprietären Programm begründete Veränderung des Open Source Teils angewiesen ist. Dies spricht argumentativ eher dafür, die Installationshandlung dem Unternehmen, nicht aber dem Kunden zuzurechnen, unabhängig davon, dass dieser den letzten Schritt auf dem Weg zur Veränderung des GPL-lizenzierten Programms anstößt.

[463] In der Literatur wird teilweise gleichwohl davon ausgegangen, dass die Rechtsprechung diesen Umgehungsmöglichkeiten künftig Grenzen setzen wird, vgl. *Jaeger* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 2 GPL Rn. 18. Hieran bestehen indes Zweifel, immerhin erscheint eine solche Begrenzung durch die Rechtsprechung nur im Hinblick auf § 242 BGB denkbar. Dafür müsste die Umgehung des Copyleft-Effekts zu untragbaren mit Recht und Gerechtigkeit offensichtlich unvereinbaren Ergebnissen führen (vgl. *Heinrichs* in: Palandt, § 242 Rn. 2). Ein solches Missverhältnis könnte die beschriebene Umgehung des Copyleft-Effekts allerdings nur dann verursachen, wenn feststehen würde, dass ein Unternehmen seinen Vertrieb einzig aus dem Grund auf die eigenen Entwicklungen beschränkt hat, um dadurch den Copyleft-Effekt zu vermeiden. Dies wird sich in der Praxis allerdings kaum einmal mit Gewissheit feststellen lassen, da die Unternehmen gerade auch aus haftungs- und gewährleistungsrechtlichen Gründen ein erhebliches Interesse daran haben, ausschließlich eigenen Produktcode, nicht aber auch „Fremdentwicklungen“ zu vertreiben. Hinzu kommt, dass eine Einschränkung im Rahmen des § 242 BGB auch im Zusammenhang mit der Parteiautonomie bedenklich erscheint. Immerhin handelt es sich bei dem in der GPL enthaltenen Copyleft-Effekt lediglich um eine Vertragsregelung, deren gerichtliche Ausweitung einem Abschlusszwang gleichkommen würde.

zustimmungsbedürftigen Handlung der Unternehmen, so dass die GPL keine Bindungswirkung entfalten kann.

Sobald Unternehmen damit beginnen GPL-lizenzierte Programme weiterzugeben, müssen sie grundsätzlich davon ausgehen, dass sie auf den Abschluss der GPL angewiesen sind. Sofern neben den GPL-lizenzierten Programmen zudem auch eigene Entwicklungen überlassen werden, sind diese daher auf die Einhaltung der in der GPL statuierten Pflichten, insbesondere auch im Hinblick auf die in Ziff. 2 Abs. 1 b GPL enthaltene Pflicht zur Offenlegung des Quellcodes zu untersuchen. Eine Ausnahme hiervon kommt lediglich in den Fällen in Betracht, in denen die GPL-lizenzierten Programmteile ausschließlich zwischen persönlich verbundenen Personen zugänglich gemacht werden.

II.Reichweite der Lizenzierungspflicht

Nachdem zuvor diejenigen Fälle herausgearbeitet wurden, in denen eine Lizenzierungspflicht bereits deshalb nicht in Betracht kommt, weil der beabsichtigte Vertrieb keinen Abschluss der GPL erfordert, sind nun diejenigen Fallkonstellationen zu untersuchen, in denen sich die GPL auswirkt.

Dies ist zunächst immer dann der Fall, wenn der Nutzer bereits beim Programmerwerb zur Annahme der GPL verpflichtet wird oder aber von einer konkludenten Annahme der GPL auszugehen ist, weil der Nutzer zustimmungsbedürftige Handlungen im Sinne von § 69 c UrhG vorgenommen hat, also insbesondere das Programm bearbeitet oder verbreitet.[464]

In diesen Fällen stellt sich die Frage, auf welche Teile die in der GPL enthaltene Lizenzierungspflicht zu erstrecken ist. Die GPL sieht vor, dass bei einer, der Veränderung der Programmkopie nachfolgenden, Vervielfältigung und bei einem Vertrieb des Werks die Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL einzuhalten sind.

Hierbei ist vor allem die in Ziff. 2 Abs. 1 b GPL enthaltene Lizenzierungspflicht problematisch, wonach sämtliche Werke, die von deren Anwendungsbereich erfasst werden, unter der GPL zu lizenzieren sind. Damit geht letztlich eine Offenlegung des Quellcodes einher, weshalb die Reichweite dieser Regelung für Unternehmen von erheblicher Relevanz ist.

Ziff. 2 Abs. 1 b) GPL: *"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."*

Im Hinblick auf die Reichweite der Lizenzierungspflicht ist Ziff. 2 Abs. 1 b GPL zu entnehmen, dass hiervon das Werk „als Ganzes" erfasst wird (*„You must cause any work (...)to be licensed* ***as a whole****..."*). Was aber ist das Werk „als Ganzes"? Welche Werkteile umfasst Ziff. 2 Abs. 1 b GPL? Sollen hierunter nur die tatsächlich veränderten Teile fallen, oder erstreckt sich die Lizenzierungspflicht mitunter auch auf weitere Werke?

Für eine Begrenzung der Lizenzierungsplicht auf die tatsächlich veränderten Werke scheint zunächst Ziff. 2 Abs. 3 GPL zu sprechen, wonach es nicht beabsichtigt ist, Rechte für Datenwerke in Anspruch zu nehmen oder Rechte für Datenwerke streitig zu machen, die das

[464] Vgl. bereits S. 100 ff.

Unternehmen komplett eigenständig geschrieben hat. Eine ähnliche Wertung findet sich auch in Ziff. 2 Abs. 4 GPL, wonach das einfache Zusammenlegen mehrerer Werke auf einem Speichermedium keine Ausweitung der Lizenzierungspflicht auf die jeweils anderen Werke[465] nach sich zieht.

Insofern könnte die von Ziff. 2 Abs. 1 b GPL vermittelte Lizenzierungspflicht ausschließlich das veränderte Programm, nicht aber hiervon zu trennende weitere eigenständige Werke, betreffen. Die Reichweite der Lizenzierungspflicht würde somit durch den Umfang des Programms eine Begrenzung erfahren. Sofern die Lizenzierungspflicht hinsichtlich einzelner Dateien zweifelhaft wäre, müsste lediglich untersucht werden, ob sie dem veränderten Programm, oder einem anderen eigenständigen Werk zuzurechnen sind.

Hiergegen spricht indes die Regelung in Ziff. 2 Abs. 2 S. 3 GPL, wonach sich die Reichweite der Lizenzierungspflicht auch auf unabhängige und eigenständige Werke erstrecken soll, sofern diese als Teil eines Ganzes vertrieben werden.[466]

Dies überrascht deshalb, weil die Lizenzierungspflicht hier nicht an die funktionale Beziehung der Programme untereinander anknüpft, sondern daran in welcher Form diese vertrieben werden. Abhängig von den formalen Vertriebsvoraussetzungen, können also auch Programme, die in funktionaler Hinsicht als unabhängig und eigenständig zu betrachten sind, als einheitliches Ganzes im Sinne von Ziff. 2 Abs. 1 b GPL anzusehen und somit insgesamt unter der GPL zu lizenzieren sein.

Hieraus folgt, dass es im Hinblick auf die Frage nach der Reichweite der Lizenzierungspflicht nicht ausreicht, mit Verweis auf Ziff. 2 Abs. 4 GPL festzustellen, dass neben dem veränderten Programm weitere unabhängige und eigenständige Werke vorhanden sind. Sofern funktional eigenständige und unabhängige Werke bestehen, ist zwar sowohl eine Anwendbarkeit von Ziff. 2 Abs. 4 GPL als auch eine Ausnahme im Sinne von Ziff. 2 Abs. 2 S. 2 GPL möglich, allerdings hängt dies einzig von der Form des Vertriebs ab.

Im Hinblick auf die Frage nach der Reichweite der Lizenzierungspflicht ist somit festzuhalten, dass allein aus dem Vorliegen verschiedener unabhängiger und eigenständiger Werke kein zwingender Rückschluss auf die Reichweite der Lizenzierungspflicht gezogen werden kann.

Andererseits lässt sich die Frage nach der Reichweite der Lizenzierungspflicht dann eindeutig beantworten, wenn es an unabhängigen und eigenständigen Werkteilen im Sinne von Ziff. 2 Abs. 2 S. 2 GPL fehlt. In diesen Fällen, steht nämlich fest, dass sämtliche Werkteile in einer

[465] Ziff. 2 Abs. 4 GPL: *„…mere aggregation of **another work** …“*

[466] Ziff. 2 Abs. 2 S. 3 GPL: *“Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you…”*

funktionalen Verbindung zueinander stehen und sich mithin einem einheitlichen Werk zurechnen lassen. Die in Ziff. 2 Abs. 1 b GPL enthaltene Lizenzierungspflicht würde sich daher auf alle vorhandenen Werkteile beziehen.

Für die Frage nach der Reichweite der Lizenzierungspflicht bietet es sich infolgedessen an, zunächst darauf abzustellen, ob unabhängige und eigenständige Werke auszumachen sind. Fehlt es hieran, steht fest, dass weder die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL in Betracht kommt noch ein Fall der Ziff. 2 Abs. 4 GPL vorliegen kann. Die Lizenzierungspflicht erstreckt sich folglich auf sämtliche Programmteile, die gemeinsam vertrieben werden sollen.
Liegen hingegen unabhängige und eigenständige Werke im Sinne von Ziff. 2 Abs. 2 S. 2 GPL vor, so kommt es maßgeblich auf die Form an, in der diese weitergegeben werden sollen.
Werden die Werke als einheitliches Ganzes vertrieben (*„ …as part of a whole…"*), so betrifft die Lizenzierungspflicht sämtliche Teile. Werden die Werke dagegen als eigenständige Teile vertrieben (*„ … as separate works…"*), so liegt die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL und – sofern der Vertrieb auf einem Speichermedium erfolgt – zugleich ein Fall von Ziff. 2 Abs. 4 GPL vor, so dass sich die Lizenzierungspflicht einzig auf das veränderte, nicht aber auf die daneben bestehenden eigenständigen Werke bezieht.

Stellt sich die Frage nach der Lizenzierungspflicht, so ist folglich dreistufig vorzugehen. Auf einer ersten Stufe ist festzustellen, ob der Anwendungsbereich der Ziff. 2 GPL eröffnet, es also zu Veränderungen der ursprünglichen Programmkopie oder Teilen davon gekommen ist.[467]
Auf einer zweiten Stufe ist sodann zu überprüfen, ob unter den vorhandenen Dateien, deren Weitergabe beabsichtigt wird, Werkteile vorhanden sind, die funktional unabhängig und eigenständig gegenüber der ursprünglichen Programmkopie sind.[468]
Sofern mehrere eigenständige Werke ausgemacht werden können, ist vor dem Hintergrund von Ziff. 2 Abs. 2 S. 2 und Ziff. 2 Abs. 4 GPL auf einer dritten Stufe die Vertriebsform daraufhin zu überprüfen, ob diese getrennt oder als einheitliches Ganzes erfolgt.[469]

[467] Hierzu sogleich ab S. 118.
[468] Vgl. S. 135 ff.
[469] Ab S. 165.

1.Stufe 1 – Anwendungsbereich von Ziff. 2 GPL eröffnet?

Maßgebliche Regelung für die Lizenzierungspflicht ist Ziff. 2 Abs. 1 b GPL, wonach das Werk als Ganzes unter der GPL zur Verfügung zu stellen ist, sofern die darin enthaltenen Voraussetzungen erfüllt sind. Die Anwendung von Ziff. 2 Abs. 1 b GPL setzt ihrerseits voraus, dass der Anwendungsbereich der Ziff. 2 GPL eröffnet ist.

Ziff. 2 Abs. 1 GPL: "*You may modify your copy or copies of the Program or any portion of it (...) and copy and distribute such modifications (...), provided that you also meet all of these conditions:"*

Eine erste Voraussetzung ist hiernach, dass es zu Veränderungen an der GPL-lizenzierten Programmkopie oder Teilen hiervon gekommen sein muss.[470]

Dementsprechend stellt sich zunächst die Frage, wann von einer solchen Veränderung der Programmkopie auszugehen ist. Diese Frage stellt sich, da aus der GPL nicht hervorgeht, was für eine Veränderung vorausgesetzt wird. Erfordert diese zwangsläufig eine Änderung des Quelltextes, oder reicht es schon aus, wenn ein GPL-lizenzierter Quelltext in kompilierter Form vertrieben werden soll?

Der Wortlaut der GPL enthält hierzu zwar keine direkte Antwort, allerdings könnte Ziff. 2 Abs. 1 2. Hs. GPL ein Hinweis zu entnehmen sein.

Ziff. 2 Abs. 1 GPL: "*You may modify your copy or copies of the Program or any portion of it,* ***thus forming a work based on the Program****…".*

Aus dem 2. Halbsatz von Ziff. 2 Abs. 1 GPL wird ersichtlich, dass die ursprüngliche Programmkopie oder Teile derselben in einer Art und Weise verändert werden müssen, dass als Ergebnis ein *„work based on the Program"* entsteht. Dieser Hinweis ist insofern wertvoll, als der Begriff des *„work based on the Program"* in Ziff. 0 GPL einem *„derivative work under copyright law"* gleichgestellt wird.[471] Daraus lässt sich folgern, dass mit Ziff. 2 GPL

[470] Zur Frage, ob auch solche Fälle erfasst werden, in denen die Programmkopie unverändert bleibt siehe unten S. 132.

[471] Ziff. 0: "*…and a "work based on the Program" means either the Program or any derivative work under copyright law…".*

solche Veränderungen erfasst werden sollen, die zu einem *„derivative work under copyright law“* führen.

Hieran knüpft die Frage an, was unter einem solchen *„derivative work under copyright law“* zu verstehen ist. In der deutschen Literatur wird vielfach darauf abgestellt, dass dies mit dem deutschen Begriff der Bearbeitung zu vergleichen sei,[472] weshalb im Rahmen dieser Untersuchungen dann zumeist auch die deutschen urheberrechtlichen Begrifflichkeiten zugrunde gelegt werden.
Ein solches Vorgehen ist allerdings zu beanstanden, sofern auf das deutsche urheberrechtliche Verständnis zur Bearbeitung zurückgegriffen wird, ohne zuvor zu berücksichtigen, dass es sich bei der GPL um einen Lizenzvertrag handelt, der vor dem Hintergrund des US-Rechts erstellt wurde und in Ziff. 0 GPL auch ausdrücklich auf das Copyright Law Bezug nimmt. Insofern ist vor einer Heranziehung der entsprechenden deutschen Begrifflichkeiten zunächst die Frage zu beantworten, vor welchem Recht der Begriff des *„derivative work“* auszulegen ist.
Damit ist allerdings nicht die Frage nach dem anwendbaren Recht gemeint, welche bereits Gegenstand der Untersuchung war.[473] Vielmehr geht es um die hiervon zu trennende Frage, wie Vertragsbegriffe auszulegen sind, die sich auf eine andere Rechtsordnung beziehen.
Art. 32 Abs. 1 Nr. 1 EGBGB liefert hierzu keine Lösung, da dieser zwar die deutschen Auslegungsregeln der §§ 133, 157 BGB für anwendbar erklärt, aber keinen direkten Einfluss auf die im Vertrag enthaltenen Begrifflichkeiten hat.[474]
Die Rechtsprechung geht daher bei der Auslegung von AGB[475] so vor, dass sie die Begriffe anhand ihres objektiven Sinns in den entsprechenden Verkehrskreisen auslegt.[476] Im Zusammenhang mit juristischen Fachtermini entspricht es der h.M., deren jeweiligen juristische Bedeutung entsprechend zu berücksichtigen.[477] Im vorliegenden Fall ist daher bei der Auslegung des Begriffs des *„derivative work“* auch die im US-Recht enthaltene Bedeutung zu beachten.

---

[472] Vgl. u.a. *Spindler*, Rechtsfragen der Open Source Software, S. S. 61, www.vsi.de/inhalte/aktuell/studie_final_safe.pdf; *Wuermeling / Deike*, CR 2003, 87, 88 f; *Lejeune*, ITRB 2003, 10, 11; a.A. wohl *Stickelbrock*, ZGS 2003, 368, 370.
[473] Vgl. Kapitel 4 S. 64 ff.
[474] Eingehend zur Auslegung ausländischer Vertragsbegriffe *Triebel / Balthasar*, NJW 2004, 2189, 2192 ff; sowie *Gruber,* DZWir 1997, 353, 354 ff.
[475] Bei der GPL handelt sich nach allgemeiner Auffassung um AGB, vgl. LG Frankfurt/M CR 2006, 729, 731; LG München MMR 2004, 693, 697; aus der Literatur vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 179 m.w.N. ; sowie die Nachweise oben S. 40, Fn. 164.
[476] BGHZ, 33, 216, 218; 49, 84, 88; 77, 116, 118; eingehend Dreher AcP 189 (1989), 342 ff.
[477] Vgl. *Spellenberg* in: Müko, Vorbem. Zu Art. 11 Rn. 111.

Unter einem *„derivative work“* wird nach amerikanischem Verständnis gem. 17 U.S.C. § 101 Folgendes verstanden:
"a work based upon one or more preexisting works, such as a translation, (...), or any other form in which a work may be recast, transformed, or adapted. A work consisting of editorial revisions, annotations, elaborations, or other modifications which, as a whole, represent an original work of authorship, is a “derivative work”."

Ähnlich wie im deutschen Recht, stellt auch das US-Recht für das Vorliegen eines *„derivative work“* darauf ab, dass dieses auf einem oder mehreren vorbestehenden Werken beruht. Weiterhin wird auch nach amerikanischem Rechtsverständnis vorausgesetzt, dass das vorbestehende Werk eigenständig urheberrechtlich schutzwürdig sein muss.[478] Zudem reicht ferner die Übernahme beliebiger Werkteile nicht aus, um ein *„derivative work“* zu erstellen. Vielmehr müssen wesentliche Teile vom vorbestehenden Werk übernommen worden sein.[479] Insgesamt entsprechen die Voraussetzungen, unter denen von einem *„derivative work“* auszugehen ist, somit im Wesentlichen denen einer Bearbeitung im Sinne von § 3 UrhG, weshalb auf deren Voraussetzungen bei einer Auslegung der GPL vor dem Deutschen Recht zurückgegriffen werden kann.[480]

Zwischenergebnis:

Somit lassen sich bis zu dieser Stelle bereits zwei wesentliche Bedingungen für die Anwendbarkeit von Ziff. 2 GPL festhalten.
Zum Einen muss es zu einer Veränderung an der GPL-lizenzierten Programmkopie oder Teilen hiervon gekommen sein. Zum Anderen muss diese Veränderung in einer Art und Weise stattgefunden haben, dass sich das hieraus resultierende Werk als *„derivative work“* darstellt, was im Deutschen Recht einer Bearbeitung im Sinne von § 3 UrhG entspricht.

---

[478] Vgl. Ets-Hokin v. Skyy Spirits, Inc., C.A.9 (Cal.) 2000, 225 F.3d 1068, 55 U.S.P.Q.2d 1769: “*Under the Copyright Act, a work is not a "derivative work" unless it is based upon one or more preexisting works and, in order to qualify as a preexisting work, the underlying work must be copyrightable.*”
[479] Vgl. M.H. Segan Ltd. Partnership v. Hasbro, Inc., S.D.N.Y.1996, 924 F.Supp. 512, 39 U.S.P.Q.2d 1619: “*Not just any work that is based on preexisting work is derivative work entitled to copyright protection; derivative work must borrow substantially from existing works…”;* Litchfield v. Spielberg, 736 F.2d 1352 (9th Cir. 1984): “*holding that a work is not derivative unless it has been substantially copied from a previous work, and not merely “vaguely connected” or based on a preexisting work…”*; Apple Computer, Inc. v. Microsoft Corp., 779 F. Supp. 133, 20 U.S.P.Q.2d (BNA) 1236 (N.D. Cal. 1991), related reference, 1992 Copr. L. Dec. P 26903, 1992 WL 75423 (N.D. Cal. 1992) and aff'd, 35 F.3d 1435, 32 U.S.P.Q.2d (BNA) 1086 (9th Cir. 1994); ebenso *Determann*, GRUR Int. 2006, 645, 646..
[480] Ebenso *Determann*, GRUR Int. 2006, 645, 650 f; *Wuermeling / Deike*, CR 2003, 87, 88 f.

Hieraus folgt letztlich eine Eingrenzung des Anwendungsbereichs von Ziff. 2 Abs. 1 GPL, da nicht sämtliche Veränderungen an der Programmkopie oder Teilen hiervon zu einer Anwendbarkeit von Ziff. 2 Abs. 1 GPL führen.[481] Es müssen vielmehr urheberrechtlich schutzwürdige Programme oder Programmteile verändert oder wesentliche Teile eines solchen Werks übernommen werden.

Während die Frage nach der urheberrechtlichen Schutzfähigkeit des ursprünglichen Programms in jedem Einzelfall gesondert zu untersuchen ist, kann hinsichtlich der Übernahme wesentlicher Programmteile eine Kategorisierung möglicher Fallgruppen danach erfolgen, in welchen Fallgestaltungen die Übernahme und die Veränderung von Programmteilen überhaupt relevant werden können.
Hierfür ist es erforderlich zunächst die verschiedenen „Aggregatzustände" und Arbeitsschritte näher zu betrachten, die im Rahmen einer Softwareentwicklung durchlaufen werden, da diese Aufschluss darüber geben, wann die (teilweise) Einbeziehung von Programmen sowie deren Veränderung überhaupt relevant werden können.

a.Softwareentwicklungsprozess

Ausgangspunkt der modernen Programmentwicklung ist stets der Quelltext. Damit dieser letztlich von einem Computer ausgeführt werden kann, muss er zuvor in ein ausführbares Programm übersetzt werden. Auf dem Weg zum ausführbaren Programm durchläuft der Quelltext dabei verschiedene „Aggregatzustände", deren Anzahl abhängig von der verwendeten Programmiersprache ein wenig variieren kann.[482] Im Zusammenhang mit Linux werden die meisten Programme in C oder C++ geschrieben, weshalb auf die Programmentwicklung mit Hilfe dieser Sprachen exemplarisch eingegangen wird.
Um aus einer in C geschriebenen Quelltextdatei ein ausführbares Programm zu erstellen, ist dessen Übersetzung in eine maschinenlesbare Form erforderlich. Diese vollzieht sich im Wesentlichen in drei Arbeitsschritten, die gemeinhin als Kompilierung bezeichnet werden.
Zunächst wird der Quelltext durch den sog. Präprozessor bearbeitet. Hierbei werden entsprechend der sog. Präprozessordirektiven (bspw. # include und # define) textuelle

[481] Ähnlich auch *Determann*, GRUR Int. 2006, 645, 649 ff.
[482] Dies liegt daran, dass nicht alle Programmiersprachen zur Ausführung des Programms dessen vorhergehende Kompilierung voraussetzen, sondern häufig erst zum Zeitpunkt der Ausführung durch einen sog. Interpreter in maschinenlesbaren Code übersetzt werden. Beispiele solcher Interpretersprachen sind BASIC, Perl, Python, Ruby und PHP.

Ersetzungen am Quellcode durchgeführt. Der Arbeitsschritt des Präprozessors lässt sich am Besten mit der „Suchen und Ersetzten“ Funktion vergleichen, die auch in den meisten Schreibprogrammen vorhanden ist. Im Hinblick auf die Übersetzung des Programms, bewirkt dieser Arbeitsschritt allerdings keinen Fortschritt, da der Quellcode danach zwar um die entsprechenden Textstellen expandiert, jedoch in sprachlicher Hinsicht unverändert in der ursprünglichen Programmiersprache vorliegt.

Die eigentliche Übersetzung wird erst im nächsten Schritt bewirkt, indem der expandierte Quelltext durch den Compiler in sog. Objektcode übersetzt wird.[483] In dieser Form ist die Datei zwar schon in einer maschinenlesbaren Form, kann aber in der Regel noch nicht vom Computer ausgeführt werden, da sie nicht sämtlichen Code beinhaltet. Dies liegt daran, dass in der modernen Programmierung ein besonderer Wert auf der Modularisierung von Programmen liegt, mithin diejenigen Teile in andere Dateien ausgelagert werden, die von mehreren Programmteilen benötigt werden.[484] Da ein Programm aber erst dann ausgeführt werden kann, wenn der Computer auf sämtlichen Code zugreifen kann, muss der Quellcode noch an denjenigen Stellen verändert werden, an denen auf Code anderer Dateien Bezug genommen wird (sog. Referenzen).

Die Auflösung solcher Referenzen wird im dritten Arbeitsschritt durch den sog. Linker bewirkt. Trifft dieser innerhalb des Codes auf Referenzen, so bestehen im Wesentlichen zwei Möglichkeiten diese aufzulösen. Entweder wird der referenzierte Code direkt in den Quellcode integriert (sog. statisches Linken) oder es wird lediglich ein Verweis auf das Verzeichnis eingefügt, an dem sich der zugehörige Code befindet (sog. dynamisches Linken). Der Code wird im letzteren Fall nicht direkt in die Datei integriert, sondern vom Computer erst geladen, wenn dieser im Rahmen der Programmausführung an derjenigen Stelle innerhalb des Programms angekommen ist, an welcher der betreffende Code benötigt wird.

[483] Dies ist etwas unpräzise, da der Compiler eigentlich nur die Assemblerdatei und erst der Assembler hieraus erst die Objektdatei erstellt. Vorliegend reicht diese Untergliederung indes aus, da am Code in aller Regel keine Änderungen mehr zwischen dem Arbeitsschritt des Compilers und dem des Assemblers vorgenommen werden.

[484] Klassisches Beispiel der Modularisierung sind die Programmbibliotheken, in denen häufig verwendete Befehle für alle Programmmodule zugänglich gemacht werden, so dass diese in den einzelnen Module nicht erneut enthalten sein müssen, sondern ein Aufruf der entsprechenden Funktion in der Programmbibliothek ausreicht.

b.GPL-Programm lag im Quelltext vor

Ausgehend davon, dass sich der Open Source Bereich gerade dadurch auszeichnet, dass die Quelltexte der Programme vorhanden sind und sie sich in dieser Form besonders leicht modifizieren und einbinden lassen, sind zunächst die Fälle zu untersuchen, in denen die Sourcen des Programms, also deren Quelltexte erworben wurden. Fraglich ist in diesen Fällen, ab wann von einer Veränderung im Sinne von Ziff. 2 GPL auszugehen ist.

c.GPL-Quelltextdatei wird unmittelbar verändert

Einfachster Fall, in dem Veränderungen an der ursprünglichen Programmkopie vorliegen, ist der, in dem Quelltextdateien der Programmkopie direkt verändert werden. Hat die ursprüngliche Programmkopie über eine ausreichende Schöpfungshöhe verfügt, führen sämtliche dieser Veränderungen zu einem *„work based on the Program“* und somit zu einem *„derivative work“*, weshalb der Anwendungsbereich von Ziff. 2 Abs. 1 GPL eröffnet ist. Der Vertrieb eines solchen Werks dürfte nur unter den zusätzlichen Bedingungen der Ziff. 2 Abs. 1 a-c GPL erfolgen.

d.GPL-Quelltextdatei oder Teile davon werden verwendet

Eine Veränderung der GPL-lizenzierten Programmkopie im Sinne von Ziff. 2 GPL liegt auch dann vor, wenn die Quelltextdatei oder Teile davon in andere Quelltextdateien übernommen werden, indem der GPL-lizenzierte Quelltext insgesamt oder auch in Auszügen kopiert und in andere Quelltextdateien eingefügt wird.[485] Die hieraus entstehende neue Quelltextdatei ist ebenfalls als Veränderung der ursprünglichen Programmkopie zu werten, da es letztlich keinen Unterschied macht, ob Teile des Quelltextes in eine andere Datei übernommen werden, oder ob die ursprüngliche Datei bis auf diesen Teil gekürzt und erst dann um den neuen Quelltext erweitert wird.

Auch in diesen Fällen können die Vervielfältigung und der Vertrieb solcher Quelltextdateien ausschließlich unter den zusätzlichen Bedingungen der Ziff. 2 Abs. 1 a-c GPL erfolgen, sofern die ursprüngliche Programmkopie urheberrechtlich schutzwürdig gewesen ist.

[485] Vgl. nur *Wuermeling / Deike*, CR 2003, 87, 89; *Andréewitch*, MR 2005, 240, 241.

Im Zusammenhang mit der Übernahme von GPL-lizenziertem Quelltext sind neben dem manuellen Kopieren von Quelltextdateien, auch solche Fallgruppen interessant, in denen die Übernahme von Quelltext automatisch erfolgt. In diesem Zusammenhang kommt sowohl der Erstellung von Diffs als auch der textuellen Ersetzung durch den Präprozessor besondere Bedeutung zu. Daneben sind aber auch solche Fälle zu betrachten, in denen allein die Programmverwendung schon zu einer Integration von GPL-lizenziertem Code im Arbeitsergebnis führen kann.[486]

aa)Erstellung von Diffs

Mit dem Unix Programm „Diff“[487] lassen sich Dateien erzeugen, welche die Unterschiede zwischen Originaldatei und geänderter Version beinhalten. Diese Dateien werden meistens als Patch oder Diff bezeichnet[488] und dafür eingesetzt, um Sie auf die Originaldateien anzuwenden. Nach der Anwendung eines Diffs auf die Originaldatei werden die im Diff enthaltenen Änderungen an diesem vorgenommen.

Eine Diff-Datei ist dabei generell so aufgebaut, dass sowohl Codebestandteile aus dem Original als auch aus der veränderten Version enthalten sind. Codebestandteile der Originalversion werden aufgeführt, um deren Löschen zu bewirken. Dafür wird jede Zeile, die gelöscht werden soll, mit einem Minuszeichen gekennzeichnet.[489] Diejenigen Codebestandteile, die ausschließlich in der veränderten Programmversion vorhanden sind, werden hingegen mit einem Pluszeichen gekennzeichnet, damit sie letztlich dem Quellcode des Originals hinzugefügt werden. Alle anderen Codezeilen, in denen keine Änderungen durchgeführt werden sollen, sind durch Leerzeichen gekennzeichnet.

Berücksichtigt man diesen Aufbau eines Diffs, so wird ersichtlich, dass die bislang in der Literatur getroffenen Annahmen hierzu, wonach diese keinen urheberrechtlich geschützten GPL-Code enthalten sollen,[490] in dieser Allgemeinheit nur dann zutreffen, wenn einem Programm durch die Anwendung eines Diffs lediglich Funktionalität hinzugefügt werden soll.

---

[486] Hierzu ab S. 127.
[487] Eine Implementierung des ursprünglichen Programms findet sich auch im Paket diffutils des GNU-Projekts, http://www.gnu.org/software/diffutils/diffutils.html.
[488] Vgl. http://www.gnu.org/software/diffutils/diffutils.html.
[489] Die genaue Bezeichnung der Zeilen hängt vom gewählten Ausgabeformat ab. Plus- und Minuszeichen werden beim Unified Format verwendet, vgl. http://www.gnu.org/software/diffutils/manual/html_node/Detailed-Unified.html#Detailed%20Unified.
[490] *Spindler*, Rechtsfragen bei Open Source, C. Rn. 124.

Dies entspricht allerdings nicht der Regel, da Diffs dem Original zumeist nicht nur Code hinzufügen, sondern auch vorbestehenden Code verändern und mithin Teile hiervon löschen.

Betrachtet man das Diff völlig losgelöst vom Einsatzzweck, so enthält dieses in all jenen Fällen Teile des GPL-lizenzierten ursprünglichen Programms, in denen Teile desselben gelöscht werden sollen. Infolgedessen stellt ein solches Diff eine veränderte Programmkopie im Sinne der Ziff. 2 Abs. 1 GPL dar, sofern nicht nur unerhebliche Programmteile gelöscht werden sollen.

bb)Textuelle Ersetzungen durch den Präprozessor

Die Vorgehensweise des Präprozessors lässt sich mit der „suchen und ersetzen" Funktion in Textprogrammen beschreiben. Der Präprozessor ersetzt entsprechende Textstellen im Programmcode durch den Quelltext anderer Dateien, die er zu diesem Zweck zuvor kopiert.[491] Die Anweisungen erhält der Präprozessor dabei durch sog. Präprozessor-Direktiven, die sich lediglich in der Syntax, nicht aber in funktionaler Hinsicht zwischen den Programmiersprachen unterscheiden. In den Programmiersprache C, in welcher der überwiegende Teil des Linux-Kernels geschrieben wurde, können für die textuellen Ersetzungen einzig die Anweisung `#include` und `#define` relevant werden. Ohne solche Präprozessor-Direktiven kommt nahezu kein Programm aus, selbst im „Hello World" Programm findet sich die #include Direktive (in der Programmiersprache C):

```
#include <stdio.h>

main() {
   printf("Hello World\n");
}
```

(1)Einbeziehung von Headern - „#include" Direktive

Die `#include` Anweisung, wie sie im oben genannten Beispielprogramm enthalten ist, führt dazu, dass der Präprozessor die bezeichnete externe Textdatei (im Beispiel:

[491] *Klima / Selberherr*, Programmieren in C, S. 25 f.

`stdio.h)` einliest und den entsprechenden Quelltext an diese Stelle kopiert. Nach der Bearbeitung durch den Präprozessor würde mithin das „Hello World“ Programm um den gesamten Quelltext aus der Datei stdio.h erweitert sein.

Durch die `#include` Anweisung ist es somit möglich beliebigen Quelltext durch den Präprozessor einfügen zu lassen. Insofern stellen textuelle Ersetzungen des Präprozessors eine Veränderung der Programmkopie dar, sofern der eingefügte Quelltext der externen Datei unter der GPL lizenziert wurde und seinerseits einen urheberrechtlich schutzfähigen Inhalt aufweist.

In der Regel werden `#include` Anweisungen dazu verwendet, um Informationen verfügbar zu machen, die für mehrere Programme von Relevanz sind. Dabei handelt es sich zumeist um Typendefinitionen, Konstanten oder Funktionsprototypen. Der Hauptanwendungsfall von `#include` Anweisungen liegt somit in der Einbindung von Standard-Header-Dateien, in denen solche Schnittstelleninformationen enthalten sind. Header-Dateien sind dabei dafür erforderlich, um den Zugriff auf bestimmte System- oder Programmbibliotheken zu ermöglichen.

Header-Dateien weisen oftmals komplexe Anweisungen und Funktionstypen auf, so dass deren urheberrechtliche Schutzfähigkeit zumindest nicht generell ausgeschlossen werden kann.[492]

Sofern die externe Textdatei dabei einen schutzfähigen Inhalt aufweist und unter der GPL lizenziert wurde, hat dies zur Folge, dass der vom Präprozessor erstellte Quelltext GPL-lizenzierte Teile enthält und mithin als *„derivative work“* zu bewerten ist. Ein Vertrieb dieser Datei könnte somit nur unter den zusätzlichen Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL erfolgen.

Das Gleiche wird häufig – allerdings nicht zwangsläufig – auch für die aus einem solchen Quelltext kompilierte Objektdatei, sowie für das ausführbare Programm gelten, das der Linker unter Zugrundelegung solcher Objektdateien erstellt. Eine Ausnahme kommt dann in Betracht, wenn die ursprüngliche Quelltextdatei zwar eine `#include` Anweisung enthalten hat, der eigentliche Quelltext hierauf aber keinen Bezug nimmt. Ist dies gegeben, wird die kompilierte Objektdatei keine Teile mehr enthalten, die aus der GPL-lizenzierten Headerdatei

[492] Ähnlich in der Einschätzung *Jaeger* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 2 GPL Rn. 38. Allerdings ist zu berücksichtigen, dass die den Schnittstellen zugrundeliegenden Ideen und Grundsätze gem. § 69 a Abs. 2 S. 2 UrhG keinen urheberrechtlichen Schutz genießen. Insofern wird allein die Einbeziehung von Schnittstellendefinitionen in der Regel nicht dazu führen, dass hierdurch ein *„derivative work“* entsteht. Etwas anderes mag gelten, wenn das betreffende Header-file auch umfassende Funktionsdefinitionen enthält; zur Einbeziehung von inline-Funktionen vgl. sogleich ab S. 128.

stammen, da der Compiler den Quelltext unter anderem auch dahin gehend optimiert, dass er ausschließlich diejenigen Teile aus einer Header-Datei übernimmt, die der ursprüngliche Quelltext unbedingt benötigt, unnötige Teile werden hingegen gelöscht.
Nach der Kompilierung muss daher erneut überprüft werden, ob überhaupt urheberrechtlich geschützte Teile aus der Header-Datei verwendet wurden. Allein aus der Feststellung, dass eine urheberrechtlich geschützte Header-Datei durch den Präprozessor eingebunden wurde, kann mithin nicht ohne weiteres darauf geschlossen werden, dass auch die durch den Compiler erstellte Objektdatei letztlich noch urheberrechtlich relevante Teile der Header-Datei beinhaltet.

(2)Textersetzung durch Makros - „#define“ Direktive

Ähnlich wie bei der `#include` Anweisung, nimmt der Präprozessor textuelle Ersetzung auch dann vor, wenn er auf die Anweisung `#define` trifft. Im Gegensatz zur `#include` Anweisung besteht im Rahmen von solchen sog. Makros, allerdings keine Möglichkeit externen Quelltext einzubeziehen. Grundsätzlich dienen Makros[493] allein dafür, besonders häufig verwendete Codefragmente (sowohl Parameter als auch Funktionen) während des Programmierens, durch eine einheitliche Bezeichnung vor dem Kompilieren zu ersetzten. Der zu ersetzende Text muss dabei allerdings bereits in der eigenen Quelltextdatei (oder auch in der zuvor einbezogenen Header-Datei) enthalten gewesen sein. Insofern kann im Rahmen der Textersetzung kein „Fremdcode“ Eingang in den Quelltext finden.[494]
Aus lizenzrechtlicher Sicht kann daher die Verwendung von Makros für sich betrachtet keine Veränderung der ursprünglichen Programmkopie im Sinne von Ziff. 2 Abs. 1 GPL zur Folge haben.

cc)Programmbenutzung führt zur Übernahme von GPL-lizenziertem Code

[493] In diesem Zusammenhang werden unter Makros sowohl sog. Substitutionsmakros als auch Funktionsmakros verstanden.
[494] Abgesehen von dem Text, der ggf. bereits durch die Header-Datei einbezogen wurde. Sofern dies erfolgt ist, liegt aber bereits die Fallgruppe der „#include“ Direktive vor, so dass dem zusätzlichen Vervielfältigen des entsprechenden Textes an den verschiedenen Stellen im Quelltext kein zusätzliches Gewicht beizumessen ist.

Ein in der Literatur immer wieder diskutierter[495] Sonderfall ist der, in dem ein GPL-lizenziertes Entwicklungsprogramm selbst, also unabhängig von den im Quellcode enthaltenen Anweisungen, GPL-lizenzierten Code in die Objektdateien einfügt. Dies führte in der Vergangenheit im Zusammenhang mit der Verwendung des Parsers Bison zu Problemen, weshalb dieser heute nicht mehr unter der Originalfassung der GPL veröffentlicht ist.[496] Außer diesem Einzelfall sind bislang jedoch keine weiteren Fälle bekannt geworden, bei denen allein die Nutzung eines Programms dazu geeignet gewesen wäre, eine Lizenzierungspflicht auszulösen.
Allerdings kann gleichwohl festgehalten werden, dass - zumindest theoretisch - die Möglichkeit besteht, dass ein Programm im Rahmen seiner Nutzung letztlich zu einer Lizenzierungspflicht für das Arbeitsergebnis führt, sofern urheberrechtlich geschützter Code in dieses übertragen wird.[497] Sollte mithin ein solches Programm zur Softwareentwicklung verwendet werden, könnte der Vertrieb der dadurch erstellten Datei nur unter den zusätzlichen Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL erfolgen.

e.GPL-Quelltext oder Teile davon werden in einer Objektdatei verwendet

Zu Veränderungen an der ursprünglichen Programmkopie kommt es weiterhin durch den Kompilierungsvorgang, mit dem Quelltextdateien in Objektdateien umgewandelt werden. Hierbei wird der ursprüngliche Quelltext in eine maschinenlesbare Form übersetzt, so dass er bereits aus diesem Grund eine Veränderung der ursprünglichen Programmkopie vorliegt.[498] Letztlich lässt sich auch aus § 69 e Abs. 1 UrhG entnehmen, dass die Übersetzung des Quellcodes in den Objektcode eine Umgestaltung im Sinne von § 69 c Nr. 2 UrhG darstellt und mithin als Bearbeitung zu bewerten ist.[499] Hinzu kommt, dass neben der reinen Übersetzung auch eine Reihe von Optimierungen am Quelltext vorgenommen werden, die unter anderem zu einem Löschen sämtlicher Kommentare, sowie einem Auflösen von Abhängigkeiten führt.

[495] Vgl. *Wuermeling / Deike*, CR 2003, 87, 87 f; *Jaeger / Metzger*, Open Source Software, Rn. 60 f; *Spindler*, Rechtsfragen der Open Source Software, S. 58, www.vsi.de/inhalte/aktuell/studie_final_safe.pdf.
[496] Vgl. http://www.gnu.org/software/bison/manual/html_mono/bison.html#Conditions.
[497] Vgl. *Wuermeling / Deike*, CR 2003, 87, 87 f.
[498] Vgl. *Dreier* in: Dreier / Schulze, § 69 c Rn. 16.
[499] *Marly*, Softwareüberlassungsverträge, Rn. 1190; *Dreier* in: Dreier / Schulze, § 69 c Rn. 16.

Ein Vertrieb dieser Datei kann somit nur unter den zusätzlichen Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL erfolgen, sofern die ursprüngliche Quelltextdatei urheberrechtlich schutzfähig war und die hieraus erstellte Objektdatei somit ein *„derivative work“* darstellt.

Eine weitere Fallgruppe, in der GPL-lizenzierter Quelltext oder Teile desselben in einer Objektdatei verwendet werden, könnte in den Fällen vorliegen, in denen sog. `inline` Funktionsaufrufe im ursprünglichen Quelltext enthalten sind. In der Programmiersprache C++ soll der Compiler durch die `inline` Anweisung dazu bewegt werden, anstatt einer Referenz auf eine externe Funktion, deren entsprechenden Funktionscode unmittelbar in die Objektdatei einzufügen. Sofern der Compiler dieser Anweisung folgt,[500] enthält die erstellte Objektdatei den Code der entsprechenden Funktion.
War der entsprechende Funktionscode dabei unter der GPL lizenziert, könnte sich die Objektdatei wiederum als veränderte Programmkopie darstellen, die nur unter den zusätzlichen Bedingungen der Ziff. 2 Abs. 1 a-c GPL vervielfältigt und vertrieben werden darf.
Voraussetzung hierfür ist jedoch, neben einer Veränderung der Programmkopie, dass durch diese ein *„derivative work“* entsteht, also nicht nur unwesentliche Teile übernommen wurden.[501]
`Inline`-Funktionen werden jedoch in aller Regel dazu verwendet, um die Programmgeschwindigkeit zu optimieren, indem kurze Funktionen nicht extern aufgerufen, sondern unmittelbar in den Quellcode geschrieben werden.[502]
Daraus wird ersichtlich, dass die Verwendung von `inline` Funktionen eher selten lizenzrechtliche Konsequenzen haben wird. Dies liegt daran, dass im Hinblick auf die Programmoptimierung in der Regel nur besonders kurze Funktionen vom Compiler berücksichtigt werden, welche die erforderliche Schöpfungshöhe nur ausnahmsweise erreichen sollten. Insofern wird in aller Regel allein durch die Übernahme solcher Codeteile noch kein *„derivative work“* entstehen.[503]

[500] Zumindest im Rahmen der Standardeinstellungen behandeln Compiler solche Anweisungen nicht als zwingend und befolgen sie daher nur, sofern durch die unmittelbare Integration des externen Codes ein Performancevorteil erreicht wird; siehe hierzu *Schmaranz*, Softwareentwicklung in C++, S. 96 f.
[501] Zum Erfordernis der Wesentlichkeit siehe oben S. 118 f.
[502] Hierdurch wird der sog. Overhead verhindert, der dadurch entsteht, dass bei jedem normalen Funktionsaufruf etwas Zeit dadurch verloren geht, dass erst in die entsprechende Funktion gesprungen und der Rückgabewert der Funktion wieder an die entsprechende Stelle zurückgeführt werden muss.
[503] Ebenso in der Einschätzung *Jaeger* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 2 Rn. 41.

f.GPL-Quelltext wird zu ausführbarem Programm

Sofern GPL-Quelltext zu einer ausführbaren Datei kompiliert werden soll, werden im Rahmen der Kompilierung die bereits beschriebenen Kompilierungsphasen durchlaufen. Falls der Quelltext oder Teile desselben dabei nicht bereits im Rahmen der textuellen Ersetzungen des Präprozessors verändert wurden, findet eine solche spätestens mit dem eigentlichen Kompilierungsvorgang – also der Übersetzung in den maschinenlesbaren Code – statt.[504] Sofern eine Objektdatei aus einem GPL-lizenzierten Quelltext generiert wird, stellt diese demnach ein *„derivative work“* dar, dass ausschließlich unter den Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL vervielfältigt oder verbreitet werden darf.

Dasselbe gilt auch für die durch den Linker erstellte ausführbare Datei, die auf dieser veränderten Objektdatei beruht. In dieser ist die veränderte Objektdatei vollumfänglich enthalten, so dass das ausführbare Programm ebenfalls ein *„derivative work“* darstellt.

g.GPL-Programm lag als Objekt-Datei vor

Im Gegensatz zum Erwerb der Programmkopie in Form des Quelltextes bestehen beim Erwerb des GPL-lizenzierten Programms in der Form einer Objektdatei nur begrenzte Möglichkeiten diese zu verändern.

Typischerweise wird eine solche Veränderung immer nur in der Form stattfinden, dass die GPL-lizenzierten Objektdateien zusammen mit weiteren Dateien zu einem einheitlichen ausführbaren Programm gelinkt werden. Wie bereits beschrieben wurde, sorgt der Linker dafür, dass alle benötigten Quellen zu einer einheitlichen ausführbaren Datei verknüpft werden. Diese Verknüpfung kann dabei auf statische und dynamische Art erfolgen, wobei sich das statische vom dynamischen Linken dadurch unterscheidet, dass der Linker nicht nur eine Referenz auf eine externe Datei,[505] sondern deren gesamten Objektcode direkt in das ausführbare Programm einfügt.

Häufigster Anwendungsfall, in dem die Verknüpfung von Objektdateien lizenzrechtlich relevant werden kann, ist der, in dem die von einem proprietären Programm benötigten Programmbibliotheken unter der GPL lizenziert sind. Unter Programmbibliotheken werden solche Sammlungen von Programmroutinen bzw. Standardbefehlen verstanden, welche

[504] Zum Softwareentwicklungsprozess vgl. bereits S. 121 ff.

[505] Eine solche Referenz enthält dabei lediglich den Hinweis, in welchen Verzeichnissen das Programm zum Zeitpunkt der Laufzeit nach den benötigten Bibliotheken suchen soll (für Linux z.B. das Verzeichnis /usr/include/…).

wesentliche Funktionalitäten des Betriebssystems bereithalten und daher besonders häufig von Programmen benötigt werden.

Das statische Linken einer GPL-lizenzierten Programmbibliothek führt dazu, dass diejenigen Funktionsaufrufe in der Objektdatei, die sich auf diese Programmbibliothek beziehen durch den Linker in den Objektcode übernommen werden. Die durch den Linker in dieser Art erstellte einheitliche ausführbare Datei enthält folglich – neben den eigenen Entwicklungen – maschinenlesbaren Code aus der Programmbibliothek, so dass sich die ausführbare Datei letztlich als Veränderung der ursprünglichen Programmkopie darstellt.

Bei den in der Programmbibliothek enthaltenen Funktionen handelt es sich überwiegend um urheberrechtlich schutzfähige Anweisungen, so dass das statische Einbinden von Programmbibliotheken regelmäßig zu einer ausführbaren Datei führen wird, die einem *„derivative work“* entspricht.

Der Vertrieb von ausführbaren Programmen, welche Funktionen aus GPL-lizenzierten Programmbibliotheken in statischer Art verwenden, kann folglich nur unter den Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL erfolgen.[506]

Im Gegensatz hierzu führt allein die dynamische Verknüpfung von GPL-lizenzierten Programmbibliotheken nicht dazu, dass das ausführbare Programm von Ziff. 2 Abs. 1 GPL erfasst wird. Sofern der Linker GPL-lizenzierte Programmbibliotheken mit einer ausführbaren Datei linkt, fügt dieser nicht deren Objektcode, sondern lediglich eine Referenz auf diese in die ausführbare Datei ein. Eine solche Referenz enthält dabei lediglich die Pfadangaben, an denen das Programm zum Zeitpunkt der Ausführung nach den benötigten Programmbibliotheken suchen soll,[507] jedoch keinen GPL-lizenzierten Objektcode.

Insofern wird durch dynamisches Linken keine Veränderung an der ursprünglichen Programmkopie herbeigeführt, so dass der Anwendungsbereich der Ziff. 2 GPL allein durch die dynamische Verlinkung nicht zwangsweise eröffnet ist.[508]

---

[506] Ebenso *Jaeger* in: Ifross, Ziff. 2 Rn. 43 ff; *Andréewitch*, MR 2005, 240, 242; *Determann*, GRUR Int. 2006, 645, 648.

[507] Bei dem „Hello world“ Beispielprogramm würde eine solche Referenz den Hinweis enthalten, dass die Programmbibliothek stdio.h bei Linux unter dem Pfad /usr/include/stdio.h zu finden ist.

[508] Zur Frage, ob durch den gemeinsamen Vertrieb von Programm und dynamischer Programmbibliothek ein einheitliches Ganzes im Sinne von Ziff. 2 Abs. 2 S. 3 GPL entsteht, vgl. die Ausführungen ab S. 165; a.A. ist allerdings die FSF, die ohne eine Differenzierung bei jeder dynamischen Verlinkung vom Vorliegen eines einheitlichen Programms ausgeht, vgl. http://www.fsf.org/licensing/licenses/gpl-faq.html#GPLAndPlugins.

h.Ursprüngliche Programmkopie wird unverändert verwendet

Lizenzrechtlich relevante Veränderungen an der ursprünglichen Programmkopie könnten selbst dann gegeben sein, wenn die ursprüngliche Programmkopie unverändert bleibt. Dies klingt zunächst widersprüchlich, würde aber auch dem urheberrechtlichen Verständnis einer Bearbeitung nach deutschem Recht entsprechen. So ist nach h.M. anerkannt, dass eine urheberrechtlich relevante Bearbeitung im Sinne von § 23 UrhG auch dann vorliegen kann, wenn das ursprüngliche Werk in seiner Substanz unberührt bleibt, aber in einen anderen Zusammenhang gestellt wird und hierdurch insgesamt ein anderer Gesamteindruck vom Werk entsteht.[509] Der BGH nahm dies in einem Fall an, in dem das geschützte Werk in ein neues „Gesamtkunstwerk" derart integriert wurde, dass es als dessen Teil erschien.[510]

Solche Fälle scheint die GPL mit Ziff. 2 Abs. 2 S. 3 erfassen zu wollen, wenn darin für die Beurteilung als *„derivative work"* auf die Vertriebsform abgestellt wird. Werden unabhängige und eigenständige Abschnitte als Teil eines Ganzen weitergegeben, so sollen die Abschnitte Teile des *„work based on the Program"* werden und mithin ein *„derivative work"* darstellen, das insgesamt unter den Voraussetzungen der GPL zu vertreiben ist.

Insofern kann auch die unveränderte Übernahme von GPL-lizenzierten Quelltext-, Objekt- oder auch ausführbaren Dateien in Programme eine Veränderung der ursprünglichen Programmkopie im Sinne von Ziff. 2 Abs. 1 GPL darstellen.

Ob letztlich tatsächlich ein *„derivative work"* erstellt wurde hängt in diesen Fällen einzig von der Vertriebsform ab, auf die in der dritten Stufe genauer eingegangen wird.

An dieser Stelle reicht insofern die Feststellung, dass auch die unveränderte Übernahme von GPL-lizenzierten Dateien unter Umständen eine lizenzrechtlich relevante Veränderung der ursprünglichen Programmkopie bewirken und mithin der Anwendungsbereich der Ziff. 2 Abs. 1 GPL auch in diesen Fällen eröffnet sein kann.[511]

Zusammenfassung:

Insgesamt lassen sich mithin eine Reihe von Fallgruppen festhalten, in denen die Vervielfältigung und der Vertrieb ausschließlich unter den zusätzlichen Voraussetzungen der Ziff. 2 Abs. 1 a-c GPL erfolgen darf, weil es zu einer Veränderung an der GPL-lizenzierten

[509] BGH GRUR 2002, 532, 534 – *Unikatrahmen*; *Dreier* in: Dreier / Schulze, § 23 Rn. 8; a.A. wohl *Ahlberg* in: Möhring / Nicolini, § 23 Rn. 11, der eine Veränderung des benutzten Werks generell vorauszusetzen scheint.
[510] BGH GRUR 2002, 532, 534 – *Unikatrahmen.*
[511] Zu den Einzelheiten siehe unten S. 165 ff.

Programmkopie oder Teilen hiervon gekommen ist und diese in einer Art und Weise stattgefunden hat, dass das daraus resultierende Werk sich als *„derivative work“* darstellt.

Einfachster Fall, in dem eine Veränderung an einer Programmkopie zu einem *„derivative work“* führt, liegt zunächst darin, dass urheberrechtlich schutzwürdigen Quelltextdateien einer GPL-lizenzierten Programmkopie direkt verändert werden.

Ein *„derivative work“* entsteht weiterhin dadurch, dass eine Quelltextdatei oder Teile davon in eigene Entwicklungen übernommen werden, indem der GPL-lizenzierte Quelltext insgesamt oder auch in Auszügen kopiert und in andere Quelltextdateien eingefügt wird.
Diese Übernahme von GPL-lizenziertem Quelltext kann dabei sowohl manuell, aber auch automatisiert, etwa durch den Präprozessor im Rahmen des Kompilierungsvorgangs, erfolgen.

Urheberrechtlich geschützter Quelltext wird zudem häufig auch in sog. Diffs enthalten sein, sofern dadurch einem GPL-lizenzierten Programm nicht nur Funktionalität hinzugefügt, sondern, Teile desselben im Rahmen der Überarbeitung auch gelöscht werden sollen.

Weitaus seltener sind hingegen solche Fallgestaltungen, in denen die schlichte Nutzung eines GPL-lizenzierten Programms dazu führt, dass damit hergestellte Arbeitsergebnisse sich als *„derivative work“* darstellen. Dies geschah in der Vergangenheit zwar im Hinblick auf den Einsatz des Programms Bison, allerdings sind derzeit keine Hinweise dafür zu erkennen, dass es weitere Programme gäbe, bei denen bereits die schlichte Nutzung zu einer Geltung von Ziff. 2 GPL führen könnte.
Ebenso wird auch die Einbindung von `inline` Funktionen eher selten zu einem *„derivative work“* führen, da es sich hierbei in der Regel nur um besonders kurze Funktionen handelt, welche die erforderliche Schöpfungshöhe bloß ausnahmsweise erreichen.
Werden hingegen Funktionen eingebunden, die in Programmbibliotheken enthalten sind, so sind diese überwiegend urheberrechtlich schutzfähig. Das statische Einbinden von Programmbibliotheken in eine ausführbare Datei wird daher regelmäßig dazu führen, die diese ein *„derivative work“* darstellt.

Die vorliegende Untersuchung hat weiterhin gezeigt, dass auch durch die Kompilierung von GPL-lizenzierter Quelltextdateien, lizenzrechtlich relevanten Veränderungen vorgenommen

werden und sich daraus hervorgehenden Objektdateien folglich als *„derivative work“* darstellen.
Dasselbe gilt auch für ausführbare Dateien, die der Linker unter Zusammenfügung mehrerer Objektdateien erstellt, sofern hierunter GPL-lizenzierte Objektdateien vorhanden sind.

Letztlich wurde auch festgestellt, dass die unveränderte Übernahme von GPL-lizenzierten Dateien unter gewissen Voraussetzungen eine lizenzrechtlich relevante Veränderung der ursprünglichen Programmkopie bewirkt und der Anwendungsbereich der Ziff. 2 Abs. 1 GPL eröffnet sein kann.

2.Stufe 2 – unabhängige und eigenständige Werke vorhanden?

Nachdem auf Stufe 1 festgestellt wurde, dass lizenzrechtlich relevante Veränderungen der ursprünglichen Programmkopie oder Teilen davon vorgenommen wurden, ist auf der zweiten Stufe nunmehr die Reichweite der dadurch hervorgerufenen Lizenzierungspflichten zu prüfen. Für Unternehmen stellt sich – einen modularen Programmaufbau vorausgesetzt[512] – die Frage, welche Werkteile von der Lizenzierungspflicht betroffen und daher zusammen mit dem dazugehörigen Quelltext zu veröffentlichen sind.

Ausgangspunkt für die Beantwortung dieser Frage ist wiederum Ziff. 2 Abs. 1 b GPL, sowie die damit im Zusammenhang stehenden Ausnahmen in Ziff. 2 Abs. 2 S. 2 und Ziff. 2 Abs. 4 GPL. Eine eindeutige Antwort auf die Frage nach der Reichweite der Lizenzierungspflicht wird sich danach immer dann geben lassen, wenn die Voraussetzungen von Ziff. 2 Abs. 1 b GPL erfüllt, also das Programm als Ganzes oder Teile hiervon übernommen wurden, und zugleich sichergestellt ist, dass weder die Ausnahme in Ziff. 2 Abs. 2 S. 2 noch die in Ziff. 2 Abs. 4 GPL eingreifen kann.[513] Nachdem auf Stufe 1 dargestellt wurde, wann die Voraussetzungen der Ziff. 2 Abs. 1 b GPL gegeben sind, bleibt zu untersuchen, unter welchen Voraussetzungen davon ausgegangen werden kann, dass weder die Ausnahme von Ziff. 2 Abs. 2 S. 2 noch die von Ziff. 2 Abs. 4 GPL eingreifen.
Diese Voraussetzungen sind – unabhängig von der Ausgestaltung des Vertriebs – immer dann gegeben, wenn innerhalb der Entwicklungen zwar GPL-lizenzierte Teile verwendet wurden, aber zugleich unter den vorhandenen Komponenten kein unabhängiger und eigenständiger Teil im Sinne von Ziff. 2 Abs. 2 S. 2 GPL zu identifizieren ist.
Liegen diese Voraussetzungen vor, steht zunächst fest, dass alle Programmteile einem einheitlichen Werk zuzurechnen sind und die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL folglich nicht in Betracht kommt. Darüber hinaus kann jedoch auch Ziff. 2 Abs. 4 GPL nicht zur Anwendung gelangen, da auch hierfür eigenständige Werke vorauszusetzen sind.[514]

[512] Andernfalls kommt in Anbetracht dessen, dass in Ziff. 2 Abs. 2 S. 2 GPL ein Vertrieb *„as separate works“* gefordert wird, die darin enthaltene Ausnahme offensichtlich nicht in Betracht, weshalb sich die Lizenzierungspflicht auf sämtliche Werkteile beziehen würde.
[513] Vgl. hierzu die Ausführungen oben S. 115 ff.
[514] Dies folgt offensichtlich aus dem Wortlaut von Ziff. 2 Abs. 4 GPL: *„…mere aggregation of **another** work not based on the Program with the Program…”.*

Als Folge davon bleibt es bei der in Ziff. 2 Abs. 1 b GPL geregelten Lizenzierungspflicht für das Werk als Ganzes.[515]

Wonach kann aber bei Vorliegen mehrerer Komponenten bestimmt werden, ob diese unabhängig und eigenständig oder lediglich unselbstständige Teile anderer Komponente sind? Dem Wortlaut der GPL lassen sich in Ziff. 2 Abs. 2 S. 2 zunächst folgende Voraussetzungen entnehmen:

"…If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms…".

Hiernach wären die identifizierbaren Teile des Werks daraufhin zu untersuchen, ob sie vernünftigerweise als unabhängig und eigenständige Werkteile zu beurteilen sind. Eine Konkretisierung von Ziff. 2 Abs. 2 S. 2 GPL könnte sich zudem daraus ergeben, dass die GPL mit den oben benannten Voraussetzungen nahezu wörtlich an die Anforderungen anknüpft, die im US-Rechts an das Vorliegen eines *„collective work"* gestellt werden. Entsprechend 17 U.S.C. § 101 ist dies *"a work, such as a periodical issue, anthology, or encyclopedia, in which a number of contributions, constituting* ***separate and independent works in themselves****, are assembled into a collective whole.".*

Insofern könnte Ziff. 2 Abs. 2 S. 2 GPL für solche Werke eine Ausnahme enthalten, die sich als *„collective work"* darstellen, welches der Werkverbindung im deutschen Recht entspricht. Für die Bestimmung der Unabhängigkeit und Eigenständigkeit der Komponenten würde es folglich darauf ankommen, ob diese zusammen mit den abgeleiteten Programmkomponenten als Teile eines einheitlichen Werks zu betrachten sind, oder hiervon zu trennende eigenständige Werkteile darstellen.

Auf eine solche Auslegung der GPL deutet, neben dem Wortlaut von Ziff. 2 Abs. 2 GPL, auch Ziff. 2 Abs. 3 GPL[516] hin, wonach es nicht nur beabsichtigt ist, den Vertrieb von *„derivative works"*, sondern auch den von *„collective works"* zu regeln, womit die GPL – mangels anderweitiger Regelungen zu *„collective works"* – ausschließlich auf Ziff. 2 Abs. 2 S. 2 GPL Bezug nehmen kann.

[515] Für die Bedeutung von Ziff. 2 Abs. 4 GPL folgt daraus, dass sie im Hinblick auf die Regelung in Ziff. 2 Abs. 2 darin liegt, dass erläuternd klargestellt wird, dass die Reichweite der Lizenzierungspflicht selbst dann ausgeschlossen sein kann, wenn unabhängige und eigenständige Werke auf einem einheitlichen Vertriebsmedium weitergegeben werden. Ziff. 2 Abs. 4 GPL: *„In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License."*

[516] Ziff. 2 Abs. 3 GPL: *"Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you****; rather, the intent is to exercise the right to control the distribution of*** *derivative or* ***collective works*** *based on the Program."*

Für die Auslegung von Ziff. 2 Abs. 2 S. 2 GPL sind daher die Kriterien heranzuziehen, anhand derer die Abgrenzung von eigenständigen zu einheitlichen Werken bei *„collective works“*, bzw. entsprechend hierzu bei Werkverbindungen im Sinne von § 9 UrhG, vorgenommen wird.

a.Eigenständiges oder einheitliches Werk?

Für Werkverbindungen erfolgt die Abgrenzung zwischen Werkeinheit und Werkmehrheit danach, ob eine gesonderte Verwertbarkeit der einzelnen Werkteile besteht. Eine solche wird zunächst einmal dann angenommen, wenn es sich um Werkteile verschiedener Werkarten handelt.[517] Aber auch wenn Werkteile derselben Werkgattung gegeben sind, entspricht es der allgemeinen Ansicht, dass eine gesonderte Verwertbarkeit möglich ist.[518]
Von einer gesonderten Verwertbarkeit wird in diesen Fällen ausgegangen, wenn sich die Anteile an dem Werk herauslösen lassen, ohne dadurch unvollständig und ergänzungsbedürftig zu werden und es zudem denkbar ist, dass sie in irgendeiner Weise wieder Verwendung finden.[519]

b.Herauslösen einzelner Werkteile möglich

Im Hinblick auf die modulare Struktur von modernen Computerprogrammen erscheint das Herauslösen einzelner Programmkomponenten[520] ohne weiteres möglich, sofern diese in getrennten Dateien vorliegen. Eine Trennbarkeit der Werke erscheint daher immer dann gegeben zu sein, wenn diese aus mehreren Dateien bestehen. Fraglich ist allerdings, wie in den Fällen zu verfahren ist, in denen zwar zum Vertriebszeitpunkt eine einheitliche Datei

[517] Vgl. nur *Loewenheim* in: Schricker, § 8 Rn. 6 und § 9 Rn. 5 m.w.N.; *Ahlberg* in: Möhring / Nicolini, § 8 Rn 11.
[518] Vgl. *Loewenheim* in: Schricker, § 8 Rn. 6; Kritik an der Voraussetzung einer gesonderten Verwertbarkeit übt vor allem *Ahlberg* in: Möhring / Nicolini, § 8 Rn. 15 ff.
[519] KG Schulze Rspr. KGZ 55, 12 – Puppenfee; *Loewenheim* in: Schricker, § 8 Rn. 5; v. Gamm § 8 Rn. 11; *Waldenberger*, Die Miturheberschaft im Rechtsvergleich, S. 18 f.
[520] Der Begriff Komponente wird vorliegend losgelöst von dessen technischer Bedeutung verwendet und umfasst daher Module, Komponenten und Objekte.

(z.B. eine ausführbare Exe-Datei) vorliegt, diese jedoch nach deren Installation wiederum die darin enthaltenen Programmkomponenten erkennen lässt.[521]

Im Gegensatz zu den klassischen Werkstücken, zu denen die benannten Kriterien herausgebildet wurden,[522] stellt sich das Problem, dass ein Computerprogramm unterschiedliche „Aggregatzustände" aufweisen kann.[523] Insofern erscheint es zum Einen fraglich, ob diese Kriterien überhaupt auf Computerprogramme angewendet werden können und zum Anderen, auf welchen Zeitpunkt für die Betrachtung abzustellen wäre?

In der Literatur wird überwiegend vertreten, dass in den Fällen, in denen ein Programm als einheitliche Datei vertrieben werde, die nötige formelle Trennbarkeit fehle, weshalb das Werk rein faktisch als einheitliches zu bewerten sei.[524] Als Beispiel wird hierfür der Vertrieb von Exe-Dateien genannt. Gegen eine solche Annahme könnten indes Bedenken bestehen, weil sie voraussetzt, dass es für die Beurteilung der Eigenständigkeit von Werkteilen ausschließlich auf den Vertriebszeitpunkt ankommen würde. Eine Annahme, die zwar im Hinblick auf die Werkverbindung im Sinne von § 9 UrhG nicht zwingend erscheint, jedoch im Zusammenhang mit der GPL keiner abschließenden Beurteilung bedarf, da die Ausnahme in Ziff. 2 Abs. 2 S. 2 GPL – abweichend vom deutschen Verständnis – nicht nur darauf abstellt, dass unabhängige und eigenständige Werke vorliegen, sondern auch an deren Vertriebsform (*„…when you distribute them as separate works."*) anknüpft und damit offensichtlich auf diesen Zeitpunkt abstellt.

Für das Vorliegen von unabhängigen und eigenständigen Werkteilen kann damit als erste Voraussetzung festgehalten werden, dass nur solche Programmkomponenten in Betracht kommen, die zum Vertriebszeitpunkt in einer trennbaren Form zu den sonstigen Programmteilen vorliegen und entsprechend Ziff. 2 Abs. 2 S. 2 GPL weder GPL-lizenzierten Code beinhalten noch von solchem abgeleitet sind.[525]

Im Hinblick auf die Frage nach der Werkeigenschaft der jeweiligen Programmteile erscheint das Kriterium der Herauslösbarkeit allerdings nur begrenzt aussagekräftig zu sein, da gerade der Softwarebereich dadurch gekennzeichnet ist, dass sich die Form eines Programms zum Zeitpunkt des Vertriebs weitgehend variabel gestalten lässt. Es ist für einen Entwickler ohne

[521] So ist es durchaus üblich Programme zusammen mit einem Installationsprogramm (sog. Installer) als einheitliche ausführbare Datei zu vertreiben, welche sich jedoch nach deren Installation wieder in die entsprechenden Verzeichnisse und Dateien unterteilt.

[522] Die maßgeblichen Entscheidungen ergingen zu Büchern und deren Verfilmung, vgl. BGH GRUR 1959, 335, 336 – Wenn wir alle Engel wären; sowie zu Liedtexten und deren musikalischen Kompositionen, vgl. GRUR 1982, 743, 744 – Verbundene Werke.

[523] Zu den einzelnen Phasen der Softwareentwicklung vgl. die Ausführungen S. 121 ff.

[524] So *Jaeger / Metzger*, Open Source Software, Rn. 58, im Hinblick auf die Integration von GPL-lizenzierten Bibliotheken in ein Programm und dessen Vertrieb als einheitliche Exe-Datei.

[525] So auch *Jaeger / Metzger*, Open Source Software, Rn. 50 ff; die eine formale Trennbarkeit fordern.

Weiteres möglich ein Programm nicht als einheitliche Exe-Datei zu vertreiben, sondern beispielsweise die darin enthaltenen Funktionalitäten in ungebundener Form an den Nutzer zu übergeben und sie erst im Rahmen der Installation zu einer ausführbaren Datei zu verbinden. Infolgedessen ist die faktische Trennbarkeit des Werks zwar Voraussetzung für das Vorliegen mehrerer eigenständiger Werkteile, sie ist allerdings nicht dazu geeignet alleine eine Eigenständigkeit zu begründen.[526]

c.Ohne dadurch unvollständig oder ergänzungsbedürftig zu werden

Für das Vorliegen eines eigenständigen Werks ist vielmehr weiter vorauszusetzen, dass die jeweiligen Komponenten als Folge ihrer Herauslösung nicht unvollständig oder ergänzungsbedürftig werden. [527]

Während bei den klassischen Werkarten eine solche Unvollständigkeit und Ergänzungsbedürftigkeit bereits dann ausgeschlossen werden kann, wenn der gesonderte und eigenständige Vertrieb von Werkteilen zumindest theoretisch denkbar ist,[528] stellt sich die Situation bei Computerprogrammen anders dar. Aufgrund der durch das Internet gegebenen kostengünstigen Vertriebsmöglichkeiten lässt sich in der Praxis bereits heute der Vertrieb von Programmkomponenten, Codefragmenten und selbst einzelner Anweisungsblöcke finden.[529] Insofern ist bereits eine tatsächliche Vertriebsmöglichkeit für sämtliche Programmteile vorhanden, unabhängig davon, ob sie nach der Verkehrsanschauung als eigenständiges Werk anzusehen sind. Würde man für die Beurteilung der Unvollständigkeit und Ergänzungsbedürftigkeit einzig darauf abstellen, ob ein Vertrieb der Werkteile denkbar ist, so würde dies für den Bereich von Computerprogrammen bedeuten, dass selbst einzelne Anweisungen eigenständige Werke sein könnten, sofern sie nur in einer gesonderten Datei vertrieben werden und über ein ausreichendes Maß an Schöpfungshöhe verfügen.

Es ist mithin zu prüfen, unter welchen Voraussetzungen die Unvollständigkeit oder Ergänzungsbedürftigkeit bei Computerprogrammen gegeben ist.

---

[526] Vgl. *Thum* in: Wandtke / Bullinger, § 8 Rn. 8.

[527] KG Schulze Rspr. KGZ 55, 12 – Puppenfee; amtl. Begr. *M. Schulze* Materialien 425; *Loewenheim* in: Schricker, § 8 Rn. 5; *Thum* in: Wandtke / Bullinger, § 8 Rn. 8.

[528] KG Schulze KGZ 55, 12 – Puppenfee; *Loewenheim* in: Schricker, § 8 Rn. 5; ebenso *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 91 f.

[529] Marktplätze, auf denen solche Quellcodeteile verbreitet werden, sind u.a. http://www.krugle.com, http://www.koders.com und http://www.codase.com. Neuerdings bietet auch Google eine umfassende Codesuche an, vgl. http://www.google.com/codesearch.

d.Gesonderte Nutzbarkeit

Bei klassischen Werkarten könnte hierfür ergänzend darauf abgestellt werden, ob das entsprechende Werk gesondert nutzbar ist.[530] Im Hinblick auf Computerprogramme scheint das Kriterium der gesonderten Nutzbarkeit, sofern man es mit der gesonderten Lauffähigkeit gleichsetzt, jedoch wiederum problematisch.

Verdeutlichen lässt sich dies am Beispiel solcher Anwendungen, die gewöhnlich im Betriebssystem enthalten sind. Nimmt man beispielsweise den Internet Explorer, der zusammen mit einem Windows Betriebssystem ausgeliefert wird, so ließe sich zwar zunächst dessen faktische Trennbarkeit festhalten, da er auf der Installations-CD in einem gesonderten Verzeichnis enthalten ist. Fraglich ist aber, ob der Internet Explorer nach dessen Herauslösung unvollständig oder ergänzungsbedürftig wird. Bemisst man dies anhand der Lauffähigkeit des Programms, könnte man daran zweifeln, da der Internet Explorer in der konkreten Form nur auf einem Windows Betriebssystem lauffähig ist. Selbst wenn also ein Herauslösen des Internet Explorers möglich ist, setzt dessen Lauffähigkeit immer wieder eine erneute Zusammenfügung der vormals getrennten Werkteile voraus. Nach klassischem Verständnis könnte man daher zu dem Ergebnis kommen, dass der Internet Explorer durch die Trennung vom Betriebssystem unvollständig und ergänzungsbedürftig wird. Ein Ergebnis, dass offensichtlich im Widerspruch zur Verkehrsanschauung steht.[531]

Wendet man das Kriterium der gesonderten Nutzbarkeit entsprechend auf Computerprogramme an und fordert damit deren eigenständige Lauffähigkeit, besteht das Problem darin, dass sämtliche Programme für ihre Lauffähigkeit, aufgrund der durch die Systemarchitektur vorgegebenen Strukturen, immer funktionelle Abhängigkeiten aufweisen werden, ohne dass diese eine Aussagekraft über deren Eigenständigkeit besitzen. Die Strukturen moderner Betriebssysteme sind vielmehr durch funktionale Abhängigkeiten geprägt, welche ihre Ursachen darin haben, dass sich Betriebssysteme durch einen aus Sicht der Hardware verallgemeinernden Aufbau kennzeichnen.[532] Insofern verfügt heute kaum noch ein Programm über sämtliche Komponenten, die zu seiner Ausführung erforderlich wären. Vielmehr wird beim Nutzer das Vorhandensein gewisser Komponenten – beispielsweise des Betriebssystems – vorausgesetzt. Allgemeine Befehle, auf die alle Anwendungen in gleicher

[530] Immerhin wird eine isolierte Nutzbarkeit bei den klassischen Werkarten stets mit der gesonderten Verwertbarkeit einhergehen.
[531] Vgl. nur *Omsels* in: FS Hertin, 161, mit ausdrücklichem Bezug auf den Internet Explorer.
[532] Diese Eigenschaft wird zumeist als Abstraktion oder auch Veredelung bezeichnet, vgl. *Tanenbaum / Baumgarten*, Moderne Betriebssysteme, S. 14 f.

Weise angewiesen sind, wie z.B. die Möglichkeit Daten zu speichern oder auf die Rechenleistung der Central Processing Unit (CPU) zuzugreifen, werden aus strukturellen Erwägungen nicht in jeder Anwendungen neu integriert, sondern ausgegliedert und für sämtliche Anwendungen als grundlegende Funktionalitäten des Betriebssystems bereitgehalten. Hierdurch kommt es dazu, dass in modernen Betriebssystemen eigentlich keine Anwendung denkbar ist, die vollkommen autonom, also ohne jegliche funktionale Bindung zu weiteren Programmkomponenten auskommt.

Gleichwohl entspricht es der einhelligen Meinung, dass Anwendungen, trotz ihrer funktionalen Abhängigkeit vom zugrundeliegenden Betriebssystem, in aller Regel eigenständige Werkteile sind. Aber nicht nur die Anwendung an sich, sondern auch Teile derselben sollen unter Umständen eigenständige Werkteile sein können. So wird in der Literatur immer wieder darauf verwiesen, dass es der Verkehrsanschauung entsprechen würde, dass sowohl Office-Pakete, als auch Betriebssysteme aus mehreren eigenständigen Werkteilen bestehen.[533] Welche dies sind und wodurch sie sich von unselbstständigen Werkteilen unterscheiden lassen, bleibt indes unbeantwortet.[534]

Allerdings lässt sich bereits festhalten, dass komplexe Programme aufgrund ihres modularen Charakters oftmals mehrere eigenständige und unabhängige Teile beinhalten.[535]

Darüber hinaus wird daraus, dass selbst einzelne Teile eines Office-Pakets als eigenständig und unabhängig gelten können, aber noch etwas anderes ersichtlich. Besteht nämlich die Möglichkeit, dass einzelne Teile von Anwendungsprogrammen eigenständige Werkteile sein können, so bedeutet dies zugleich, dass die für die Lauffähigkeit der Office-Anwendung bestehende funktionale Abhängigkeit der jeweiligen Werkteile vom Vorhandensein eines Betriebssystems nicht dazu führt, dass diese nach ihrer Herauslösung unvollständig oder ergänzungsbedürftig werden.

Mit anderen Worten kann es für die Beurteilung der Unvollständigkeit und Ergänzungsbedürftigkeit nicht darauf ankommen, ob ein Programm völlig losgelöst von anderen Programmen lauffähig ist. Nicht jede funktionale Abhängigkeit führt somit dazu, dass eine Komponente unselbstständiger Teil eines einheitlichen Werks wäre.

---

[533] Vgl. *Jaeger / Metzger*, Open Source Software, Rn. 52; ebenso *Spindler*, Rechtsfragen bei Open Source, C. Rn. 118.

[534] Sowohl *Jaeger / Metzger*, Open Source Software, Rn. 52; als auch *Spindler*, Rechtsfragen bei Open Source, C. Rn. 118; verweisen insofern lediglich darauf, dass diese Annahme als gesichert angesehen werden könne, ohne sich um eine weitergehende Begründung oder eine genauere Abgrenzung der einzelnen Komponenten zu bemühen.

[535] Zu diesem Ergebnis kommt letztlich auch *Koch*, GRUR 2000, 191, 191, 197, wenn er davon ausgeht, dass jedenfalls Komponenten, Module und Klassen eigenständige Werkteile darstellen können.

Im Hinblick auf das Kriterium der gesonderten Verwertbarkeit der Werkteile folgt daraus, dass es jedenfalls nicht mit der gesonderten Nutzbarkeit der Anteile gleichgesetzt werden darf, da diese aufgrund der Abhängigkeit der jeweiligen Programmkomponenten von einem zugrunde liegenden Betriebssystem, letztlich für keine Anwendung gegeben wäre.

Hieraus wird ersichtlich, dass den Kriterien der Unvollständigkeit und Ergänzungsbedürftigkeit im Zusammenhang mit Software eine engere Reichweite zugrunde zu legen ist, als sie im Rahmen der klassischen Werkarten vertreten wird. Eine Anwendung kann folglich nicht nur dann als selbstständig gelten, wenn sie vollkommen autonom, also ohne jegliche funktionale Bindung zu anderen Programmkomponenten lauffähig ist.

Fraglich ist allerdings, wonach unter den vorhandenen funktionalen Abhängigkeiten unterschieden werden kann, ob diese für die Beurteilung der Unvollständigkeit oder Ergänzungsbedürftigkeit relevant, oder lediglich unbeachtliche Folge der Systemarchitektur sind. Die Schwierigkeit besteht dabei offensichtlich darin eine Maßeinheit dafür zu finden, ab wann eine funktionale Abhängigkeit zwischen Programmkomponenten so eng ist, dass sie zu deren Unselbstständigkeit führt und diese mithin lediglich Teile eines größeren Werks sind.

e.Technische Form der Kommunikation

Eine denkbare Vorgehensweise zur Bewertung der funktionalen Bindungen zwischen mehreren Programmkomponenten könnte darin bestehen, die technische Art der Kommunikation zwischen diesen zu untersuchen. Diese wäre relevant, wenn unter den vorhandenen Kommunikationsmöglichkeiten solche auszumachen wären, die zwingende Rückschlüsse auf die Werkeigenschaft der „kommunizierenden“ Programmteile zuließen.
Die Möglichkeit solcher Rückschlüsse wird sowohl von der FSF als auch von der Literatur erörtert. Die FSF weist in diesem Zusammenhang darauf hin, dass ihrer Ansicht nach sowohl die Form der Kommunikation als auch der Inhalt der ausgetauschten Daten relevant sein

dürften.[536] Im Übrigen verweist sie jedoch darauf, dass die Beantwortung dieser Frage durch die Gerichte erfolgen müsse.[537]

In der Literatur werden die Kriterien der FSF überwiegend aufgegriffen und darauf verwiesen, dass etwa sowohl das gemeinsame Laden mit vorbestehendem Code, die Programmausführung in einem Adressraum, sowie die Kommunikationsform zwischen den Programmkomponenten maßgebliche Kriterien seien, wobei die Verwendung von Pipes, Queues, Sockets und Kommandozeilenargumenten für das Vorliegen eines eigenständigen Programms sprechen würden.[538]

Letztlich besteht allerdings Einigkeit darüber, dass die benannten technischen Kriterien allesamt nicht dazu geeignet sind, allgemeingültig zu sein.[539] Dies trifft insofern zu, als den Entwicklern im Rahmen der Softwareerstellung eine Reihe von Variationsmöglichkeiten zustehen, in denen die bezweckten funktionalen Bindungen zwischen den Programmkomponenten sich technisch auf verschiedene Arten verwirklichen lassen. Insofern kann beispielsweise allein aus dem Umstand, dass die Kommunikation zwischen Programmkomponenten über Sockets[540] erfolgt, nicht zwingend auf eine funktionale Bindung zwischen eigenständigen Programmkomponenten geschlossen werden. Immerhin lassen sich Sockets – entgegen ihres eigentlichen Einsatzzweckes – auch lokal verwenden,[541] weshalb sich zumindest nicht ausschließen lässt, dass sie nicht auch für die Kommunikation zwischen unselbstständigen Teilen eines Ganzen eingesetzt werden.

Vergleichbar ist die Sachlage auch bei den weiteren Formen der Interprozesskommunikation.[542] So ist allein die Verwendung von Pipes,[543] named Pipes,[544]

---

[536] Vgl. die Ausführungen der FSF auf der Faq-Seite zur GPL: *... We believe that a proper criterion depends both on the mechanism of communication (exec, pipes, rpc, function calls within a shared address space, etc.) and the semantics of the communication (what kinds of information are interchanged)*, http://www.fsf.org/licensing/licenses/gpl-faq.html#MereAggregation.

[537] Vgl die Faq-Seite der FSF unter http://www.fsf.org/licensing/licenses/gpl-faq.html#MereAggregation: *„...What constitutes combining two parts into one program? This is a legal question, which ultimately judges will decide..."*

[538] *Jaeger / Metzger*, Open-source-Software, Rn. 53; ebenso *Spindler*, Rechtsfragen bei Open Source, C. Rn. 117; für die Maßgeblichkeit des gemeinsamen Ladens von Programmen *Andréewitch*, MR 2005, 240, 242; sowie *Lejeune*, ITRB 2003, 10, 11.

[539] *Jaeger / Metzger*, Open Source Software, Rn. 52; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 118; sowie *Andréewitch*, MR 2005, 240, 242.

[540] *Sockets* funktionieren in der Weise, dass ein Serverprozess mithilfe eines entsprechenden Aufrufs einen Kommunikationspunkt aufbaut, der sodann von verschiedenen Anwendungen (als Client-Prozess) aufgerufen und zum Datenaustausch genutzt werden kann. Bei Sockets handelt es sich somit um das Ende einer Kommunikationsschnittstelle zwischen zwei Programmen, welche Daten über ein Netzwerk austauschen. Weiterführende Erläuterungen und Einzelheiten zu Sockets *Gulbins / Obermayr*, AIX UNIX, S. 149 f.

[541] Ausdrücklich weist hierauf *Gulbins / Obermayr*, AIX UNIX, S. 149 Fn. 1, hin.

[542] Zum Begriff und Umfang der Interprozesskommunikation, *Herold*, Linux-Unix-Systemprogrammierung, S. 549 ff; sowie in einem weiteren Verständnis *Heuer*, Unix-Systemadministration, S. 102 ff, der nicht nur die lokalen Formen der Interprozesskommunikation betrachtet.

Message Queues, shared memory oder auch remote procedure calls[545] nicht dazu geeignet zwingende Rückschlüsse auf die Eigenständigkeit der kommunizierenden Programmkomponenten zuzulassen. Der Grund dafür liegt wiederum darin, dass sich auch diese Kommunikationswege sowohl innerhalb eines Programms als auch zwischen eigenständigen Werken verwenden lassen. Selbst wenn also named Pipes, shared memory und remote procedure calls üblicherweise für die Kommunikation eigenständiger Werkteile verwendet werden, schließt ihr Einsatz nicht aus, dass sie nicht faktisch auch programmintern verwendet werden können. Für einen Entwickler stellt es kein wesentliches Hindernis dar, den maßgeblichen Datenaustausch innerhalb eines Programms beispielsweise über einen shared memory zu verwirklichen, obwohl hierfür auch eine einfache Pipe ausgereicht hätte.

Insofern muss an dieser Stelle festgehalten werden, dass allein der Umstand, dass ein bestimmtes Kommunikationsmittel verwendet wurde, nicht dazu geeignet ist, einen zwingenden Rückschluss auf die Werkeigenschaft der beteiligten Komponenten zu ermöglichen. Fraglich ist allerdings, ob aus dieser Feststellung zwangsläufig folgt, dass für die Beurteilung der Eigenständigkeit eines Werks im Übrigen ausschließlich auf die Verkehrsanschauung abgestellt werden kann. Dies wird zwar überwiegend in der Literatur vertreten,[546] führt aber vor dem Hintergrund zu Problemen, dass sich weder anhand der benannten technischen Kriterien noch mit dem Verweis auf die Verkehrsanschauung eine konkrete Grenzziehung der Lizenzierungspflicht vollziehen lässt. Es überrascht daher nicht, dass in der Praxis ein hohes Maß an Rechtsunsicherheit herrscht, wenn es um die Reichweite der Lizenzierungspflicht geht. Für den Anwender bleibt die Frage, ob die vorhandenen funktionalen Abhängigkeiten zwischen den Komponenten dazu führen, dass diese als unselbstständige Teile eines größeren Ganzen zu betrachten sind.

---

[543] Pipes stellen einen einfachen (unidirektionalen) Mechanismus zur Verfügung, der einen Datenaustausch zwischen zwei Prozessen über einen systeminternen Puffer erlaubt. Eine Pipe verhält sich dabei für die Prozesse wie eine Datei, in die der eine Prozess Daten schreibt und die vom anderen Prozess ausgelesen werden können. Einzelheiten unter *Herold*, Linux-Unix-Systemprogrammierung, S. 550 f.

[544] Im Gegensatz zu *Pipes* stellen *named pipes* eine Erweiterung dar, da sie einen externen Namen besitzen, über den sie von beliebigen Prozessen angesprochen werden können. Einzelheiten anschaulich beschrieben bei *Brause*, Kompendium der Informationstechnologie, S. 77 f.

[545] Der remote procedure call ist ein Mechanismus, der dazu dient, die Programmierung verteilter Anwendungen zu unterstützen. Bekannte Implementierungen von rpc sind der RPC von Sun, der DCE-RPC der Open Software Foundation, XML-RPC sowie SOAP, vgl. *Dadam*, Verteilte Datenbanken und Client/Server-Systeme, S. 292 f; Einzelheiten zu remote procedure calls außerdem bei *Brause*, Kompendium der Informationstechnologie, S. 79 ff.

[546] Vgl *Jaeger / Metzger*, Open Source Software, Rn. 52; *Spindler*, Rechtsfragen bei Open Source, C. Rn. 118. .

f.Anderweitige Verwendbarkeit der Komponente

Um eine weitergehende Differenzierung der bestehenden funktionalen Abhängigkeiten zu ermöglichen, könnte deshalb – entsprechend der Auffassung zu den klassischen Werkarten[547] – darauf abzustellen sein, ob es zumindest theoretisch möglich ist, die betreffenden Programmkomponenten nach deren Trennung wieder anderweitig zu verwenden. Dabei ist die anderweitige Verwendungsmöglichkeit, aufgrund der dargestellten funktionalen Abhängigkeiten innerhalb moderner Betriebssysteme, jedoch nicht im Sinne einer losgelösten eigenständigen Nutzbarkeit, sondern im Sinne einer anderweitigen Integrationsmöglichkeit, zu verstehen. Auch im Hinblick auf die klassischen Werkarten wurde bereits vereinzelt die Auffassung vertreten, dass eine gesonderte Verwertbarkeit des Werks auch dann vorliegen könne, wenn die Verwertbarkeit dadurch erreicht wird, dass das übrige Werk durch etwas Gleichartiges ersetzt wird.[548] Größere Einigkeit herrscht hingegen, wenn die einzelnen Werkteile zwar nicht allein, aber doch ohne die konkreten anderen Teile und ohne einen gleichartigen Ersatz verwertet werden können. In diesen Fällen wird im Hinblick auf die klassischen Werkarten zum Teil von einer gesonderten Verwertbarkeit ausgegangen.[549]

Ein ähnlicher Ansatz lässt sich im Zusammenhang mit Computerprogrammen teilweise auch in der Literatur finden, wonach ein Programm, welches nicht nur mit dem GPL-lizenzierten Programmen sondern auch mit anderer Software ablauffähig ist, wohl als inhaltlich selbstständig angesehen werden soll.[550] Entscheidendes Kriterium für die Unabhängigkeit wäre somit die anderweitige – nicht aber die isolierte – Einsetzbarkeit einer Komponente.

Sofern diese gegeben ist, kann hieraus geschlossen werden, dass die betreffende Komponente im Hinblick auf die bestehenden funktionalen Bindungen variabel ist. Die Komponente könnte zwar nicht isoliert, aber doch ohne die konkreten anderen Teile verwendet werden. Um ihre Funktionalität zu entfalten, ist sie folglich auf keine der vorhandenen Komponenten zwingend angewiesen, wodurch ein hohes Maß an Eigenständigkeit ersichtlich wird,[551] dass

---

[547] Vgl. nur *Loewenheim* in: Schricker, § 8 Rn. 5 m.w.N.

[548] So im Zusammenhang mit Filmmusik und dem restlichen Filmwerk, *Berthold*, Filmrecht, S. 15 f; *Hertin* in: Nordemann / Fromm, § 88 Rn. 6, hinsichtlich Filmarchitekten, Dekorateuren, Kostümbildnern und Choreografen; a.A. *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 62 f.

[549] *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 93; a.A. v. Gamm § 8 Rn. 11; sowie *Loewenheim* in: Schricker, § 8 Rn. 6, die eine gesonderte Verwertbarkeit nur dann annehmen, wenn die einzelnen Beiträge ohne weitere Ergänzung oder Umgestaltung verkehrsfähig sind; ebenso auch *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 63, der die Unabhängigkeit im Hinblick auf einen gegenüber dem Restwerk gleichartigen Ersatz verlangt.

[550] Vgl. *Jaeger / Metzger*, Open Source Software, Rn. 53, die hierin ein überzeugendes Argument für die inhaltliche Selbstständigkeit des Programms sehen wollen.

[551] So auch *Jaeger / Metzger*, Open Source Software, Rn. 53, die bei einer sinnvollen anderweitigen Verwendbarkeit davon ausgeht, dass hierdurch die Selbstständigkeit des betreffenden Softwaremoduls augenfällig wird.

es rechtfertigt, die entsprechende Komponente, trotz der durch den Aufbau moderner Betriebssysteme bedingten funktionalen Abhängigkeiten, als eigenständiges Werk zu bewerten.

g.Technische Voraussetzungen der anderweitigen Verwendbarkeit

Wann aber lässt sich eine Komponente anderweitig verwenden? In technischer Hinsicht müssen hierfür einige Voraussetzungen gegeben sein. So benötigt man zunächst Kenntnis von der Funktionalität der Komponente. Weiterhin muss die Kommunikationsform bekannt sein; man benötigt also Kenntnis davon, in welcher Form die Funktionalität der Komponente angesprochen werden kann, ob Daten übergeben werden können, welche Form sie haben müssen und auf welche Weise die Komponente ggf. Daten zurück gibt.

Insgesamt sind also sämtliche Informationen über die Komponente erforderlich, welche gewöhnlich in einer dokumentierten Schnittstelle bereitgehalten werden. Fehlen diese Informationen, ist jedenfalls im proprietären Bereich, die Kommunikation und damit auch eine anderweitige Einsetzbarkeit der Komponente ausgeschlossen.[552] Dies gilt unabhängig davon, ob die Komponente die Schnittstelle selbst bereithält oder lediglich dazu ausgestaltet ist, über eine solche zu kommunizieren.

Das gleiche Ergebnis wird auch dadurch erreicht, dass zwar dokumentierte Schnittstelleninformationen bereitgehalten werden, diese jedoch nicht dazu geeignet sind, anderen einen Zugriff auf die Funktionalitäten der Komponente zu ermöglichen, weil beispielsweise die Komponente die Daten ausschließlich in einer unbekannten Form[553] annimmt oder zurückgibt und dadurch die anderweitige Nutzbarkeit ausgeschlossen ist. Es reicht also nicht allein, dass eine dokumentierte Schnittstelle gegeben ist. Diese muss vielmehr in einer generischen Art und Weise gestaltet sein, so dass beliebige Komponenten hierüber auf sinnvolle Weise kommunizieren können.

Sofern die Komponente dabei die Schnittstelle selbst bereithält, folgt ihre anderweitige Einsetzbarkeit daraus, dass sie für ihre eigene Funktionalität auf keine der Komponenten angewiesen ist und sich aufgrund der allgemein zugänglichen Schnittstelle auch mit beliebigen anderen Komponenten koppeln lässt, soweit diese die bereitgestellte Schnittstelle unterstützen.

---

[552] Allerdings bestünde aufgrund von § 69 e UrhG die Möglichkeit zur Dekompilierung des Programms, um so an die Schnittstelleninformationen zu gelangen.

[553] Hierbei ist insbesondere an einen verschlüsselten Datenaustausch oder unbekannte Datenformate zu denken.

Hält die betreffende Komponente hingegen keine Schnittstelle bereit, sondern ist darauf ausgerichtet über eine solche zu kommunizieren, ist ihre anderweitige Einsetzbarkeit zunächst einmal eindeutig gegeben, wenn sie ohne Veränderungen auch mit anderen Komponenten einzusetzen ist. Dies ist immer dann möglich, wenn diese Komponenten über die gleiche Schnittstelle verfügen.

Aber selbst wenn geringfügige Umstellungen oder Veränderungen am Werk erforderlich sein sollten, besteht im Hinblick auf die klassischen Werkarten Einigkeit darüber, dass diese einer gesonderten Verwertbarkeit des Werks nicht entgegenstehen.[554] Überträgt man dies auf das Umfeld von Computerprogrammen, kann von einer anderweitigen Einsetzbarkeit auch dann ausgegangen werden, wenn für die Verbindung zwischen Komponenten kleinere Anpassungen der Schnittstelle erforderlich sind, ihre eigentliche Funktionalität aber unverändert bleibt.[555]

Letztlich sind daher sowohl die Komponente, welche eine generische Schnittstelle bereithält, als auch jene, die hierüber eingebunden werden kann, anderweitig einsetzbar. Als maßgebliches Kriterium für eine gesonderte Verwertbarkeit einer Komponente könnte mithin das Bereithalten einer dokumentierten Schnittstelle festzuhalten sein.

Im Umfeld proprietärer Entwicklungen ist dieses Kriterium dazu geeignet, die anderweitige Einsetzbarkeit einer Komponente zu bestimmen. Immerhin lässt sich eine Komponente im Umfeld proprietärer Entwicklungen – mangels Kenntnis des Quellcodes – ausschließlich dann anderweitig einsetzten, wenn diese die erforderlichen Schnittstelleninformationen bereithält. Der Umstand, dass dokumentierte Schnittstelleninformationen bereitgehalten werden, lässt daher den Schluss zu, dass die betreffende Komponente darauf ausgerichtet ist, auch außerhalb der spezifischen Anwendung einsetzbar zu sein.

Bedenken daran, das Bereithalten einer dokumentierten Schnittstelle als maßgebliches Kriterium zu verwenden, könnten einzig deshalb bestehen, da sich Open Source Komponenten, im Gegensatz zu proprietären Entwicklungen, dadurch auszeichnen, dass der gesamte Quellcode der Komponenten öffentlich zugänglich ist. Gilt es mithin die Eigenständigkeit einer Open Source Komponente zu beurteilen, könnte das Erfordernis einer dokumentierten Schnittstelle dadurch entkräftet werden, dass sich die erforderlichen Schnittstelleninformationen auch durch eine Quellcodeanalyse herausfinden lassen und eine dokumentierte Schnittstelle somit keine Voraussetzung für die anderweitige Einsetzbarkeit

[554] Vgl. *Siefert*, Die Abgrenzung von Werkeinheit und Werkmehrheit im Urheberrecht, S. 93; und auch *Stroh*, Werkeinheit und Werkmehrheit im Urheberrecht, S. 64, im Hinblick auf eine in einem Roman erzählte Binnengeschichte.

[555] So auch *Jaeger* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 2 Rn. 26, der allerdings neben der Schnittstellenanpassung auch weitere Portierungserfordernisse für unschädlich hält.

der Komponente ist. Eine Kommunikation wäre also faktisch selbst dann möglich, wenn die Komponente offensichtlich unselbstständig ist, weil sie ausschließlich darauf ausgerichtet ist innerhalb einer bestimmten Anwendung gewisse Funktionalitäten zu erfüllen, und als Folge davon keine generischen Schnittstelleninformationen bereithält.
Würde man diesen Umstand zum Anlass nehmen, um für die Beurteilung der anderweitigen Einsetzbarkeit auf das Vorliegen einer dokumentierten Schnittstelle zu verzichten, hätte dies zur Folge, dass letztlich jede beliebige Komponente – zumindest theoretisch – anderweitig einsetzbar wäre, sofern nur deren Quellcode bekannt und die erforderlichen Schnittstelleninformationen dadurch zu ermitteln sind.
Hierauf kann es aber nicht ankommen, wenn es um die Beurteilung der Eigenständigkeit eines Werks geht. Dies liegt daran, dass die anderweitige Einsetzbarkeit in solchen Fällen nicht durch Eigenschaften des Werks, sondern einzig dadurch begründet wird, dass Dritte aufgrund ihrer technischen Fähigkeit dazu in der Lage sind, die benötigten Informationen zu ermitteln. Für die Beurteilung der Eigenständigkeit eines Werks kommt es aber nicht auf diese Fähigkeiten, sondern einzig auf dessen konkrete Ausgestaltung an. Zeichnet sich dieses dadurch aus, dass dokumentierte Schnittstelleninformationen bereitgehalten werden, um einen generellen Zugriff zu ermöglichen, so spricht dies dafür, dass die Komponente auch anderweitig verwendet werden kann. Fehlen hingegen diejenigen Informationen, die für eine Kommunikation mit der entsprechenden Komponente erforderlich wären, so wird hieraus ersichtlich, dass die Komponente unselbstständiger Teil eines größeren Ganzen ist.
Aus technischer Sicht ist es daher für eine anderweitige Einsetzbarkeit unerlässlich, dass eine Komponente sämtliche Schnittstelleninformationen bereithält, die dazu erforderlich sind, beliebigen anderen Komponenten den Zugriff auf die spezifischen Funktionalitäten zu ermöglichen.

aa)Plugins /Shared Libraries

Diese Voraussetzungen könnten zunächst zwischen Anwendungen und Plugins[556] gegeben sein. Aus funktionaler Sicht zeichnen sich Plugins dadurch aus, dass sie einer Anwendung über eine darin enthaltene Schnittstelle Funktionalitäten hinzufügen. Voraussetzung dafür, dass von einem Plugin ausgegangen werden kann, ist somit, dass die eigentliche Anwendung durch das Plugin unverändert bleibt und dessen Einbindung über eine, in der Anwendung für

[556] Der Begriff der Plugins wird vorliegend weit verstanden, so dass hiervon auch Add-ons und Makros umfasst werden.

diesen Zweck bereitgehaltene, Schnittstelle erfolgt.[557] Der wesentliche Unterschied zwischen Plugins und sonstigen unselbstständigen Programmerweiterungen liegt somit in der Art und Weise, wie diese technisch realisiert werden.

Damit gewährleistet ist, dass sowohl Anwendung als auch Plugin anderweitig einsetzbar sind, muss sich die in der Anwendung vorhandene Schnittstelle zudem dadurch auszeichnen, dass sie durch beliebige Plugins verwendet werden kann. Die Schnittstelle darf demnach über keine Spezifikationen verfügen, die es lediglich einzelnen Plugins ermöglicht hierüber eine sinnvolle Kommunikation aufzubauen.

Diese Voraussetzungen werden in der Praxis regelmäßig erfüllt. Zahlreiche Anwendungen verfügen mittlerweile über dokumentierte Schnittstellen, um Funktionserweiterungen durch Plugins zuzulassen. Gerade im Bereich von Grafik- und Audioanwendungen haben sich dabei auch schon erste Standardisierungen der Anwendungsschnittstellen durchgesetzt,[558] so dass – nicht nur theoretisch sondern auch faktisch – die Möglichkeit besteht die jeweiligen Plugins unverändert in anderen Anwendungen einzusetzen.[559]

Hierdurch wird ersichtlich, dass sowohl die Plugins als auch die Anwendung – unabhängig voneinander – anderweitig einsetzbar sind und somit nach der Verkehrsanschauung eigenständige Werkteile darstellen. Bestätigt wird dieses Ergebnis zudem dadurch, dass auch in der Praxis regelmäßig ein getrennter Vertrieb von Plugins und Anwendungen besteht, weshalb auch von einer gesonderten Verwertbarkeit – im Sinne eines gesonderten Vertriebs – der Werkstücke auszugehen wäre.

Die Abgrenzung zwischen Plugins und unselbstständigen Programmerweiterungen könnte allerdings dann problematisch sein, wenn einer GPL-lizenzierten Anwendung erst eine entsprechende Plugin-Schnittstelle hinzugefügt oder die Vorhandene verändert werden muss, um eine Einbindung der Erweiterung zu ermöglichen. Welche Programmkomponenten sind in diesen Fällen unselbstständige Programmveränderung und welche stellen daneben bereits eigenständige Plugins dar? Die Grenzziehung zwischen eigenständigem Plugin und unselbstständiger Bearbeitung ist hier schwierig und wird letztlich in jedem Fall gesondert zu untersuchen sein.

[557] Technisch werden Plugins zumeist als shared libraries verwirklicht. Ihre Einbindung in die Anwendung erfolgt dabei häufig durch ein Verfahren, dass als Inversion of control bezeichnet wird. Hierbei wird das Plugin zunächst bei der Anwendung registriert und sodann von dieser eigenständig hierauf zugegriffen.

[558] Im Audio-Bereich ist hier vor allem an die VST-Schnittstelle und im Grafik-Bereich an die Adobe Plug-in-Schnittstelle zu denken.

[559] So ist es bereits möglich Plugins, die für das Grafik-Programm Photoshop entwickelt wurden, ohne Veränderung u.a. in den Programmen Photoshop Elements, Paint Shop Pro, Photo Impact, Corel Draw, Corel Photo-Paint und Picture Publisher zu verwenden.

Festgehalten werden kann allerdings, dass es für eine Abgrenzung darauf ankommt, welche der vorhandenen Programmkomponenten zur Anpassung oder Erstellung der Schnittstelle erforderlich sind, um die Integration des Plugins zu ermöglichen. Insofern ist zu untersuchen, welchen Zustand das Plugin beim Anwendungsprogramm voraussetzt. Alle Programmkomponenten, die zur Anpassung oder Erstellung der Schnittstelle erforderlich werden, sind sicherlich dem Anwendungsprogramm als Ganzes zuzurechnen und folglich lizenzrechtlich entsprechend zu behandeln.
Um zudem zu gewährleisten, dass es sich bei der Programmerweiterung um ein Plugin und nicht etwa um eine unselbstständige Programmveränderung handelt, muss die Programmveränderung in einer Art und Weise erfolgt sein, die für Plugins typisch ist. Insofern muss die Schnittstelle auch dazu geeignet sein, beliebige andere Erweiterungen zuzulassen. Hierfür reicht es nicht aus, dass zumindest irgendeine weitere Erweiterung über diese Schnittstelle denkbar ist, vielmehr muss die Schnittstelle eine sinnvolle Nutzung für eine unbestimmte Zahl von Erweiterungsmöglichkeiten unterstützen.
Sind diese Voraussetzungen nicht gegeben, kann die Schnittstelle insofern ausschließlich von *einem* Plugin sinnvoll angesprochen werden, so spricht dies dafür, dass das Plugin kein eigenständiger Werkteil ist, sondern dem Werk als Ganzes zuzurechnen wäre.
Wird die Anwendung hingegen in einer Art und Weise angepasst, dass sie künftig über eine dokumentierte und vielfältig zu verwendende Schnittstelle verfügt, so spricht dies eindeutig dafür, dass das Plugin gegenüber der Anwendung eigenständigen Charakter hat. Immerhin besteht im Hinblick auf das Plugin kein Unterschied zu den Fallgruppen, in denen ein solches zusammen mit einem Programm vertrieben wird, das bereits über die erforderliche Schnittstelle verfügte. Das Plugin kann ohne Anpassungen in anderen Anwendungen verwendet werden, sofern diese die entsprechende Schnittstelle unterstützen.

bb)Middleware

Die Voraussetzung der anderweitigen Einsetzbarkeit von Komponenten könnte weiterhin zwischen Anwendungen und der sog. Middleware bestehen. Unter Middleware wird dabei vorliegend die Ansammlung von anwendungsunabhängigen Technologien verstanden, welche Dienstleistungen zur Vermittlung und Verteilung von Daten und Meldungen zwischen

Anwendungen unterstützen.[560] Im Unterschied zur „klassischen“ Anwendung dient Middleware nicht dazu Benutzereingaben unmittelbar zu verwerten und verfügt mithin auch über keine Benutzerschnittstelle. Vielmehr wird die anwendungsübergreifende Kommunikation durch eine dokumentierte Anwendungsschnittstelle[561] ermöglicht, über welche der Zugriff beliebiger Anwendungen erfolgen kann.

Insofern sind diejenigen Programmkomponenten, die der Middleware zuzurechnen sind, für ihre Funktionsfähigkeit auch nicht von einzelnen Anwendungen abhängig. Die einzige „Verbindung“ zwischen Anwendungen und Middleware besteht in der von der Middleware hierfür bereitgestellten Schnittstelle, über die Daten – mittels Messaging – ausgetauscht oder andere Anwendungen über sog. Remote Procedure Calls (rpc)[562] aufgerufen werden können.

Von einer anderweitigen Einsetzbarkeit der Middlewarelösung wird daher regelmäßig auszugehen sein, sofern die dokumentierte Schnittstelle wiederum eine abstrakte Form aufweist und folglich keine Nutzungsbeschränkungen für einzelne Anwendungen enthält.

Liegen diese Voraussetzungen vor, ist zugleich auch die anderweitige Einsetzbarkeit der jeweiligen Anwendung gewährleistet, da auch diese mit jeder anderen Middlewarelösung einsatzfähig wäre, sofern diese die gleiche Schnittstelle bereithält. In der Praxis haben sich dabei mit XML-RPC, CORBA und SOAP in der Vergangenheit bereits einige Standardisierungen etabliert, die eine anderweitige Einsetzbarkeit der darauf zugreifenden Komponenten gewährleisten.

Nach der Verkehrsanschauung stellen daher sowohl die Anwendung als auch die Middleware voneinander abgrenzbare eigenständige Werkteile dar. Bestätigt wird dieses Ergebnis letztlich auch wieder durch die Vertriebspraxis, in der Middleware und Anwendungen von unterschiedlichen Unternehmen[563] entwickelt und getrennt vertrieben werden.

[560] Vgl. *Ruh / Maginnis, u.a.*, Enterprise application integration, S. 3 ff; *Frielingsdorf*, Einfache IT-Systeme, S. 216.

[561] Im Gegensatz zur Benutzerschnittstelle wird diese Schnittstelle, da sie ausschließlich für die Kommunikation mit Anwendungen ausgestaltet ist, auf Quelltextebene als Application Interface (API), sowie auf Binärebene als Application Binarie Interface (ABI) bezeichnet.

[562] Vgl. S. 144 Fn. 545.

[563] Beispiele für proprietäre Middlewarehersteller sind beispielsweise ORACLE (ORACLE Fusion Middleware), SAP (SAP Exchange Infrastructure), Borland (Visionbroker) und IBM (WebSphere Application Server).

cc)Anwendungszugriff über das User Interface

Eine funktionale Abhängigkeit zwischen Programmkomponenten könnte deren Eigenständigkeit weiterhin dann nicht entgegenstehen, wenn sie über eine sog. Benutzerschnittstelle miteinander verbunden sind.

Als Benutzerschnittstelle wird derjenige Teil einer Anwendung bezeichnet, den diese für den Datenaustausch mit dem Nutzer bereithält.

Verbreitete Formen von Benutzerschnittstellen sind sowohl das Command Line Interface (CLI),[564] das Text User Interface (TUI),[565] das Web-based User Interface (WUI)[566] oder auch das Graphical User Interface (GUI).[567] Eine funktionale Abhängigkeit zwischen verschiedenen Programmkomponenten über eine solche Schnittstelle kommt beispielsweise immer dann vor, wenn proprietäre Anwendungen, anstelle des Nutzers, über die Benutzerschnittstelle mit einer GPL-lizenzierten Anwendung kommunizieren. Dies erfolgt zumeist bei solchen Anwendungen, die über keine eigene GUI verfügen, sondern für den Nutzer lediglich das CLI, das TUI oder auch Web-based User Interfaces bereithalten. Diese Schnittstellen sind für viele Nutzer schwer zu bedienen, da sie die Eingabe von Befehlen und Parametern erfordern. Aufgrund dessen knüpfen zahlreiche Anwendungen, anstelle des Nutzers, an diese Schnittstelle an und ersetzen für diesen die textuelle Eingabe der Befehle und Parameter.[568] Dem Nutzer wird, anstelle der vom eigentlichen Programm vorgesehenen Schnittstelle, zumeist eine bedienfreundlichere GUI geboten, in welcher dieser die entsprechenden Parameter auswählen kann. Neben solchen rein „vermittelnden“ Funktionen, deren urheberrechtliche Schutzwürdigkeit mitunter zweifelhaft sein könnte, beinhalten diese Anwendungen häufig auch noch Erweiterungen, durch die zusätzliche Funktionalitäten – gegebenenfalls auch solche anderer Anwendungen – integriert und miteinander kombiniert werden. Der Benutzer bedient dadurch letztlich nur die GUI einer Anwendung, um hierdurch

[564] Das Command Line Interface erfordert vom Benutzer die Eingabe der entsprechenden Befehle in Textform. Ein Beispiel für CLIs sind Shells.

[565] Ein Beispiel für eine solche Zeichenorientierte Benutzerschnittstelle ist der Dateiverwalter Norton Commander, der vom Benutzer keine Befehlseingaben erfordert, sondern sich in Form von Menüs, darstellt, die sich zumeist ausschließlich mit der Tastatur und teilweise auch mit der Maus bedienen lassen.

[566] Web-based user interfaces können vom Benutzer über dessen Browser bedient werden. Hierüber gibt der Benutzer Daten ein. Das Programmergebnis wird zumeist als Website ausgegeben, die wiederum über den Browser visualisiert wird.

[567] Die grafischen Benutzeroberflächen wie beispielsweise KDE oder Aqua entsprechen wohl dem, was sich die meisten Benutzer unter einer Benutzerschnittstelle vorstellen. Sie lassen sich, üblicherweise mit der Maus, optional aber auch mit anderen Eingabegeräten bedienen.

[568] Solche Programme existieren nahezu in jedem Bereich, vgl. beispielsweise das Concurrent Version System „cvs“, für das mit *TortoiseCVS* (http://www.tortoisecvs.org), *WinCVS* (http://www.wincvs.org), *MacCVS* (http://www.maccvs.org) und *Cervisia* (http://cervisia.kde.org/) zahlreiche grafischen Benutzeroberflächen existieren.

auf die Funktionalitäten mehrerer Anwendungen über deren Benutzerschnittstellen zuzugreifen.
Hält eine Anwendung mithin eine dokumentierte Benutzerschnittstelle bereit, so kann diese – abhängig von ihrer Ausgestaltung – nicht nur vom Benutzer, sondern zugleich auch von beliebigen anderen Anwendungen verwendet werden, ohne dass die eigentliche Anwendung hierfür verändert werden müsste.

Hieraus folgt, dass die zugrunde liegende Anwendung, im Hinblick auf die zugreifenden Komponenten eigenständig ist. Sie entfaltet ihre Funktionalität unabhängig davon, ob ein Benutzer oder eine Anwendung auf die bereitgestellte Schnittstelle zugreift. Insofern ist auch eine anderweitige Einsetzbarkeit dieser Anwendung gegeben.
Aber auch im Hinblick auf die zugreifenden Komponenten ist von deren Eigenständigkeit auszugehen. Immerhin ist es auch hier denkbar, dass die zugrunde liegende Anwendung durch eine andere ausgetauscht wird und die Funktionalität gleichwohl erhalten bleibt.[569] Voraussetzung ist einzig, dass auch die neue Anwendung die gleiche Benutzerschnittstelle bereithält und hierüber mit den gleichen Funktionsaufrufen angesprochen werden kann. Selbst wenn in der Praxis diejenigen Fälle, in denen verschiedene Anwendungen nicht nur über die gleiche Benutzerschnittstelle, sondern auch über gleichen Befehls- und Parameterbezeichnungen verfügen, eher selten sind, wird hieraus die theoretische Austauschbarkeit der zugrunde liegenden Anwendung ersichtlich.
Letztlich ist eine Austauschbarkeit der zugrunde liegenden Anwendung aber auch dann ohne größeren Aufwand möglich ist, wenn sich die jeweiligen Anwendungen lediglich in der Befehlssyntax nicht aber in ihrer Funktionalität unterscheiden. In diesen Fällen reicht es für eine anderweitige Einsetzbarkeit aus, die Schnittstelle in der Weise zu verändern, dass die Befehlssyntax an die neue Anwendung angepasst wird.[570]
Nach der Verkehrsanschauung sollten daher sowohl die Anwendung, welche die Benutzerschnittstelle bereithält, als auch die Programmkomponenten, die hierüber zugreifen, als abgrenzbare und eigenständige Werkteile anzusehen sein. Dieses Ergebnis wird letztlich wiederum durch die unterschiedlichen Vertriebswege bestätigt, die sich in der Praxis zwischen solchen Anwendungen vorfinden lassen.

[569] Beispiele hierfür sind die Mailprogramme Sendmail, Postfix und Exim sowie die Routing Deamon Zebra, Quagga und Cisco IOS.
[570] Eine solche Syntaxänderung wäre ohne weiteres durch einen sog. Wrapper zu automatisieren, der im Grunde genommen nichts anderes machen würde, als das klassische „Suchen und Ersetzen"- Programm in einer Textverarbeitung. Die alte Schnittstelle der Komponente würde hierdurch durch eine Neue ersetzt, vgl. *Masak*, Legacysoftware, S. 96 f.

dd)Kernelschnittstelle - Systemmodule

Mit der Kernelschnittstelle könnte eine weitere Schnittstelle gegeben sein, die für die Beurteilung der Eigenständigkeit der Werkteile aussagekräftig ist.

Die Kernelschnittstelle wird dabei vom eigentlichen Betriebssystemkern für die Kommunikation mit den sog. Systemmodulen bereitgehalten. Beim Betriebssystemkern handelt es sich um den Teil des Betriebssystems, der die zentralen Aufgaben, wie z.B. die Prozess- und Speicherverwaltung oder auch elementare Ein- und Ausgaben[571] erledigt. Aus lizenzrechtlicher Sicht ist insbesondere der GPL-lizenzierte Betriebssystemkern von Linux bedeutsam. Dieser wird häufig in einer monolithischen Form vertrieben,[572] weshalb der gesamte Code des Kerns – einschließlich seiner Teilsysteme wie beispielsweise Speicherverwaltung, Dateisysteme oder Gerätetreiber – zusammengefügt aus einer einzigen Datei bestehen.[573] Eine weitergehende Untergliederung zwischen Betriebssystemkern und Systemmodulen scheidet in diesen Fällen bereits aufgrund der fehlenden Herauslösbarkeit einzelner Werkteile aus.

Andererseits lassen sich vom Betriebssystemkern diejenigen Systemmodule abgrenzen, die zum Vertriebszeitpunkt noch nicht Fest in den Kern einkompiliert wurden, weil beispielsweise die Kompilierung erst beim Nutzer stattfinden soll, oder weil sie erst während des laufenden Betriebs dynamisch hinzugeladen werden (sog. loadable kernel modules).[574]

Aus funktionaler Sicht unterscheiden sich die statischen nicht von den loadable Kernel Modulen.[575] Beide werden integraler Bestandteil des Kernels und kommen als solche innerhalb des Kernelspace[576] zur Ausführung. Die Auslagerung einzelner Funktionalitäten aus dem Kern in externe Systemmodule ist in der Vergangenheit ausschließlich deshalb erfolgt,

---

[571] *Achilles*, Betriebssysteme, S. 2.

[572] Allerdings ist nicht der gesamte Linux-Kernel monolithisch, sondern nur der Kern, in den häufig zumindest die Treiber für den Zugriff auf das Dateisystem fest eingebunden sind; vgl. *Heuer*, Unix-Systemadministration, S. 3 f.

[573] Eine feste Einbindung von Systemmodulen findet sich zumeist bei den Standard Linux Kernelimage Paketen, bei denen es nicht vorgesehen ist, dass der Nutzer den Kern im Rahmen der Installation kompilieren muss; vgl. die Distributionspakete von Suse, Red Hat, Mandriva und Ubuntu. Unabhängig von solchen Paketen sind im Kernel aber jedenfalls solche Module statisch eingebunden, die zwingend für den Systemstart erforderlich sind. Dies sind beispielsweise die Treiber, die für den Zugriff auf das Dateisystem nötig sind, vgl. *Mauerer*, Linux Kernelarchitektur, S. 3.

[574] Zum Laden eines Moduls reicht es unter Linux dieses mit dem Befehl insmod bzw. modprobe Aufzurufen. Danach lassen sich Systemmodule während der Laufzeit durch den Befehl modprobe oder auch rmmod wieder aus dem Kernel entladen, vgl. *Ganten / Alex*, Debian GNU, Linux, S. 427 f; *Mauerer*, Linux Kernelarchitektur, S. 3.

[575] Bei den meisten Modulen kann zudem der Nutzer darüber entscheiden, ob er es fest in den Kernel einbinden oder dynamisch hinzuladen will.

[576] Unter Linux wird zwischen Kernelspace und Userspace unterschieden. Im Gegensatz zu Anwendungsprogrammen besitzen Kernelmodule die Besonderheit, dass sie nicht wie „gewöhnliche“ Programme im Userspace ausgeführt werden, sondern dem Kernel durch die gemeinsame Ausführung im gleichen Speicherbereich erheblich näher stehen.

um dessen Flexibilität zu erhöhen.[577] Abhängig von der vorhandenen Hardware reicht es nunmehr aus, bestimmte Funktionalitäten (z.B. Treiber) dynamisch hinzuzuladen, ohne dass hierfür eine erneute Kompilierung und die damit einhergehende Veränderung des Kerns erforderlich wird.

Die Einbindung von Systemmodulen in den Kern erfolgt dabei, unabhängig davon, ob sie statisch oder dynamisch vollzogen wird, über die sog. Kernelschnittstelle, mit der beliebigen Modulen ein sinnvoller Zugriff ermöglicht wird.

Im Hinblick auf die funktionalen Abhängigkeiten zwischen dem Kern und den jeweiligen Systemmodulen lässt sich aufgrund dieser Ausgangslage zunächst festhalten, dass der Kern durch Kernelmodule lediglich um zusätzliche Funktionalitäten bereichert wird, ohne aber für seine eigene Lauffähigkeit zwingend auf diese angewiesen zu sein. Aus urheberrechtlicher Sicht stellt somit der Kern ein eigenständiges Werk dar.

Im Zusammenhang mit der Regelung in Ziff. 2 Abs. 2 S. 2 GPL bleibt allerdings zu klären, ob die entsprechenden Systemmodule unter die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL fallen können, also vernünftigerweise als eigenständige und unabhängige Werke zu bezeichnen sind. Zweifel hieran könnten sich bereits daraus ergeben, dass die Benutzung der Kernelschnittstelle die Einbeziehung bestimmter Header-files erfordert.[578] Sofern diese Header-files dabei die erforderliche Schöpfungshöhe erreichen – mithin urheberrechtlichen Schutz genießen – und zudem unter der GPL lizenziert sind, kommt die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL jedenfalls für kompilierte Systemmodule nicht in Betracht.[579] Die entsprechenden Systemmodule wären *„derivative works“* im Sinne der GPL und als solche mitsamt des dazugehörigen Quellcodes zu veröffentlichen.

Selbst wenn aber die Systemmodule ohne die Einbeziehung urheberrechtlich geschützter Header-files auskommen sollten, stellt sich die Frage, ob sie unabhängige und eigenständige Werkteile im Sinne von Ziff. 2 Abs. 2 S. 2 GPL sind.

Bedenken daran werden teilweise vor dem Hintergrund geäußert,[580] dass die Systemmodule – anders als andere Anwendungen – zur Laufzeit nicht im Userspace, sondern im Kernelspace

[577] Die Fähigkeit des Linux Kernels, Module dynamisch zu laden, wurde erst in Version 1.2 aufgenommen. Bis dahin war der Linux-Kernel rein monolithisch ausgestaltet, d.h. sämtliche Treiber etc. mussten fest in den Kern einkompiliert werden, was diesen nicht nur groß, sondern auch unflexibel machte. Immerhin bedurfte es zu jeder Treiberänderung einer erneuten Kompilierung des gesamten Kernels.

[578] Hier ist vor allem an die Einbindung von module.h, kernel.h und auch init.h zu denken; Zwingend ist diese allerdings nicht, da es auch möglich ist, den Kernel um eigene Header-files zu erweitern, die nicht unter der GPL lizenziert sind.

[579] Vgl. die Ausführungen S. 125 ff.

[580] Exemplarisch sei auf die zahlreichen Diskussionen um loadable Kernel Module in den Linux Kernel Mailinglisten unter http://lkml.org verwiesen, vgl. beispielsweise http://lkml.org/lkml/2006/12/13/370, sowie

ablaufen würden.[581] Die hierdurch vermittelte unmittelbare Integration der Systemmodule in den Kern soll sich insofern unmittelbar auf deren rechtliche Beurteilung durchschlagen. Maßgebliches Kriterium für die Eigenständigkeit von Systemmodulen wäre folglich der Ort ihrer Ausführung.

Hiergegen bestehen indes erhebliche Bedenken. So hätte ein solches Kriterium zur Folge, dass selbst diejenigen Kernelmodule als abgeleitete Werkteile zu beurteilen wären, die dies offensichtlich nicht sein können, weil sie beispielsweise bereits älter als der Linux-Kernel sind oder primär für den Einsatz in anderen Unix-derivaten entwickelt wurden.[582]

Aufgrund dieser Erwägungen wird in der Literatur angenommen, dass eine Eigenständigkeit der Module jedenfalls in den Fällen gegeben sei, in denen diese bereits vor dem Linux-Kern bestanden haben, oder ggf. nach entsprechender Anpassung der Schnittstellen auch mit anderen Unix-artigen Betriebssystemkernen ablauffähig sind.[583] Außerdem soll eine Eigenständigkeit auch bei denjenigen Modulen in Betracht kommen, die ausschließlich über die Kernelschnittstelle eingebunden werden.[584] Insofern wird in der Literatur letztlich auch im Hinblick auf Kernelmodule maßgeblich darauf abgestellt, ob diese tatsächlich anderweitig einzusetzen sind.

Dem ist mit der Ausnahme zuzustimmen, dass nicht die *tatsächliche* sondern nur die *theoretische* Einsetzbarkeit der jeweiligen Kernelmodule gefordert werden kann. Andererseits hinge die Werkeigenschaft von Systemmodulen maßgeblich von dem Zufall ab, ob zum Zeitpunkt der Werkerstellung bereits ein anderer Betriebssystemkern vorhanden ist, der die vom Systemmodul erforderten Schnittstellenteile aufweist und dadurch eine anderweitige Einsetzbarkeit des Systemmoduls ermöglicht. Hierauf kann es aber offensichtlich nicht

---

http://dir.gmane.org/gmane.law.gpl.violations.legal und http://dir.gmane.org/gmane.linux.kernel; a.A. ist hingegen Linus Torwalds, der aus diesem Grund zur GPL folgende Bemerkung hinzugefügt hat:
*"NOTE! This copyright does *not* cover user programs that use kernel services by **normal system calls** - this is merely considered **normal use** of the kernel and does ***not*** fall under the heading of »derived work«. Also note that the GPL below is copyrighted by the Free Software Foundation, but the instance of code that it refers to (the Linux kernel) is copyrighted by me and others who actually wrote it. Also note that the only valid version of the GPL as far as the kernel is concerned is _this_ license (ie v2), unless explicitly otherwise stated."*

[581] Um diesen Bedenken entgegenzuwirken, gab es erst jüngst Bestrebungen, die Ausführung von Systemmodulen zur Laufzeit durch die Installation zusätzlicher Software künstlich in den Userspace zu verlegen (vgl. http://www.heise.de/newsticker/meldung/77457). Hieraus wird letztlich ersichtlich, wie wenig Aussagekraft der Ort der Programmausführung im Speicher, auf die Werkeigenschaft der jeweiligen Programmkomponenten hat. Diese wäre nämlich letztlich von der Installation der Zusatzsoftware und damit von einem Umstand abhängig, der offensichtlich nicht dazu geeignet ist, etwas über die Werkeigenschaft von Systemmodulen auszusagen.

[582] Hierauf weisen zu Recht *Jaeger / Metzger*, Open Source Software, Rn. 56, hin.

[583] So insbesondere *Jaeger / Metzger*, Open Source Software, Rn. 56.

[584] Vgl. *Jaeger* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 2 Rn. 29.

ankommen. Vielmehr muss es ausreichen, dass eine anderweitige Einsetzbarkeit zumindest denkbar ist.[585]

Fraglich bleibt allerdings, ob aus dem Umstand, dass ein Kernelmodul über die Kernelschnittstelle eingebunden ist, zwangsläufig dessen anderweitige Einsetzbarkeit folgt. Zweifel daran könnten hinsichtlich solcher Module bestehen, die externe Funktionen referenzieren. Hierunter werden vorliegend[586] solche Funktionen verstanden, die nicht im eigentlichen Kern, sondern in anderen Modulen enthalten sind.[587]
Dieser Umstand wirkt sich in technischer Hinsicht zunächst einmal dadurch aus, dass diese Module nur dann in den Kernel geladen werden können, wenn zuvor sämtliche externen Referenzen aufgelöst werden konnten. [588]
Im Gegensatz zu sonstigen Anwendungen reicht es somit für einen funktionsentsprechenden Einsatz des Moduls nicht aus, ausschließlich eine passende Schnittstelle bereitzuhalten. Vielmehr ist es unerlässlich, dass neben der Schnittstelle, auch alle Module vorhanden sind, zu denen Abhängigkeiten bestehen. Aufgrund dieses Umstandes lässt sich ein entsprechendes Modul, sofern es auf externe Funktionen angewiesen ist, nicht getrennt von den anderen Modulen einsetzten, selbst wenn ein anderer Kern über die gleiche Schnittstelle verfügen würde. Ein solches Modul kann daher kein eigenständiges Werk sein.

Insofern lässt sich festhalten, dass eine Werkeigenschaft einzelner Module immer dann ausscheidet, wenn diese Funktionen anderer Module referenzieren und somit auf deren vorherige Einbindung in den Kern zwingend angewiesen sind.
Ein eigenständiges Werk kann beim Bestehen solcher Abhängigkeiten ausschließlich das „Gesamtmodul“ darstellen, in dem sämtliche abhängigen Module vereint sind. Dieses lässt sich – mangels weiterer Abhängigkeiten – durchaus auch mit anderen Betriebssystemkernen einsetzen, sofern diese über eine entsprechende Schnittstelle verfügen. Es ist daher zumindest nicht auszuschließen, dass dieses „Gesamtmodul“ auch nach der Verkehrsanschauung als eigenständiges Werk betrachtet würde.

[585] Vgl. KG Schulze KGZ, 55, 12 – Puppenfee; *Loewenheim* in: Schricker, § 8 Rn. 5.
[586] Von der Begrifflichkeit her werden sämtliche Funktionen als extern bezeichnet, die nicht direkt im Modul enthalten sind. Hierunter fallen somit auch die sämtliche Funktionen, die der Betriebssystemkern entsprechend der Schnittstellenbeschreibung bereithält.
[587] Die durch die Referenzierung externer Module entstehenden Abhängigkeiten eines Moduls lassen sich unter Linux mithilfe des Programms depmod (Bestandteil der modutils-Standard-Tool-Sammlung) analysieren.
[588] Andernfalls gibt der Kernel eine Fehlermeldung aus und verweigert das Laden des Moduls; vgl. *Mauerer*, Linux Kernelarchitektur, S. 318.

Aus lizenzrechtlicher Sicht ist allerdings zu berücksichtigen, dass sich die Frage nach der Werkeigenschaft eines solchen „Gesamtmoduls“ lediglich dann stellt, wenn unter den referenzierten Kernelmodulen keine sind, die unter der GPL lizenziert wurden. Andererseits wäre nämlich bereits die Anwendung von Ziff. 2 Abs. 2 S. 2 GPL ausgeschlossen, weil das Werk als „Gesamtmodul“ GPL-lizenzierte Teile beinhaltet und die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL somit nicht in Betracht kommt. Das „Gesamtmodul“ würde vollständig von der Lizenzierungspflicht aus Ziff. 2 Abs. 1 b GPL erfasst werden.

Liegt hingegen, entweder ein Kernelmodul vor, dass keine Funktionen anderer Module referenziert, oder aber ein „Gesamtmodul“ in dem kein GPL-lizenzierter Werkteil enthalten ist, so liegt die Anwendung von Ziff. 2 Abs. 2 S. 2 GPL nahe. Immerhin lassen sich diese Module nach einer entsprechenden Anpassung der Schnittstelle auch mit anderen Betriebssystemkernen verwenden.[589] Hieraus wird letztlich ihre Eigenständigkeit ersichtlich.
Eine Ausnahme ist hiervon wiederum dann vorzusehen, wenn die Systemmodule nicht nur über die Kernelschnittstelle, sondern auch auf andere Weise[590] in den Kern eingebunden wurden. In diesen Fällen scheidet eine anderweitige Einsetzbarkeit der Systemmodule aus, da sie für ihre Funktionsfähigkeit neben der Schnittstelle weitgehende funktionale und technische Spezifikationen beim Kern voraussetzen. Hierdurch ist es in diesen Fällen nicht denkbar, den Betriebssystemkern durch einen anderen zu ersetzen und lediglich die Schnittstellen anzupassen. Die Systemmodule sind somit zwingend auf den speziellen Betriebssystemkern angewiesen und stellen keine eigenständigen Werkteile im Sinne von Ziff. 2 Abs. 2 S. 2 GPL dar.

Zusammenfassend können Systemmodule somit nur unter folgenden Voraussetzungen eigenständige Werke darstellen:

- sie dürfen nicht auf die Einbindung von urheberrechtlich geschützten Header-Files angewiesen sein, sofern diese unter der GPL lizenziert wurden.

[589] Faktisch wird der anderweitige Einsatz von Kernelmodulen letztlich dadurch verhindert, dass die Linux-Kernelschnittstelle häufigen Änderungen unterliegt, die stets auch eine erneute Anpassung der Modulschnittstellen erfordert. Technisch wäre es hingegen durchaus möglich die gesamte Kernelschnittstelle auf einem anderen Betriebssystemkern zu verwenden, und hierdurch eine Funktionsfähigkeit sämtlicher Kernelmodule zu ermöglichen. Dies wäre in der Praxis zwar im Hinblick auf eine Portabilität der Module wünschenswert, ist aber derzeit nicht realisierbar, da gerade bei den Kernelentwicklern keine Bereitschaft dazu besteht, die Kernelschnittstelle über einen längeren Zeitraum unverändert zu lassen. Für die rechtliche Beurteilung ist dieser Umstand allerdings nicht maßgeblich.
[590] *Jaeger / Metzger*, Open Source Software, Rn. 56, nennt als Beispiel die Einbindung über Hooks oder eine nicht nur oberflächliche Nutzung der Infrastruktur des Kernels.

- Weiterhin dürfen sie ausschließlich über die Kernelschnittstelle mit dem Betriebssystemkern verbunden sein und für ihren funktionsentsprechenden Einsatz nicht die vorherige Einbindung GPL-lizenzierter Module voraussetzen.[591]

Liegen diese Voraussetzungen vor, ist es gerechtfertigt, Systemmodule als eigenständige Werke anzusehen. Dieses Ergebnis wird wiederum auch durch den in der Praxis anzutreffenden getrennten Vertrieb von Systemmodulen und Betriebssystemkern bestätigt, welcher seine Ursache nicht nur in den rechtlichen Unwägbarkeiten,[592] sondern auch darin hat, dass beispielsweise jede Hardware unterschiedliche Systemmodule benötigt, für den Betriebssystemhersteller jedoch nicht vorherzusagen ist, welchen Hardwareanforderungen das Betriebssystem im Einzelfall gerecht zu werden hat. Insofern vertreiben die Hardwarehersteller, deren Module nicht bereits in den gängigen Distributionen enthalten sind, ihre Module unabhängig von diesen.[593]

ee)Programmbibliotheken

Eine weitere Schnittstelle, die zu einer anderweitigen Einsetzbarkeit der hierüber verbundenen Komponenten führen könnte, wird von Programmbibliotheken bereitgehalten.

Programmbibliotheken sind für die Softwareentwicklung von besonderer Bedeutung. In ihnen werden üblicherweise solche Funktionen und Daten untergebracht, die von mehreren Programmen oder Programmteilen benötigt werden.

Sofern eine Komponente auf eine Programmbibliothek zugreift, erfolgt dieser Zugriff stets über eine Schnittstelle, die oftmals auch dokumentiert ist. Gerade im Open Source Bereich ist allerdings, aufgrund der bestehenden Möglichkeit zur Quellcodeanalyse, besondere Rücksicht darauf zu nehmen, ob die verwendete Schnittstelle lediglich für einen programminternen Gebrauch, oder auch für eine anderweitige Einsetzbarkeit konzipiert wurde.

[591] Hierfür ist es unerheblich, ob diese Module bereits statisch in den Kernel integriert wurden oder erst nachträglich hinzugeladen werden sollen.

[592] Gerade in jüngster Zeit ist der Trend zu erkennen, dass die großen Distributoren (z.B. Suse ab Version 10.1) dazu übergehen, die vom Kunden benötigten Module zweistufig zu vertreiben. Auf der ersten Stufe erfolgt der Vertrieb der Distribution zusammen mit Kernelmodulen, die ausschließlich unter der GPL oder einer hierzu kompatiblen Lizenz stehen. Auf der zweiten Stufe werden dem Kunden sodann möglicherweise benötigte proprietäre Treiber auf einem Server zum Download bereitgestellt, vgl. http://www.golem.de/0602/43285.html.

[593] Vgl. nur den Vertrieb der Treiber für die Grafikkarten von nVidia (http://www.nvidia.de/page/drivers.html) und AMD (http://ati.amd.com/support/driver-de.html).

Eine programmübergreifende Einsetzbarkeit der Bibliothek wurde dabei offensichtlich beabsichtigt, sofern standardisierte Schnittstellen verwendet wurden.[594] Andererseits ist die tatsächliche Standardisierung einer Schnittstelle aber keine Voraussetzung für deren anderweitige Einsetzbarkeit. Es reicht auch hier die theoretische Einsetzbarkeit, da die Werkeigenschaft einer Bibliothek ansonsten wiederum von dem Zufall abhängen würde, ob bereits genügend andere Bibliotheken die gleiche Schnittstelle verwenden.
Bis hierhin lässt sich somit bereits festhalten, dass diejenigen Programmbibliotheken, die ihre Funktionalität unabhängig von einzelnen Anwendungen über eine dokumentierte Schnittstelle bereithalten, anderweitig einsetzbar sind und somit eigenständige Werke darstellen.[595] Die Ausnahmeregelung in Ziff. 2 Abs. 2 S. 2 GPL käme somit grundsätzlich in Betracht, auch wenn sie in der Praxis kaum einmal an Bedeutung gewinnen dürfte, da in den streitigen Fällen zumeist die Programmbibliotheken, nicht aber die hiermit verbundenen Komponenten, unter der GPL lizenziert sind.[596]

Fraglich bleibt aber, ob auch die zugreifenden Programmkomponenten eigenständige Werkteile im Sinne von Ziff. 2 Abs. 2 S. 2 GPL sind. Zweifel könnten wiederum deshalb bestehen, weil die Benutzung von Bibliotheken – unabhängig davon, ob diese letztlich statisch oder dynamisch eingebunden werden – die Einbeziehung bestimmter Header-files voraussetzt. Soweit diese urheberrechtlich schutzfähig[597] und zudem unter der GPL lizenziert sind, wären jedenfalls die kompilierten Programmkomponenten, aufgrund der darin enthaltenen GPL-lizenzierten Codeteile nicht dazu geeignet einen eigenständigen Werkteil im Sinne von Ziff. 2 Abs. 2 S. 2 GPL darzustellen.[598]

Sofern hingegen den eingebundenen Header-files die ausreichende Schöpfungshöhe fehlt, stellt sich weiterhin die Frage, ob die Verbindung zwischen einer Programmkomponente und einer Programmbibliothek zwangsläufig dazu führt, dass diese als unselbstständiges Werk zu betrachten ist.

---

[594] Verbreitete Standards sind IEEE POSIX, X/Open XPG4, C89/C99 sowie Linux Standard Base; Einzelheiten hierzu unter *Herold*, Linux-Unix-Systemprogrammierung, S. 36 ff.
[595] Ähnlich in der Einschätzung *Wuermeling / Deike*, CR 2003, 87, 90; und wohl auch *Spindler*, Rechtsfragen bei Open Source, C. Rn. 120.
[596] Bekannte Bibliotheken, die unter der GPL lizenziert wurden, sind bspw. die PING Utility Library, vgl. http://www.gnu.org./directory/pingutils.html, die GNU cgicc Library http://www.gnu.org./software/cgicc/cgdi-doc.html und die GNet (network library), vgl. http://www.gnu.org./gnulist/production/gnet.html.
[597] Hierfür reicht es allerdings nicht aus, dass die betreffenden Header-files über umfangreiche Schnittstellendefinitionen verfügen, da diesen gem. § 69 a Abs. 2 UrhG kein urheberrechtlicher Schutz zukommt.
[598] Einzelheiten zur Einbindung von Header-files siehe oben S. 125 ff.

Ausgehend vom Kriterium der anderweitigen Einsetzbarkeit der Komponente kann jedenfalls festgehalten werden, dass die Bindung über eine dokumentierte Schnittstelle eine anderweitige Einsetzbarkeit nicht ausschließt. Gerade das Gegenteil wird erreicht und ist auch im Bereich der Systembibliotheken ersichtlich, bei denen sich in der Vergangenheit zahlreiche Standardisierungen durchgesetzt haben.[599] Eine Programmkomponente, die beispielsweise über die Linux Standard Base-Schnittstelle auf die Programmbibliothek zugreift, kann unverändert mit anderen Bibliotheken eingesetzt werden, die diese Schnittstelle ebenfalls unterstützen. Ihre anderweitige Einsetzbarkeit ist somit offensichtlich gegeben, weshalb keine Bedenken daran bestehen, dass diese Komponente ein eigenständiges Werk im Sinne von Ziff. 2 Abs. 2 S. 2 GPL ist.[600]

Sofern in der Literatur weiterhin danach unterschieden wird, ob die Verbindung zwischen Programmkomponente und Bibliothek statisch oder dynamisch erfolgt,[601] betrifft dies nicht die Frage nach der Werkeigenschaft, sondern die nach der Vertriebsform, die zwar ebenfalls in Ziff. 2 Abs. 2 S. 2 GPL konkretisiert wird („*...as separate works...*"), aber erst Gegenstand der nachfolgenden Untersuchung ist.[602]

Insofern kann auch im Hinblick auf Programmbibliotheken festgehalten werden, dass sowohl die Bibliothek als auch die zugreifende Komponente eigenständige Werke im Sinne von Ziff. 2 Abs. 2 S. 2 GPL sein können. Voraussetzung ist einzig, dass die Bibliothek für eine anderweitige Einsetzbarkeit ausgestaltet ist und zu diesem Zweck eine abstrakte und dokumentierte Schnittstelle bereithält.

Zusammenfassung

Ausgangspunkt der Untersuchung war die Frage, welche Werkteile eines modularen Werks von der in der GPL enthaltenen Lizenzierungspflicht umfasst werden. Eine eindeutige Antwort auf diese Frage lässt sich, unabhängig von der gewählten Vertriebsform, lediglich für diejenigen Fallgruppen geben, in denen GPL-lizenzierte Teile innerhalb des modularen Werks verwendet wurden und zugleich feststeht, dass unter den vorhandenen Komponenten kein

[599] Verbreitete Standards sind IEEE POSIX, X/Open XPG4, C89/C99 sowie Linux Standard Base; Einzelheiten hierzu unter *Herold*, Linux-Unix-Systemprogrammierung, S. 36 ff.
[600] In der Praxis führt die Bindung zu Systembibliotheken zumeist zu keinen Problemen, da diese häufig unter Lizenzen stehen, die eine solche Bindung ausdrücklich zulassen. Für Linux ist hierbei insbesondere an die glibc zu denken, die unter der LGPL lizenziert wurde.
[601] Vgl. nur *Jaeger / Metzger*, Open Source Software, Rn. 57 ff; *Spindler*, Rechtsfragen bei Open Source, 119 ff; *Andréewitch*, MR 2005, 240, 242.
[602] Hierzu sogleich ab S. 165.

unabhängiger und eigenständiger Teil im Sinne von Ziff. 2 Abs. 2 S. 2 GPL zu identifizieren ist.
Hieran knüpfte sich die Frage an, wonach sich beim Vorliegen mehrerer Komponenten bestimmen lässt, ob diese unabhängig und eigenständig oder lediglich unselbstständiger Teil einer größeren Komponente sind?

Erste Voraussetzung, die ein unabhängiger und eigenständiger Werkteil erfüllen muss, ist dessen Trennbarkeit von den sonstigen Werkteilen. Diese ist zwar nicht allein dazu geeignet eine Eigenständigkeit zu begründen, ihr Fehlen schließt sie allerdings zwangsläufig aus.
Weiterhin darf der Werkteil weder GPL-lizenzierte Programme oder Teile davon beinhalten, um nicht bereits aus diesem Grund von der Lizenzierungspflicht erfasst zu sein.
Die Untersuchung von Ziff. 2 Abs. 2 und 3 GPL hat weiterhin ergeben, dass sich für die Auslegung von Ziff. 2 Abs. 2 S. 2 GPL solche Kriterien heranziehen lassen, die bereits für die Abgrenzung von eigenständigen zu einheitlichen Werken bei Werkverbindungen im Sinne von § 9 UrhG entwickelt wurden.
Dementsprechend war das Vorliegen eines eigenständigen Werks an die weitere Voraussetzung zu knüpfen, dass die betreffende Komponente bei ihrer Trennung von den aus den sonstigen Werkteilen nicht unvollständig oder ergänzungsbedürftig wird. Dabei hat sich gezeigt, dass im Gegensatz zu den klassischen Werkarten, die Unvollständigkeit und Ergänzungsbedürftigkeit des Werks nicht anhand der bislang favorisierten Kriterien – insbesondere nicht anhand der Möglichkeit einer eigenständigen und gesonderten Verwertung – bestimmen lässt.
Weiterhin hat sich herausgestellt, dass auch das Kriterium der gesonderten Verwertbarkeit der Werkteile im Kontext mit modularer Software insofern einer Einschränkung bedarf, als es jedenfalls nicht mit deren gesonderter Nutzbarkeit gleichgesetzt werden kann, da diese aufgrund der funktionalen Abhängigkeiten der jeweiligen Programmkomponenten von einem zugrunde liegenden Betriebssystem, letztlich für keine Anwendung gegeben wäre, obwohl es der allgemeinen Ansicht entspricht, dass nicht jede funktionale Abhängigkeit zugleich die Unselbstständigkeit einer Programmkomponente zur Folge hat.
Es stellte sich daher die Frage, ob die bestehenden funktionalen Abhängigkeiten einer Programmkomponente danach qualifiziert werden können, ob sie für die Beurteilung der Unvollständigkeit oder Ergänzungsbedürftigkeit relevant, oder lediglich unbeachtliche Folge der Systemarchitektur sind.

Die Untersuchung hat gezeigt, dass die technische Form der Kommunikation zwischen Programmkomponenten aufgrund deren beliebigen Austauschbarkeit nicht dazu geeignet ist, hieran zwingende Schlussfolgerungen zu knüpfen.
Entscheidendes Kriterium für die Qualifikation der funktionalen Abhängigkeiten zwischen Programmkomponenten ist vielmehr deren Variabilität. Ist es zumindest theoretisch möglich die betreffenden Programmkomponenten nach deren Trennung wieder anderweitig zu integrieren, wird hieraus ersichtlich, dass die betreffende Komponente im Hinblick auf die bestehenden funktionalen Bindungen variabel ist. Für ihre eigene Funktionalität ist sie nicht von einzelnen anderen Komponenten abhängig, weshalb sie die Voraussetzungen eines eigenständigen Werks aufweist.
Aus technischer Sicht ist es für die Erfüllung dieser Voraussetzungen zwingend erforderlich, dass die Komponente auf der einen Seite eine eigene Funktionalität aufweist und auf der anderen Seite sämtliche Schnittstelleninformationen bereithält, die dazu erforderlich sind, beliebigen anderen Komponenten den Zugriff auf diese spezifischen Funktionalitäten zu ermöglichen.

Diese Voraussetzungen werden in der Praxis regelmäßig zwischen Plugins und Anwendungen aber auch zwischen Anwendungen und einer Middleware erfüllt.
Weiterhin werden eigenständige Werke auch dann vorhanden sein, wenn funktionale Abhängigkeiten zwischen Programmkomponenten bestehen, die über eine Benutzerschnittstelle verbunden sind.

Etwas differenzierter waren die funktionalen Abhängigkeiten zu betrachten, die zwischen dem Betriebssystemkern und den damit verbundenen Systemmodulen bestehen. Entgegen der weitläufigen Ansicht unter den Programmierern des Linux Kerns verbietet sich auch bei Systemmodulen eine pauschale Betrachtung. Sofern diese nämlich weder die Einbindung von GPL-lizenzierten Header-Files, noch die Einbindung GPL-lizenzierter Module voraussetzen und lediglich über die Kernelschnittstelle mit dem Betriebssystemkern verbunden sind, stellen auch diese eigenständige Werke dar, die von der Lizenzierungspflicht nicht zwangsläufig erfasst werden.

Letztlich hat die Untersuchung weiterhin gezeigt, dass auch im Hinblick auf Programmbibliotheken festgehalten werden kann, dass sowohl die Bibliothek als auch die zugreifende Komponente eigenständige Werke im Sinne von Ziff. 2 Abs. 2 S. 2 GPL sein

können. Voraussetzung ist einzig, dass die Bibliothek für eine anderweitige Einsetzbarkeit ausgestaltet ist und zu diesem Zweck eine abstrakte und dokumentierte Schnittstelle bereithält. Die bislang in der Literatur zumeist vorgenommene Unterscheidung danach, ob die Verbindung zwischen Programmkomponente und Bibliothek statisch oder dynamisch erfolgt, tangiert hingegen nicht die Frage nach der Werkeigenschaft, sondern bezieht sich ausschließlich auf die Vertriebsform, auf die im Folgenden einzugehen ist.

3.Stufe 3 – Vertriebsform der Programme

Nachdem festgestellt wurde, in welchen Fällen ein Einheitliches und in welchen mehrere eigenständige und unabhängige Programme in funktionaler Hinsicht vorliegen, bleibt für die Frage nach der Reichweite der Lizenzierungspflicht zu klären, unter welchen Voraussetzungen sich diese, auch auf solche eigenständigen Werke erstreckt, die nachweislich weder GPL-lizenzierten Code beinhalten noch von diesem abgeleitet sind.
In diesem Zusammenhang kommt es maßgeblich auf die Regelung in Ziff. 2 Abs. 2 S. 3 GPL an, wonach sich die Reichweite der Lizenzierungspflicht auf sämtliche Werkteile erstreckt, sofern sie als einheitliches Ganzes weitergegeben werden (*„…as part of a whole…“*). [603] Werden sie hingegen als eigenständige Teile vertrieben (*„… as separate works…“*), so kommt die Ausnahme von Ziff. 2 Abs. 2 S. 2 GPL und – sofern der Vertrieb auf einem Speichermedium erfolgt – ein Fall von Ziff. 2 Abs. 4 GPL in Betracht, so dass sich die Lizenzierungspflicht einzig auf die veränderten, nicht aber auf die eigenständigen Werke bezieht, welche daneben bestehen. Für die Beurteilung der Lizenzierungspflicht stellt die GPL somit in Ziff. 2 Abs. 2 nicht auf die funktionale Verbindung der Werkteile untereinander, sondern darauf ab, in welcher Form diese vertrieben werden.

Unter welchen Voraussetzungen ist aber nun beim Vertrieb eigenständiger und unabhängiger Komponenten davon auszugehen, dass diese als Teile eines Ganzen vertrieben werden?
Der GPL lässt sich lediglich entnehmen, dass es hierfür unerheblich ist, ob die Werke auf einem einheitlichen Vertriebsmedium vorhanden sind. Immerhin sieht sie in Ziff. 2 Abs. 4 ausdrücklich vor, dass das bloße Zusammenfügen mehrerer Werke auf einem Speichermedium keinen Einfluss auf die Reichweite der Lizenzierungspflicht hat.
Im Übrigen enthält die GPL zu dieser Frage jedoch keine weiteren Erläuterungen. Nicht zuletzt aus diesem Grund wird die Regelung von Ziff. 2 Abs. 2 GPL vielfach für widersprüchlich und auslegungsbedürftig gehalten.[604] So geht aus ihr insbesondere nicht hervor, wie ein Softwarebestandteil auf der einen Seite ein eigenständiges Werk im Sinne von Ziff. 2 Abs. 2 S. 2 sein kann, das weder GPL-lizenzierten Code beinhaltet noch von solchem abgeleitet ist, auf der anderen Seite aber zugleich gem. Ziff. 2 Abs. 2 S. 3 dem Werk als Ganzes zuzurechnen ist, das seinerseits gem. Ziff. 0 GPL wiederum einem *„derivative work“*

[603] Ziff. 2 Abs. 2 S. 3 GPL: *„… 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, …“*
[604] *Jaeger / Metzger*, Open Source Software, Rn. 50.

gleichgesetzt wird. In der Literatur wird die Regelung von Ziff. 2 Abs. 2 S. 3 GPL deshalb zu Recht anhand des damit verfolgten Zwecks ausgelegt, wonach mit ihr vermieden werden soll, dass GPL-Software mit selbstständigen Softwaremodulen in einer Form verbunden wird, die es dem Nutzer unmöglich macht, die Bestandteile ohne weiteres als eigenständige Teile zu erkennen und zu nutzen.[605]

a.Vertrieb in monolithischer Form

Nimmt man dies zum Ausgangspunkt, so wird ersichtlich, dass sich eine Lizenzierungspflicht immer dann auf sämtliche Programmteile erstreckt, wenn diese in einer Form weitergegeben werden, in der das Werk aus einer einheitlichen und nicht weiter untergliederbaren Datei besteht, unabhängig davon, ob dies eine Quelltext-, eine Objektcode- oder aber eine ausführbare Datei ist.[606] Sofern eine solche Datei überlassen wird, können zwar in funktionaler Hinsicht durchaus mehrere unabhängige und eigenständige Teilbereiche vorliegen, allerdings lassen sich diese aufgrund ihrer monolithischen Vertriebsform vom Nutzer weder als eigenständige Werkteile erkennen noch getrennt voneinander nutzen. Insofern lässt sich festhalten, dass ein Vertrieb als *„separate works“* in solchen Fällen ausgeschlossen ist, in denen an den Nutzer eine einheitliche und nicht weiter untergliederungsfähige Datei weitergegeben wird.

Auch für eine Anwendbarkeit von Ziff. 2 Abs. 4 GPL ist in diesen Fällen kein Raum, da die monolithische Programmverbindung jedenfalls kein einfaches Zusammenlegen verschiedener Werke auf einem Speichermedium darstellt, wie es von Ziff. 2 Abs. 4 GPL vorgesehen ist. Auch hierfür wäre zumindest das Vorliegen verschiedener Dateien zum Vertriebszeitpunkt erforderlich.

Aus diesem Grund erstreckt sich beispielsweise die in der GPL enthaltene Lizenzierungspflicht, auf das Werk als Ganzes, sofern eine urheberrechtlich geschützte Programmbibliothek statisch in eine ausführbare Datei eingebunden wurde.[607] Aber auch bei

[605] *Jaeger / Metzger*, Open Source Software, Rn. 52.
[606] Im Zusammenhang mit dem Vertrieb einer einheitlichen Exe-Datei so auch *Jaeger / Metzger*, Open Source Software, Rn. 53; und wohl auch *Lejeune*, ITRB 2003, 10, 10 f, der hierin zumindest ein aussagekräftiges Kriterium sehen will.
[607] Ebenso *Jaeger* in: Ifross, Ziff. 2 Rn. 43 ff; *Andréewitch*, MR 2005, 240, 242; *Determann*, GRUR Int. 2006, 645, 648.

der dynamischen Einbindung von GPL-lizenzierten Programmbibliotheken kann eine Lizenzierungspflicht für die Anwendung entstehen. Der Umstand, der in diesen Fällen zu einer Lizenzierungspflicht führen kann, liegt dabei jedoch nicht in der eigentlichen Verknüpfung zur Programmbibliothek, sondern in der hierfür erforderlichen Einbindung von Header-files, die unter Umständen urheberrechtlichen Schutz genießen[608] und nach der Kompilierung zusammen mit der Anwendung in einer monolithischen Form vorliegen.

Insgesamt lässt sich daher festhalten, dass eine Lizenzierungspflicht immer dann in Betracht kommt, wenn dem Nutzer eine Anwendung in monolithischer Form überlassen wird. In diesen Fällen ist das Werk, unabhängig von seinem funktionalen Aufbau, als Ganzes unter der GPL zu lizenzieren, sofern hierin urheberrechtlich geschützte Werkteile enthalten sind, die unter der GPL lizenziert wurden.

b.Vertrieb in modularer Form

Schwieriger ist die Beurteilung von Fallgestaltungen, in denen die Programmdateien in modularer Form weitergegeben werden. Probleme bereitet in diesem Zusammenhang vor allem die Regelung in Ziff. 2 Abs. 2 S. 3 GPL, wonach sich die Reichweite der Lizenzierungspflicht auf sämtliche Werkteile erstreckt, sofern diese als einheitliches Ganzes weitergegeben werden. Hieran knüpft sich zwangsläufig die Frage an, unter welchen Voraussetzungen der Vertrieb von mehreren eigenständigen und unabhängigen Werken in einer Art und Weise ausgestaltet ist, dass diese nicht mehr als solche zu erkennen und zu nutzen sind. Die GPL „lehrt“ uns in Ziff. 2 Abs. 4, dass es hierfür nicht darauf ankommt, ob die Werkteile auf einem gemeinsamen Speichermedium weitergegeben werden.
Es bleibt allerdings die Frage, ob mehrere eigenständige und unabhängige Werkteile auch dann als einheitliches Ganzes vertrieben werden können, wenn sie – im Gegensatz zum monolithischen Vertrieb – nicht aus einer einheitlichen Datei bestehen und möglicherweise zudem auch noch auf unterschiedlichen Vertriebswegen zum Nutzer gelangen. Unter diesen Voraussetzungen könnten Zweifel daran bestehen, dass überhaupt Fallgestaltungen denkbar sind, in denen trotz solcher Vertriebsmodalitäten die einzelnen Werkteile nicht als solche zu erkennen oder zu nutzen sind.

[608] Vgl. bereits S. 125 ff.

Allerdings ist zu berücksichtigen, dass das Internet den Unternehmen zahlreiche neue Vertriebswege eröffnet hat, die sich gerade in diesem Zusammenhang auswirken können. Hierbei ist zunächst an solche Fälle zu denken, in denen der Nutzer lediglich einzelne Werkteile auf einem Datenträger überreicht bekommt und die Restlichen erst im Rahmen der Installation über das Internet hinzugeladen werden. Dieses Hinzuladen weiterer Programmkomponenten kann dabei auf unterschiedliche Weise erfolgen.
So ist es denkbar, dass der Nutzer darauf hingewiesen wird, dass das Programm weitere Programmkomponenten benötigt, um die gewünschten Funktionalitäten zu erhalten und er sich diese eigenständig besorgen muss. Andererseits ist es aber auch möglich, dass das Programm die benötigten Programmkomponenten beispielsweise im Installations- oder auch Updatevorgang eigenständig von firmeneigenen Servern hinzulädt, ohne dass dies für den Nutzer erkennbar wird. Während im ersten Fall die eigenständigen Werkteile für den Nutzer als solche erkennbar bleiben, würde es hieran im zweiten Fall, trotz der körperlichen Trennung der Werkteile zum Vertriebszeitpunkt, fehlen.
Hieraus wird ersichtlich, dass es für die Beurteilung, ob mehrere eigenständige und unabhängige Werkteile als einheitliches Ganzes im Sinne von Ziff. 2 Abs. 2 S. 3 GPL weitergegeben werden maßgeblich auf den Installationsprozess ankommen kann.
Andererseits lässt sich aber auch festhalten, dass mehrere eigenständige Werke jedenfalls dann nicht als einheitliches Ganzes weitergegeben werden, wenn dem Nutzer alle Werkteile als solche in getrennter Form auf einem Datenträger überlassen werden. Das Vorhandensein eines Installationsskripts, welches die vorhandenen eigenständigen Werkteile während der Installation zu einem einheitlichen Programm zusammenführt, ändert nämlich nichts daran, dass sich diese auch unabhängig hiervon nutzen lassen, sofern sie dem Nutzer in getrennter Form überlassen werden.

Eine andere Beurteilung ist hingegen gerechtfertigt, wenn die eigenständigen Werke in einer Form vorliegen, in der sie ausschließlich im Rahmen der Programmausführung, nicht aber getrennt hiervon eigenständig zu nutzen sind, weil sie beispielsweise in einem unbekannten Komprimierungsformat oder auch verschlüsselt übergeben werden und in dieser Form ausschließlich nach der Installation, zusammen mit der eigentlichen Anwendung, nicht aber getrennt hiervon, vom Benutzer zu verwenden sind. In diesen Fällen fehlt es an der Nutzbarkeit der einzelnen Teile, so dass es dem Zweck von Ziff. 2 Abs. 2 S. 3 GPL entspricht, diese ein einheitliches Ganzes anzusehen.

Vergleichbar sind solche Fallgestaltungen, in denen die eigenständigen Werkteile zwar getrennt voneinander überlassen werden, aber auf einem Speichermedium sind, das seinerseits ausschließlich durch eine spezifische Anwendung nicht aber durch den Nutzer ausgelesen werden kann. In diesen Fällen fehlt es offensichtlich sowohl an der Erkennungs- als auch an der Nutzungsmöglichkeit der eigenständigen Werkteile, so dass aufgrund der Vertriebsmodalitäten von einem einheitlichen Ganzen im Sinne von Ziff. 2 Abs. 2 S. 3 GPL und mithin von einem *„derivative work“* auszugehen ist.

Diese Fälle kommen insbesondere im Bereich der Embedded Devices in Betracht, wenn der Nutzer die Software zusammen mit Hardware bereits in einem installierten Zustand erhält. Hier ist somit besondere Rücksicht darauf zu nehmen, dass die einzelnen Werkteile zum Übergabezeitpunkt in einer Form vorliegen, in der sie ohne weiteres durch den Nutzer aus den bestehenden funktionalen Abhängigkeiten herausgelöst und auch genutzt werden können.

Dabei wird eine Nutzbarkeit der Werkteile immer dann ausgeschlossen sein, wenn die Werkteile auf einer Hardware überlassen werden, die durch den Nutzer überhaupt nicht oder nur eingeschränkt ausgelesen werden kann, weil sie beispielsweise in einem Gerät enthalten ist, dass keine Benutzerschnittstelle bereithält oder diese verschlüsselt. In diesen Fällen hat der Nutzer unabhängig von der konkreten Ausgestaltung der funktionalen Abhängigkeiten keine Möglichkeit auf die überlassenen Werkteile zuzugreifen, so dass es sowohl an der Erkennungs- als auch an der Nutzungsmöglichkeit der jeweiligen Werkteile fehlt. Ausgehend vom Zweck der Regelung in Ziff. 2 Abs. 2 GPL ist daher vom Vorliegen eines einheitlichen Ganzen auszugehen und die Lizenzierungspflicht folglich auf alle Werkteile zu erstrecken, die zusammen mit den GPL-lizenzierten Codeteilen auf einem solchen Speichermedium überlassen werden.

G. Wesentliches Ergebnis

Ausgangspunkt der vorliegenden Arbeit war die Frage, unter welchen Voraussetzungen sich GPL-lizenzierte Programme im Umfeld proprietärer Entwicklungen einsetzen lassen, ohne dadurch von der Pflicht betroffen zu werden die eigenen Entwicklungen ebenfalls unter den Bestimmungen der GPL lizenzieren zu müssen. Die Beantwortung dieser Frage, hängt maßgeblich von der Reichweite des in Ziff. 2 GPL geregelten Copyleft-Effekts ab, der in der GPL anhand auslegungsbedürftiger Begriffe definiert und in dessen unbestimmter Reichweite oftmals ein viraler Effekt befürchtet wird.
Die bestehenden Unsicherheiten im Zusammenhang mit der Reichweite des viralen Effekts waren dabei vor allem auf das Fehlen von eindeutigen Abgrenzungskriterien zurückzuführen, welche ihrerseits wiederum zumindest teilweise durch fehlende Entscheidungen der Rechtsprechung bedingt wurden. Um diese Unsicherheiten zu mindern, wurde mit der vorliegenden Arbeit das Ziel verfolgt, den technischen Rahmen aufzuzeigen, in dem die Nutzung von GPL-lizenzierten Programmen für Unternehmen rechtlich möglich ist, ohne dadurch von der Pflicht betroffen zu werden die eigenen Entwicklungen ebenfalls unter den Bestimmungen der GPL lizenzieren zu müssen. Es ging mithin um den Umfang der durch die GPL vermittelten Pflichten und insbesondere um die Reichweite des viralen Effekts.

Um eine Eingrenzung der Fallgruppen vorzunehmen, in denen sich der virale Effekt überhaupt auswirken kann, wurde im Rahmen der Untersuchung zunächst der konkludente Abschluss der GPL eingehender beleuchtet.

Dabei hat sich herausgestellt, dass die GPL zumeist erst durch eine konkludente Willensäußerung wirksam abgeschlossen wird. Ihr Wirkungsbereich ist somit nur dann eröffnet, wenn der Nutzer zustimmungsbedürftige Handlungen im Sinne von § 69 c UrhG vornimmt, also Open Source Programme verändert oder verbreitet.

Im Rahmen der Untersuchung hat sich jedoch weiterhin gezeigt, dass auch der Abschluss der GPL nicht zwangsläufig die Pflicht zur Offenlegung des Quellcodes nach sich zieht. Diese greift gem. Ziff. 2 Abs. 1 GPL vielmehr erst dann ein, wenn zustimmungsbedürftige Veränderungen vervielfältigt oder vertrieben werden sollen.

Eine legitime Möglichkeit dem viralen Effekt der GPL zu entgehen liegt somit darin, den Vertrieb der eigenen Entwicklungen vor die eigentliche Bearbeitung zu setzten. Anstatt die, für den Einsatz der eigenen Produkte erforderlichen, Veränderungen an GPL-lizenzierten

Programmen bereits firmenintern durchzuführen, könnten sich die Unternehmen darauf beschränken, ausschließlich die eigenen Werkteile weiterzugeben und den Kunden im Übrigen darauf zu verweisen sich die GPL-lizenzierten Programme aus öffentlich zugänglichen Quellen zu besorgen. Die Koppelung von GPL-Code und eigenem Code, welche lizenzrechtlich relevant sein könnte, ließe sich hierdurch vermeiden, da der Vertrieb der eigenen Werkteile keiner Beschränkung unterliegt und hierfür auch keine Zustimmung des Urhebers gem. § 69 c Nr. 2 UrhG erforderlich ist.

Damit kann bereits festgehalten werden, dass die Unternehmen durch die Strukturierung ihres Vertriebs die Geltung der GPL maßgeblich beeinflussen können. Besteht die Möglichkeit GPL-lizenzierte Programme getrennt von den eigenen Entwicklungen zu vertreiben und ist für die Entwicklung der eigenen Programme keine zustimmungsbedürftige Veränderung an GPL-lizenzierten Programmen erforderlich, so kann der Abschluss der GPL und mithin auch ihre Wirkung vermieden werden. Ein solches Vorgehen widerspricht zwar offensichtlich den mit dem Open Source Modell verfolgten Zielen, lässt sich aber wohl anhand der derzeitigen Regelungen nicht vermeiden. Immerhin fehlt es zum Zeitpunkt des Vertriebs der Eigenentwicklung noch an einer zustimmungsbedürftigen Handlung der Unternehmen, so dass die GPL keine Bindungswirkung entfalten kann.

Vertreiben Unternehmen hingegen, neben den eigenen Entwicklungen, auch GPL-lizenzierte Programmkopien, so wird regelmäßig von einem konkludenten Abschluss der GPL auszugehen sein. Eine Ausnahme besteht lediglich in den Fällen, in denen sich die Unternehmen auf den Vertrieb von Programmkopien beschränken, die sie zuvor selbst in körperlicher Form erworben haben und an denen sich somit das Verbreitungsrecht gem. §§ 69 c Nr. 3 S. 2, 17 Abs. 2 UrhG erschöpft hat.

Vertreiben Unternehmen demgegenüber neben eigenen Entwicklungen auch GPL-lizenzierte Programme, an denen sich das Verbreitungsrecht noch nicht erschöpft hat, so hängt die Pflicht zur Offenlegung des Quellcodes maßgeblich von der Reichweite des Copyleft-Effekts ab.
Die Pflicht zur Offenlegung des Quellcodes erstreckt auf die eigenen Entwicklungen, wenn der Anwendungsbereich der Ziff. 2 GPL eröffnet, es also zu Veränderungen der ursprünglichen Programmkopie oder Teilen davon gekommen ist und zugleich feststeht, dass unter den weitergegebenen Dateien, keine Werkteile vorhanden sind, die sich gegenüber der ursprünglichen Programmkopie als funktional unabhängig und eigenständig darstellen.

Lizenzrechtlich relevante Veränderungen im Sinne von Ziff. 2 GPL werden dabei regelmäßig vorliegen, wenn:

- urheberrechtlich schutzwürdige Quelltextdateien einer GPL-lizenzierten Programmkopie direkt verändert werden,

- eine urheberrechtlich geschützte Quelltextdatei oder Teile davon in eigene Entwicklungen übernommen werden, indem der GPL-lizenzierte Quelltext insgesamt oder auch in Auszügen kopiert und in andere Quelltextdateien eingefügt wird. Dies gilt unabhängig davon, ob diese Übernahme manuell oder automatisiert, etwa durch den Präprozessor im Rahmen des Kompilierungsvorgangs, erfolgt,

- ein Diff vertrieben wird, durch das einem GPL-lizenzierten Programm nicht nur Funktionalität hinzugefügt, sondern auch Teile desselben im Rahmen der Überarbeitung gelöscht werden sollen,

- urheberrechtlich geschützte Funktionen statisch Eingebunden werden,

- GPL-lizenzierte Quelltextdateien kompiliert werden,

- aus mehreren Objektdateien eine ausführbare Datei erstellt wird und der Linker hierfür auch Objektdateien einbindet, die unter der GPL lizenziert sind.

Die vorliegende Untersuchung hat weiterhin gezeigt, dass allein aus dem Vorliegen einer lizenzrechtlich relevanten Veränderung im Sinne von Ziff. 2 GPL noch nicht zwangsläufig auf die Pflicht zur Offenlegung des Quelltextes für sämtliche Komponenten geschlossen werden kann, die zusammen vertrieben werden sollen. Vielmehr setzt die Pflicht zur Offenlegung des Quelltextes auch voraus, dass unter den weitergegebenen Dateien, keine Werkteile vorhanden sind, die sich gegenüber der ursprünglichen Programmkopie als funktional unabhängig und eigenständig darstellen.
Für die Beurteilung, ob Werkteile gegenüber der ursprünglichen Programmkopie funktional unabhängig und eigenständig sind, hat sich dabei herausgestellt, dass hierfür maßgeblich auf die anderweitige Einsetzbarkeit der Komponente abzustellen ist.

Aus technischer Sicht setzt die anderweitige Einsetzbarkeit einer Komponente voraus, dass sie sämtliche Schnittstelleninformationen bereithält, die dazu erforderlich sind, um beliebigen anderen Komponenten den Zugriff auf ihre spezifischen Funktionalitäten zu ermöglichen. Liegen diese technischen Voraussetzungen vor, lässt sich die entsprechende Komponente zwar unter Umständen nicht isoliert, aber doch ohne die konkreten anderen Teile verwenden. Für die Entfaltung ihrer spezifischen Funktionalität ist sie folglich auf keine der anderen Komponenten zwingend angewiesen, weshalb es gerechtfertigt ist die entsprechende Komponente als eigenständiges Werk im Sinne von Ziff. 2. Abs. 2 GPL zu bewerten.

Unter Zugrundelegung dieser Voraussetzungen besteht eine funktionale Unabhängigkeit und Eigenständigkeit von Programmkomponenten vor allem:

- wenn zwischen Plugins und der entsprechenden Anwendung zu unterscheiden ist,

- wenn Anwendungen von der Middleware abzugrenzen sind,

- zwischen den Komponenten verschiedener Anwendungen, sofern diese über eine definierte Benutzerschnittstelle aufeinander zugreifen,

- bei Systemmodulen, die zusammen mit dem Kernel überlassen werden, sofern
  - sie nicht auf die Einbindung von GPL-lizenzierten Header-Files angewiesen sind und
  - sie ausschließlich über die Kernelschnittstelle mit dem Betriebssystemkern verbunden sind und für ihren funktionsentsprechenden Einsatz nicht die vorherige Einbindung GPL-lizenzierter Module voraussetzen,

- zwischen Bibliotheks- und Anwendungskomponenten, sofern die Bibliothek für eine anderweitige Einsetzbarkeit ausgestaltet ist und zu diesem Zweck eine abstrakte und dokumentierte Schnittstelle bereithält.

Liegen danach mehrere eigenständige Werke vor, kommt eine Pflicht zur Offenlegung des Quellcodes nur noch dann in Betracht, wenn diese als einheitliches Ganzes vertrieben werden. Hiervon ist auszugehen, wenn dem Nutzer eine Anwendung in monolithischer Form überlassen wird und sich daher weder die eigenständigen Werkteile erkennen noch getrennt voneinander nutzen lassen. Werden hingegen mehrere eigenständige Werke in modularer

Form überlassen, kann von einem einheitlichen Ganzen nur dann ausgegangen werden, wenn die GPL-lizenzierten Programmkomponenten vom Nutzer nicht ohne weiteres als eigenständige Teile zu erkennen und auch als solche zu nutzen sind.

Abschließend ist daher festzuhalten, dass für die Unternehmen sowohl durch eine geschickte Ausgestaltung des Vertriebs als auch durch eine Eindeutige technische Grenzziehung zwischen Eigenentwicklungen und Open Source Programmen, zahlreiche Möglichkeiten bestehen, proprietäre Entwicklungen im Open Source Umfeld vorzunehmen, ohne zugleich von Pflicht zur Offenlegung des Quellcodes betroffen zu werden.

H. Ausblick auf die GPL v3

Die FSF arbeitet seit Anfang 2006 daran, die Version 3 der GPL zu erstellen. Seit dem 27. Juli 2006 ist der 2. Entwurf der GPL v3 öffentlich zur Diskussion gestellt worden. Die offizielle Fassung soll spätestens im März 2007 folgen.[609] Erklärtes Ziel des Entwurfs ist es, den technischen und rechtlichen Veränderungen der letzten Jahre Rechnung zu tragen.[610]
Im Folgenden werden zunächst die wesentlichen Änderungen im Hinblick auf die Version 2 der GPL dargestellt, bevor sodann eingehender auf die, für die vorliegende Untersuchung maßgeblichen, Änderungen am Copyleft-Effekt eingegangen wird.

I.Internationalisierung der GPL

Neben einer Anpassung an die technischen und rechtlichen Veränderungen der letzten Jahre soll die GPL v3 vor allem auch dem Umstand Rechnung tragen, dass die Lizenz weltweit zum Einsatz gelangt und mithin zahlreichen Rechtsordnungen genügen muss.[611] Aus diesem Grund wurden vereinzelt die Begrifflichkeiten im Entwurf der GPL v3 ausgetauscht, um hierdurch eine Bindung an deren gegebenenfalls beschränkte nationale Bedeutung zu vermeiden.
So wurde beispielsweise der Begriff *„distribute“* aus Ziff. 0 der GPL v2 durch den Begriff *“propagate”*[612] ersetzt, um klarzustellen, dass dem Lizenznehmer mit Abschluss der GPL sämtliche Nutzungsrechte eingeräumt werden, unabhängig davon, welche abweichende Begrifflichkeit das nationale Recht hierfür im Einzelfall vorsieht. Hierdurch erübrigen sich künftig etwa Diskussionen darüber, ob vom Begriff *„distribute“* auch die öffentliche Zugänglichmachung erfasst wird.[613]
Zudem wurde auch die in Ziff. 0 GPL v2 enthaltene Definition des *„work based on the Program“* ersetzt und künftig nicht mehr auf den Begriff des *„derivative work“*, sondern auf sämtliche Veränderungen abgestellt, für die es nach dem jeweils anwendbaren Urheberrecht

[609] Vgl. http://gplv3.fsf.org/process-definition.
[610] Vgl. http://gplv3.fsf.org.
[611] *Free Software Foundation*, Opinion on Denationalization of Terminology, S. 1, http://gplv3.fsf.org/denationalization-dd2.pdf
[612] Der Begriff wird in Ziff. 0 GPL v3 wie folgt definiert: *”...To **"propagate"** a work means doing anything with it that requires permission under applicable copyright law, except executing it on a computer, or making modifications that you do not share. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well…“.*
[613] Hieran zweifelte bislang *Spindler*, Rechtsfragen bei Open Source, C. Rn. 76 ff; a.A. *Jaeger / Metzger*, Open Source Software, Rn. 29; ebenso *Andréewitch*, MR 2005, 36, 38.

einer Zustimmung bedarf.[614] Im Ergebnis führen diese Begriffsanpassungen zwar zu keinen konkreten Änderungen an dem bereits vorherrschenden Verständnis der Begriffe in Deutschland, jedoch können künftig – stärker als zuvor – auch nationale Auslegungstendenzen Berücksichtigung finden.[615]

Eine aus deutscher Sicht wichtige Änderung hat sich im Hinblick auf den in Ziff. 11 und 12 GPL v2 enthaltenen Haftungs- und Gewährleistungsausschluss ergeben. Um eine verbesserte internationale Verwendbarkeit der Lizenz zu ermöglichen, wurde in Ziff. 7 b Nr. 2 GPL v3 die Möglichkeit aufgenommen, abweichende – und somit dem nationalen Recht entsprechende – Regelungen aufzunehmen. Entgegen dem ersten Entwurf der GPL v3[616] wurde zudem das Erfordernis gestrichen, dass dies nur für zusätzlich hinzugefügte Softwarebestandteile gelten würde,[617] weshalb für deutsche Urheber nunmehr erstmals die Möglichkeit besteht, wirksame Haftungs- und Gewährleistungsregelungen aufzunehmen.

Trotz dieser Veränderungen ist das Ziel einer globalen Lizenz zumindest im Entwurf noch nicht vollständig erfüllt worden, wie insbesondere aus Ziff. 3 GPL v3 ersichtlich wird, in welcher im Hinblick auf technische Schutzmaßnahmen noch eine ausdrückliche Bezugnahme auf das US-Recht enthalten ist.[618]

Interessant ist zudem, dass der Entwurf in Ziff. 7 b GPL v3 unter anderem die Möglichkeit einer Rechtswahl ausdrücklich ausschließt,[619] weshalb auch in Zukunft weiterhin Unsicherheiten im Zusammenhang mit der Frage nach dem anwendbaren Recht zu erwarten sind.

---

[614] In Ziff. 0 GPL v3 heißt es nunmehr: *"…A work "based on" another work means any modified version, formation of which requires permission under applicable copyright law…."*; missverständlich ist insofern die inoffizielle deutsche Übersetzung, die den entsprechenden Teil des Entwurfs mit: *„Ein auf einem anderen Werk „basierendes Werk" bezeichnet ein modifiziertes Werk, dessen Erstellung bei Anwendung des Urheberrechts eine Erlaubnis erfordert."* übersetzt und es damit unerwähnt lässt, dass die GPL nicht nur wie bislang auf das *„copyright law"* abstellt, was auch eine Auslegung entsprechend des US-Rechts zulassen könnte, sondern als Folge der Internationalisierung ausdrücklich auf das *"**applicable** copyright law"* Bezug nimmt.

[615] Bislang war im Rahmen der Auslegung auch auf den Bedeutungsgehalt der Begriffe im US-Recht Rücksicht zu nehmen, vgl. Kapitel 5 S. 118.

[616] Ziff. 7 GPL v3 sah im ersten Entwurf noch folgendes vor: *"When you release a work based on the Program, you may include your own terms covering added parts for which you have, or can give, appropriate copyright permission…"*, vgl. http://gplv3.fsf.org/gpl-draft-2006-01-16.html

[617] Unter diesen Voraussetzungen wäre die Auswirkung von Ziff. 7 GPL wohl nur sehr begrenzt gewesen; zu Recht kritisiert von *Jaeger / Metzger*, Open Source Software, Rn. 71.

[618] Vgl. Ziff. 3 GPL v3: *"…No covered work constitutes part of an effective technological "protection" measure under section 1201 of Title 17 of the United States Code…"*; eine Bezugnahme auf die InfoSocRL schlagen *Wiebe / Heidinger*, MR 2006, 258, 259, vor.

[619] Die maßgebliche Regelung in Ziff. 7 b GPL v3 lautet: *"All other additional requirements, including attorney's fees provisions, choice of law, forum, and venue clauses, arbitration clauses, mandatory contractual acceptance clauses, requirements regarding changes to the name of the work, and terms that require that conveyed copies be governed by a license other than this License, are prohibited."*. Zu den Gründen für die Aufnahme eines solchen Verbots, vgl. *Free Software Foundation*, Opinion on Denationalization of Terminology, S. 3 f, http://gplv3.fsf.org/denationalization-dd2.pdf.

II.Softwarepatente

Eine Neuerung im Entwurf der GPL v3, die bereits für zahlreiche Diskussionen gesorgt hat, ist die Aufnahme einer Regelung zu Softwarepatenten in Ziff. 11 GPL v3.[620] Während der erste Entwurf zur GPL v3 noch die Erteilung einer einfachen Patentlizenz an den Programmnutzer vorgesehen hatte,[621] soll nach dem zweiten Entwurf mit der Verbreitung eines GPL-lizenzierten Programms zugleich der Verzicht darauf verbunden sein, die Nutzung des Programms durch die Geltendmachung patentrechtlicher Ansprüche zu behindern.
Die Reichweite der Ziff. 11 GPL v3 wird nicht zuletzt deshalb kritisiert,[622] weil der Verzicht auch künftige Patentrechte erfasst und die Geltendmachung von Ansprüchen zudem dann ausgeschlossen sein soll, wenn erst die Bearbeitungen Dritter in den Schutzbereich des Patents eingreifen. Es bleibt daher abzuwarten, ob sich auch Unternehmen mit großem Patentbestand auf das mit einem solchen Verzicht einhergehende Risiko einlassen werden.[623]

III.Technische Schutzmaßnahmen

Umfassende Diskussionen hat es zudem im Kontext mit dem von der GPL v3 als „Digital Restriction Management" bezeichneten Themenbereich gegeben.[624] Während sich der erste Entwurf zur GPL v3 noch pauschal gegen das „Digital Restriction Management" ausgesprochen hat,[625] wurde im zweiten Entwurf auf die hieran geäußerte Kritik reagiert und nur noch die Unvereinbarkeit freier Software mit Trusted Computig in der Präambel ausgedrückt.[626]
Die aktuelle Fassung der GPL v3 beinhaltet in Ziff. 3 eine Regelung zu DRM-Systemen. Zwar enthält die Regelung kein generelles Verbot der Integration von GPL-lizenzierten Werken in Kopierschutzsystemen, allerdings wird der Versuch unternommen, diese

[620] Vgl. Ziff. 11 GPL v3: *"You receive the Program with a covenant from each author and conveyor of the Program, and of any material, conveyed under this License, on which the Program is based, that the covenanting party will not assert (or cause others to assert) any of the party's essential patent claims in the material that the party conveyed, against you, arising from your exercise of rights under this License…"*
[621] Vgl. Ziff. 11 GPL v3: *"When you distribute a covered work, you grant a patent license to the recipient, and to anyone that receives any version of the work…"*; veröffentlicht unter http://gplv3.fsf.org/gpl-draft-2006-01-16.html.
[622] Vgl. u.a. *Wiebe / Heidinger*, MR 2006, 258, 260; *Jaeger*, Eine Lizenz entsteht, S. 1, http://www.heise.de/open/artikel/76248. Aus der Industrie hat sich insbesondere Hewlett Packard kritisch gegenüber der Regelung geäußert, vgl. http://www.zdnetasia.com/news/software/0,39044164,39378902,00.htm.
[623] Von einem unkalkulierbaren Risiko gehen *Wiebe / Heidinger*, MR 2006, 258, 260, aus.
[624] Vgl. nur die Kritik Torvalds, http://www.zdnetasia.com/news/software/0,39044164,39378902,00.htm.
[625] Die Ausführungen in der Präambel des ersten Entwurfs lauteten: *"… DRM is fundamentally incompatible with the purpose of the GPL…"*; veröffentlicht unter http://gplv3.fsf.org/gpl-draft-2006-01-16.html.
[626] Vgl. Präambel GPL v3: *"…Some computers are designed to deny users access to install or run modified versions of the software inside them. This is fundamentally incompatible with the purpose of the GPL…"*.

weitgehend zu erschweren. Zu diesem Zweck ist in Ziff. 3 eine Regelung aufgenommen worden, wonach die Verwendung von GPL-lizenziertem Code in einem DRM-System zugleich als Zustimmung zu dessen Umgehung zu werten sein soll.[627]

An der Wirksamkeit dieser Vermutung sind in der Literatur bereits begründete Zweifel geäußert worden, die zum Einen daraus resultieren, dass die in der GPL enthaltene Zustimmung zur Umgehung von technischen Schutzmaßnahmen ausschließlich vom Rechtsinhaber des geschützten Werks, nicht aber vom Urheber der entsprechenden Software, erteilt werden kann.[628] Zum Anderen wäre aber selbst in denjenigen Fällen, in denen der Rechtsinhaber der entsprechenden Inhalte zugleich Urheber der technischen Schutzmaßnahme wäre, die Annahme einer solchen Zustimmung vor dem Hintergrund widersprüchlich, dass der Rechtsinhaber die betreffenden Werke offensichtlich mit einer technischen Schutzmaßnahme versieht und damit seinen entgegenstehenden Willen zum Ausdruck bringt.[629]

IV. Rechtsfolgen bei Lizenzverletzungen

Gegenüber der GPL v2 unterscheidet sich der aktuelle Entwurf grundlegend in der Rechtsfolge, die für diejenigen Fälle vorgesehen ist, in denen es zu Verstößen gegen die Lizenzbedingungen kommt.

Hat eine Verletzung der Lizenz nach Ziff. 4 GPL v2 noch zu einem automatischen Erlöschen der Nutzungsrechte geführt, sieht die GPL v3 nunmehr in Ziff. 8 vor, dass dem Rechtsinhaber lediglich Kündigungsrecht zusteht, welches zudem von der vorherigen Unterrichtung des Verletzers abhängig und innerhalb einer 60-Tages-Frist nach der letzten Verletzungshandlung geltend zu machen ist.[630]

---

[627] Vgl. Ziff. 3 GPL v3: *"...When you convey a covered work, you waive any legal power to forbid circumvention of technical measures that include use of the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing the legal rights of third parties against the work's users."*; der Hintergrund dieser Klausel ist letztlich in den Regelungen zum DMCA und der europäischen Richtlinie zum Urheberrecht in der Informationsgesellschaft zu sehen (vgl. *Jaeger*, Eine Lizenz entsteht, S. 1, http://www.heise.de/open/artikel/76248), in denen die Umgehung von technischen Schutzmaßnahmen verboten wird.

[628] *Jaeger*, Eine Lizenz entsteht, S. 1, http://www.heise.de/open/artikel/76248; im Hinblick auf das insofern vergleichbare österreichische Recht, vgl. *Wiebe / Heidinger*, MR 2006, 258, 260.

[629] Ebenso *Wiebe / Heidinger*, MR 2006, 258, 260, allerdings im Zusammenhang mit dem österreichischen Recht.

[630] Ziff. 8 GPL v3: *"...If you violate this License, any copyright holder may put you on notice by notifying you of the violation, by any reasonable means, provided 60 days have not elapsed since the last violation..."*.

Hintergrund für die Veränderung der aktuell geltenden Ziff. 4 GPL v2, deren Wirksamkeit im deutschen Recht bereits mehrmals gerichtlich bestätigt wurde,[631] ist die Auffassung der FSF, wonach der durch eine Verletzungshandlung verursachte automatische Rechtswegfall im US-Recht nicht durch ein lizenzkonformes Verhalten geheilt werden kann, sondern einer ausdrücklichen Rechtseinräumung durch den Rechtsinhaber bedarf.[632] Eine solche ausdrückliche Rechtseinräumung wird jedoch, aufgrund der im Open Source Bereich vorherrschenden Verbreitungsstrukturen, kaum einmal zu erlangen sein und erweist sich nicht zuletzt deshalb als unpraktikabel.

Die Literatur steht der geplanten Änderung überwiegend kritisch gegenüber, da mit ihr eine Schwächung der Rechtsdurchsetzung befürchtet wird. Immerhin werde der Rechtsinhaber nur selten innerhalb der 60-Tage-Frist Kenntnis von der Verletzungshandlung bekommen.[633]

Dem kann allerdings nur bedingt gefolgt werden. Betrachtet man nämlich die bislang bekannt gewordenen Lizenzverletzungen,[634] so zeichnen sich diese allesamt dadurch aus, dass die jeweiligen Verletzungshandlungen in einem Vertrieb von Produkten bestanden, bei denen die Voraussetzungen der GPL nicht erfüllt wurden. Die für Ziff. 8 GPL v3 maßgebliche letzte Verletzungshandlung würde somit in der jeweils letzten Vertriebshandlung bestehen, die aber jedenfalls über die gesamte Produktlaufzeit gegeben ist und sich auch zeitlich durch Probekäufe nachweisen lässt. Die Gefahr, dass der Rechtsinhaber nicht innerhalb der 60-Tage-Frist Kenntnis von der Verletzungshandlung erhält, besteht daher lediglich bei solchen Produkten, die ausschließlich über einen sehr kurzen Zeitraum vertrieben werden und deren Marktrelevanz daher eher unbedeutend bleiben dürfte.

V.Lizenzkompatibilität

Um die Kompatibilität der GPL mit anderen Open Source Lizenzen zu verbessern,[635] wurde in Ziff. 7 GPL v3 nunmehr die Möglichkeit eingeführt von einzelnen Lizenzbedingungen abzuweichen. Neben der bereits angesprochenen Möglichkeit wirksame Haftungs- und Gewährleistungsregelungen aufzunehmen, sieht Ziff. 7 des Entwurfs zahlreiche weitere

[631] Eingehend LG Frankfurt/M CR 2006, 729, 731 ff; Beschluss des LG Berlin CR 2006, 735; LG München I CR 2004, 774, 775 f.
[632] Vgl. *Jaeger / Metzger*, c't 2006, 46, 46.
[633] *Jaeger / Metzger*, c't 2006, 46, 46; *Wiebe / Heidinger*, MR 2006, 258, 259.
[634] LG Frankfurt/M CR 2006, 729; Beschluss des LG Berlin CR 2006, 735; LG München I CR 2004, 774; eine Auflistung einiger außergerichtlicher Verfahren findet sich unter http://www.gpl-violations.org.
[635] Die Relevanz der Probleme, die aus der Inkompatibilität von Open Source Lizenzen folgt, lässt gut mit einem Blick auf die Liste derjenigen Lizenzen ermessen, die – jedenfalls nach Auffassung der FSF – mit der GPL v2 inkompatibel sein sollen, vgl. http://www.fsf.org/licensing/licenses/index_html#GPLIncompatibleLicenses.

Änderungsmöglichkeiten vor, die in die Kategorien „zusätzliche Genehmigungen“ und „zusätzliche Anforderungen“[636] unterteilt wurden.

Unter die Regelung der „zusätzlichen Genehmigungen“ fallen künftig solche Fälle, in denen Code zusammen mit GPL-lizenzierten Programmen verwendet werden soll, der unter keiner oder einer abgeschwächten Copyleft-Klausel lizenziert wurde. Dementsprechend wird die Programmverbindung mit BSD-lizenziertem Code nunmehr ausdrücklich von Ziff. 7 a GPL v3 gestattet. Weiterhin wird auch die LGPL künftig nicht mehr als eigenständiges Lizenzmodell, sondern vielmehr als „zusätzliche Genehmigung“ im Sinne von Ziff. 7 a GPL v3 verstanden.[637]

Mit der Regelung der „zusätzlichen Anforderungen“ werden solche Fälle ermöglicht, in denen eine Programmverbindung zwischen GPL-lizenzierten Programmen und Code vorgesehen ist, dessen Lizenz zusätzliche Bedingungen enthält. Hierdurch wird unter anderem auch die in der Praxis bedeutsame Programmverbindung mit Code, der unter der Apache License v2 steht, ermöglicht,[638] da die bislang fehlende Möglichkeit der Kündigung für den Fall von Patentklagen gem. Ziff. 7 b Nr. 5 GPL v3 aufgenommen werden kann.[639]

Trotz der Aufnahme von diesen Änderungsmöglichkeiten im Entwurf der GPL v3 ist jedoch nicht zu erwarten, dass damit das Problem der Lizenzinkompatibilität abschließend gelöst wurde.[640] Vielmehr wird der Frage, ob eine Verbindung von Programmen, die unter verschiedenen Lizenzen veröffentlicht wurden, zulässig ist, aufgrund der großen Zahl an verschiedenen Open Source Lizenzen auch in Zukunft, eine erhebliche Bedeutung zukommen.

---

[636] So die Bezeichnung in der inoffiziellen deutschen Übersetzung, vgl. http://www.gnu.de/documents/gplv3.de.html. In der englischen Originalfassung werden in Ziff. 7 GPL v3 die Begriffe *„Additional Permissions“* und *„Additional Requirements“* verwendet.

[637] *Free Software Foundation*, Opinion on Denationalization of Terminology, S. S. 17 Fn. 61, http://gplv3.fsf.org/denationalization-dd2.pdf; - “*We offer version 3 of the GNU LGPL as a model for the use of additional permissions as exceptions from requirements of the GPL.”*.

[638] Vgl. *Jaeger / Metzger*, c't 2006, 46, 46.

[639] Die Apache License v2 wurde daher von der FSF zu den inkompatiblen Lizenzen gezählt, vgl. http://www.fsf.org/licensing/licenses/index_html#GPLIncompatibleLicenses: *“…The Apache Software License is incompatible with the GPL because it has a specific requirement that is not in the GPL: it has certain patent termination cases that the GPL does not require.”*.

[640] *Wiebe / Heidinger*, MR 2006, 258, 260, gehen – freilich ohne Benennung spezifischer Lizenzen – sogar davon aus, dass von den derzeit vorgesehenen Änderungsmöglichkeiten lediglich Randbereiche erfasst würden und die GPL v3 auch künftig zu wesentlichen Open Source Lizenzen inkompatibel bleibt.

VI. Copyleft-Effekt

Nahezu unbemerkt wurden im aktuellen Entwurf zur GPL v3 zudem einige Änderungen am Copyleft-Effekt vorgenommen, die jedoch – zumindest nach Auffassung der FSF – keine inhaltliche Neuausrichtung zur Folge haben sollen.[641] Dies trifft indes nur insoweit zu, als dass sich an dem Grundprinzip des Copyleft-Effekts nichts Wesentliches geändert hat.[642] Andererseits wurden im aktuellen Entwurf aber zahlreiche Änderungen vorgenommen, die durchaus dazu geeignet sind, neue Diskussionen über die Reichweite der GPL zu verursachen.[643]

1. Ziff. 9 GPL v3 – "using peer-to-peer transmission"

Eine erste Änderung, die im Zusammenhang mit dem Copyleft-Effekt zugegebenermaßen eher unbedeutend ist, folgt aus Ziff. 9 GPL v3, in dem nunmehr ausdrücklich eine Regelung zum Erwerb von GPL-lizenzierten Programmkopien über Peer-to-Peer Netzwerke enthalten ist.[644] Hiernach soll allein das Herunterladen einer GPL-lizenzierten Programmkopie über ein Peer-to-Peer Netzwerk nicht zugleich als Annahme der GPL gewertet werden, selbst wenn hierin – wie beispielsweise beim Download von sog. Torrent-files –[645] zugleich eine zustimmungsbedürftige öffentliche Zugänglichmachung liegt.

Problematisch an dieser Regelung ist allerdings, wie sich die hierin vorgesehene Rechtsfolge im Deutschen Recht realisieren lässt. Diese Frage drängt sich insofern auf, als dass es für die während des Downloads erfolgende öffentliche Zugänglichmachung gem. § 69 c Nr. 4 UrhG zwangsläufig einer Zustimmung durch den Rechtsinhaber bedarf. Als eine solche Zustimmung ließe sich zwar die Regelung in Ziff. 9 GPL v3 unproblematisch auslegen, allerdings vermag sie erst nach dem Abschluss der GPL Wirkung zu entfalten, so dass sie letztlich zwecklos wäre. Diese Rechtsfolge lässt sich ausschließlich dadurch vermeiden, dass bereits im Angebot eines Programms, dass unter der GPL lizenziert wurde – jedenfalls aber im Anbieten von Downloadmöglichkeiten über ein Peer-to-Peer Netzwerk – die konkludente

---

[641] *Free Software Foundation*, GPL v3 Second Discussion Draft Rationale, S. 12 f, http://gplv3.fsf.org/rationale.

[642] Ähnlich in der Einschätzung auch *Wiebe / Heidinger*, MR 2006, 258, 259, die allerdings lediglich Änderungen hinsichtlich der Definition des Quellcodes und der Möglichkeit diesen zur Verfügung zu stellen, problematisieren.

[643] Insofern weist *Jaeger*, Eine Lizenz entsteht, S. 2, http://www.heise.de/open/artikel/76248, zu Recht darauf hin, dass der Interpretation über die Reichweite des Copyleft-Effekts neuer Spielraum eröffnet wurde.

[644] Ziff. 9 GPL v3: *„You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance…"*

[645] Vgl. hierzu bereits S. 104, Fn. 427.

Zustimmungshandlung im Sinne von § 69 c Nr. 4 UrhG gesehen wird, die der Nutzer in Kenntnis dieser Eigenschaften ebenfalls konkludent annimmt.
Dies lässt sich zunächst sicherlich in solchen Fällen vertreten, in denen der Rechtsinhaber selbst einen Download über ein Peer-to-Peer Netzwerk anbietet. Aber auch wenn dies erst durch einen Dritten erfolgt, lässt sich wohl vertreten, dass in dessen Downloadangebot zugleich die erforderliche Zustimmung des Rechtsinhabers liegt.

2.Ziff. 2 GPL v3 – "output from running"

Eine weitere Änderung, die der aktuelle Entwurf gegenüber der GPL v2 aufweist, lässt sich in Ziff. 2 GPL v3 finden, wo nunmehr ausdrücklich darauf hingewiesen wird, dass sich die von der GPL vermittelte Lizenzierungspflicht ausschließlich dann auf das Arbeitsergebnis einer Programmbenutzung erstreckt, wenn dieses aus urheberrechtlicher Sicht als zustimmungsbedürftige Bearbeitung einzustufen wäre. Hierdurch ändert sich zwar nichts an der bisherigen Rechtslage, allerdings erübrigen sich künftig Diskussionen darüber, ob allein das Arbeitsergebnis, welches durch die Benutzung eines GPL-lizenzierten Programms hergestellt wird, bereits von der Lizenzierungspflicht erfasst wird.[646]

3.Ziff. 5 Abs. 1 S. 1 GPL v3 - "convey"

Stellt man die Kernregelung des Copyleft-Effekts in Ziff. 2 GPL v2 der neuen Fassung in Ziff. 5 GPL v3 gegenüber, so hat sich hieran vom äußeren Umfang her tatsächlich wenig getan, was allerdings nichts daran zu ändern vermag, dass auch die bislang erfolgten „kleineren" Anpassungen Diskussionspotenzial aufweisen.
Eine erste Anpassung ist bereits in Ziff. 5 Abs. 1 S. 1 GPL v3 enthalten, indem hierin für die Lizenzierungspflicht nicht mehr an eine Weitergabe von veränderten Programmversionen im Sinne von *„distribute"*, sondern stattdessen an den Begriff *„convey"* angeknüpft wird. Ziel dieser Änderung war es, die durch den Begriff *„distribute"* gegebene Koppelung an dessen mitunter differierende Bedeutung im jeweils anwendbaren nationalen Recht aufzuheben und

[646] Vgl. zu den Überlegungen hierzu bereits die Ausführungen oben Kapitel 5 S. 127, sowie *Wuermeling / Deike*, CR 2003, 87, 87 f.

im Hinblick auf die internationale Verwendung der GPL einen Begriff zu verwenden, der rechtlich nicht „vorbelastet“ ist.[647]

Der Begriff *„convey“* ist in Ziff. 0 GPL v3 definiert[648] und umfasst sämtliche Übertragungsvorgänge, bei denen es einem Dritten ermöglicht wird, das Werk zu kopieren oder Kopien zu erhalten. Bei unbefangenem Lesen könnten diese Voraussetzungen eine Erweiterung gegenüber dem bisherigen Begriffsverständnis darstellen, da die im Verbreitungsbegriff nach deutschem Recht enthaltene Beschränkung für die Weitergabe von Programmkopien im persönlich verbundenen Umfeld[649] hierin keinen Platz zu haben scheint. Bei einer solchen Auslegung würde indes unberücksichtigt bleiben, dass die Fälle, in denen eine Übertragung im Sinne von *„convey“* vorliegt, lediglich eine Teilmenge derjenigen Fälle darstellen, in denen eine Verbreitungshandlung im Sinne von *„propagate“* gegeben ist – für die es also im Hinblick auf das anwendbare Recht einer Zustimmung des Rechtsinhabers bedarf. Mit der Einführung des Begriffs *„convey“* wurde bezweckt, dass der Nutzer nicht in sämtlichen Fällen, in denen das anwendbare nationale Recht gegebenenfalls eine zustimmungsbedürftige Vertriebstätigkeit vorsieht, von den Pflichten des Copyleft-Effekts erfasst wird. Der Copyleft-Effekt sollte vielmehr nur dann eingreifen, wenn tatsächlich eine Übertragung der Programmkopie in der Weise stattfindet, dass der Empfänger eine Programmkopie oder die Möglichkeit zur Erstellung einer solchen erhält.[650]

4.Ziff. 5 Abs. 1 S. 1 GPL v3 – “or the modifications to produce it”

Eine „echte“ Neuerung stellt hingegen der Zusatz in Ziff. 5 Abs. 1 S. 1 GPL v3 dar, wonach nicht nur die modifizierten Programmkopien, sondern auch diejenigen Werkteile unter der GPL zu lizenzieren sind, mit denen solche Modifikationen hergestellt werden. Damit sollen – ausweislich der Entwurfsbegründung – Patches von der Lizenzierungspflicht erfasst werden.[651] Die FSF hat offensichtlich die Gefahr erkannt, dass nach der bisherigen Copyleft-

---

[647] Vgl. *Free Software Foundation*, Opinion on Denationalization of Terminology, S. 1 f, http://gplv3.fsf.org/denationalization-dd2.pdf.

[648] Ziff. 0 GPL v3: *“...To "convey" a work means any kind of propagation that enables other parties to make or receive copies, excluding sublicensing...”*.

[649] Zur Auslegung des Öffentlichkeitsbegriffs im deutschen Recht vgl. nur *Dreier* in: Dreier / Schulze, § 15 Rn. 44 f, m.w.N.

[650] Ausgeschlossen ist insofern die Herstellung von Programmkopien für den eigenen Gebrauch. Ebenso dürfte auch der Einsatz von Open Source Software in ASP-Lösungen ausgeschlossen sein, sofern der Nutzer keine Client-Software installieren muss; so wohl auch *Jaeger / Metzger*, Open Source Software, Rn. 65.

[651] *Free Software Foundation*, GPL v3 Second Discussion Draft Rationale, S. 12 Fn. 43, http://gplv3.fsf.org/rationale - “*Conveying a patch that is used to produce a modified version is equivalent to conveying the modified version itself.”*.

Regelung zumindest die theoretische[652] Möglichkeit bestand Patches zu GPL-lizenzierten Programmen unter proprietären Lizenzen zu verbreiten, sofern diese keinen urheberrechtlich geschützten GPL-Code enthalten.

Zweifelhaft ist allerdings, ob der Zusatz tatsächlich dazu geeignet ist, die Lizenzierungspflicht auf solche Patches zu erstrecken. Es ist nämlich zu beachten, dass Ziff. 5 GPL ausschließlich dann Wirkung entfalten kann, wenn die GPL zuvor, zumindest in konkludenter Form durch die Vornahme einer zustimmungsbedürftigen Handlung, angenommen wurde. Eine solche konkludente Annahme wird jedoch regelmäßig fehlen, wenn sich Unternehmen darauf beschränken isolierte Patches zu vertreiben, die keine urheberrechtlich geschützten GPL-Codeteile enthalten, da diese weder eine vorherige Bearbeitung des GPL-Programms voraussetzen noch in deren Vertrieb eine zustimmungsbedürftige Verbreitung liegt. Es bleibt also auch unter der GPL v 3 weiterhin möglich isoliert Patches für GPL-lizenzierte Programme zu vertreiben, in denen kein urheberrechtlich geschützter Code enthalten ist und für deren Erstellung der Abschluss der GPL – etwa durch eine zustimmungsbedürftige Bearbeitungshandlung – nicht erforderlich war.

5.Ziff. 5 Abs. 2 GPL v3 – "not specifically for use in combination"

Eine Änderung, mit der nach Ansicht der FSF lediglich eine Neuformulierung, nicht aber eine inhaltliche Ausweitung des Copyleft-Effekts beabsichtigt wurde,[653] liegt in Ziff. 5 Abs. 2 GPL v3, wonach der getrennte Vertrieb von eigenständigen und unabhängigen Werkteilen künftig nicht mehr dazu ausreichen soll, die Lizenzierungspflicht entfallen zu lassen, sofern die betreffenden Werkteile speziell für einen gemeinsamen Einsatz ausgestaltet sind.[654] Durch diesen Zusatz wurde letztlich Ziff. 2 Abs. 2 S. 3 GPL v2[655] ersetzt, wonach es für die Frage der Lizenzierungspflicht noch auf die Unterscheidung ankam, ob eigenständige und unabhängige Werkteile als einheitliches Ganzes vertrieben wurden.[656] Dies wurde im

[652] Die praktische Relevanz dürfte hingegen eher gering sein, da die Patches zumeist auch urheberrechtlich geschützte Werkteile beinhalten und bereits aus diesem Grund unter der GPL zu lizenzieren sind; vgl. hierzu bereits Kapitel 5 S. 124.

[653] *Free Software Foundation*, GPL v3 Second Discussion Draft Rationale, S. 13, http://gplv3.fsf.org/rationale.

[654] Ziff. 5 Abs. 2 GPL v3: *„…To the extent that identifiable sections of the modified work, added by you, are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you convey them as separate works,* ***not specifically for use in combination with the Program****…"*.

[655] Ziff. 2 Abs. 2 S. 3 GPL v2: *"…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."*.

[656] Ausführlich hierzu vgl. Kapitel 5 ab S. 165.

deutschen Recht dahin gehend ausgelegt,[657] dass sich die Lizenzierungspflicht bei einem gemeinsamen Vertrieb von eigenständigen und unabhängigen Werkteilen, immer dann auf die eigenen Entwicklungen erstreckt, wenn diese in einer Form überlassen werden, in der es dem Nutzer nicht möglich ist, die Werke als Eigenständige zu erkennen und auch getrennt zu nutzen.[658]

Anhand der neuen Regelung, kommt es nun offensichtlich nicht mehr darauf an, wie sich der Vertrieb aus Sicht des Nutzers darstellt, sondern einzig darauf, welcher Einsatzzweck damit verfolgt wird. Es stellt sich mithin die Frage, ab wann Werke, die grundsätzlich als eigenständig und unabhängig anzusehen sind, zugleich darauf ausgerichtet sind, speziell miteinander verwendet zu werden.

Ist dies nur dann der Fall, wenn in technischer Hinsicht kein anderer sinnvoller Einsatzzweck für die betreffende Komponente besteht, oder liegt eine spezielle Ausrichtung auf einen gemeinsamen Einsatzzweck auch dann vor, wenn sie zwar in einer Vielzahl von unterschiedlichen Programmen eingesetzt werden kann, aber ausschließlich für den Einsatz mit der betreffenden GPL-Komponente beworben wird?

Um den Anwendungsbereich der Ziff. 5 Abs. 2 GPL v3 nicht ausufern zu lassen und um eine – nicht zuletzt auch aus AGB-rechtlicher Sicht gebotene –[659] eindeutige Abgrenzung zu ermöglichen, wird es wohl ausschließlich auf eine technische Betrachtungsweise ankommen können.[660] Dies Bedeutet jedoch zugleich auch eine starke Schwächung der Regelung, da die betroffenen Unternehmen die Rechtsfolge einer Lizenzierungspflicht letztlich dadurch umgehen könnten, dass sie zumindest *eine* alternative Einsatzmöglichkeit für die betreffende Komponente anbieten und eine spezifische Ausrichtung damit ausschließen.

Die Reichweite der Regelung wird aber letztlich auch dadurch begrenzt, dass sie in denjenigen Fällen, in denen sich Unternehmen darauf beschränken ausschließlich eigene Entwicklungen zu vertreiben, nur selten Wirkung entfalten wird. Immerhin wird es nur schwer nachzuweisen sein, dass die GPL auch durch die Vornahme einer zustimmungsbedürftigen Handlung im Sinne von § 69 c UrhG konkludent angenommen

[657] Vgl. vor allem *Jaeger / Metzger*, Open Source Software, Rn. 52.
[658] Vgl. Kapitel 5 ab S. 165.
[659] Würde es nicht ausschließlich auf den technischen Einsatzzweck ankommen, könnten Bedenken im Hinblick auf das in § 307 Abs. 1 S. 2 BGB enthaltene Transparenzgebot in diesem Kontext geboten sein (im Zusammenhang mit Ziff. 2 GPL v2, vgl. S. 41 Fn. 170, sowie zu den Bedenken hieran *Spindler*, Rechtsfragen bei Open Source, C. Rn. 120).
[660] Für die Beurteilung, ob eine Programmkomponente speziell für einen gemeinsamen Einsatz ausgerichtet ist, wird es auch künftig maßgeblich auf die Ausgestaltung der Schnittstellen ankommen (Einzelheiten hierzu vgl. Kapitel 5 S. 146 ff). Sind diese in einer Art und Weise ausgestaltet, dass sich die Programmkomponente ohne inhaltliche Veränderung auch in einem anderen Umfeld einsetzen lässt, so wird die Annahme einer *speziellen* Ausrichtung wohl nicht haltbar sein. Beim Vertrieb der entsprechenden Programmkomponente kann der Copyleft-Effekt deshalb nicht eingreifen.

wurde, wenn eine solche Handlung nicht bereits für die eigentliche Vertriebstätigkeit erforderlich ist.

6.Ziff. 5 Abs. 3 GPL – "compilation"

Eine weitere sprachliche Änderung findet sich in Ziff. 5 Abs. 3 GPL, in der nun nicht mehr auf den Begriff *„mere aggregation"*, sondern auf den der *„compilation"* abgestellt wird.
In inhaltlicher Sicht hat sich, trotz dieser sprachlichen Änderung, nichts daran geändert, dass der Copyleft-Effekt sich ausschließlich auf diejenigen Werke bezieht, die ein *„work based on the Program"* im Sinne von Ziff. 5 Abs. 1 GPL v3 darstellen und für deren Erstellung und Vertrieb es mithin einer Zustimmung der Rechtsinhaber bedarf. Somit kommt es auch künftig maßgeblich darauf an, unter welchen Voraussetzungen von der Schaffung eines eigenständigen Werks auszugehen ist.[661]
Dies wird nicht zuletzt auch daraus ersichtlich, dass in Ziff. 5 Abs. 3 GPL v3 darauf Bezug genommen wird, dass die Zusammenstellung mehrerer Werke nur in solchen Fällen keine Lizenzierungspflicht zur Folge hat, in denen die jeweiligen Werkteile ihrer Natur nach nicht Erweiterungen des betroffenen Werks sind.[662]
Bei unbefangenem Lesen dieses Abschnitts wird freilich nicht ohne weiteres ersichtlich, ob hierin lediglich ein Beispiel dafür liegt, was ohnehin bereits aus Ziff. 5 Abs. 2 GPL v3 folgt, oder ob die Lizenzierungspflicht nicht konstitutiv auf solche Werkteile ausgedehnt werden soll, die zwar als eigenständig und unabhängig im Sine von Ziff. 5 Abs. 2 GPL v3 anzusehen sind, in funktionaler Hinsicht aber gleichwohl im Verhältnis von Programm zu -erweiterung zueinander stehen. Wäre Letzteres der Fall, so würde die Lizenzierungspflicht durch den in Ziff. 5 Abs. 3 GPL v3 enthaltenen Nebensatz auch auf solche Werkteile erstreckt werden, die zwar nicht speziell für die Benutzung in Kombination mit dem anderen Programm ausgestaltet sind – weil sie sich etwa auch mit anderen Werken kombinieren lassen – aber ihrer Natur nach gleichwohl eine Programmerweiterung darstellen.
Gegen eine solche Auslegung spricht jedoch, dass es sodann einen Unterschied machen würde, ob eigenständige Werke im Sinne von Ziff. 5 Abs. 3 GPL v3 zusammen auf einem Speichermedium vertrieben werden – es mithin zusätzlich darauf ankommt, ob eines der Programme als Programmerweiterung zu bezeichnen ist –, oder ob sie als *„separate works"*

[661] Vgl. hierzu Kapitel 5 ab S. 137.
[662] Ziff. 5 Abs. 3 GPL v3: *"A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work…"*.

im Sinne von Ziff. 5 Abs. 2 GPL v3 weitergegeben werden und es somit – unabhängig von ihrer funktionalen Eigenschaft als Programmerweiterung – einzig darauf ankommt, ob sie nicht *speziell* auf eine Programmverbindung ausgerichtet sind.
Eine solche Differenzierung war unter der GPL v2, ausweislich der Regelung in Ziff. 2 Abs. 4 GPL v2, nicht beabsichtigt. Den Entwurfsbegründungen ist ebenfalls kein Anzeichen dafür zu entnehmen, dass sich an der Einstellung hierzu etwas Grundlegendes verändert hätte. Es ist daher auch weiterhin davon auszugehen, dass es für die Reichweite der Lizenzierungspflicht unerheblich ist, ob die verschiedenen Werkteile auf einem einheitlichen Speichermedium oder getrennt voneinander vertrieben werden.
Ziff. 5 Abs. 3 GPL v3 ist folglich so zu verstehen, dass hierin lediglich eine deklaratorische Feststellung der Reichweite der Lizenzierungspflicht liegt. Beim Vorliegen eigenständiger Werke hängt die Reichweite der Lizenzierungspflicht folglich einzig davon ab, ob diese auch als eigenständige Werke weitergegeben werden und nicht speziell auf eine Verbindung mit den GPL-lizenzierten Werkteilen ausgerichtet sind.

7.Ziff. 6 GPL v3 – "Corresponding Source"

Mit Ziff. 6 GPL v3 wird die bisherige Regelung in Ziff. 3 GPL v2 ersetzt. Trotz der Zunahme an Umfang hat sich am Regelungsgehalt der Klausel im Kontext mit dem Copyleft-Effekt nichts Wesentliches verändert. Bei der Übertragung von kompilierten Programmen, bei denen es sich definitionsgemäß um ein *„covered work"* handelt[663] trifft den Vertreiber nach wie vor die Pflicht, den kompletten korrespondierenden Quelltext in maschinenlesbarer Form mitzuliefern.[664] Hierbei ist in Ziff. 1 GPL v3 jedoch nunmehr eine Klarstellung davon enthalten, welche Quellmaterialien hierunter zu verstehen sind.
Vom *„Corresponding Source"* sind künftig sämtliche Materialien umfasst, die benötigt werden, um das Werk zu erzeugen, es zu installieren, den Objekt-Code auszuführen und um das Werk zu modifizieren, mit Ausnahme der Systembibliotheken und allgemein einsetzbarer Werkzeuge und freier Computerprogramme, die in unmodifizierter Form verwendet werden. Dementsprechend sind als *„Corresponding Source"* sowohl die zur Kompilierung

[663] Entsprechend Ziff. 0 GPL v3 fallen hierunter sowohl das unveränderte Programm als auch die Modifikationen desselben, für die es im Hinblick auf das anwendbare Urheberrecht einer Zustimmung bedarf.
[664] Diese Pflicht war auch schon in Ziff. 3 Abs. 1 a GPL v2 enthalten: *"Accompany it with the complete corresponding machine-readable source code…"*.

notwendigen Skripte, als auch dynamisch verlinkte Unterprogramme und Bibliotheken umfasst, auf die das Programm speziell[665] angewiesen ist.
Weiterhin sollen zum Quellcode neuerdings aber auch alle Schlüssel zählen, die für die Ausführung des Programms erforderlich sind, unabhängig davon, ob dieses in seiner ursprünglichen Form oder in einer modifizierten Fassung verwendet wird. Zu einer Ausweitung des korrespondierenden Quelltexts auf solche Schlüssel sah sich die FSF nicht zuletzt durch die Geschäftspraktiken einiger Hardwarehersteller gezwungen, die zwar die entsprechenden Quellen der verwendeten Open Source Programme wiederum unter der GPL v2 angeboten haben, eine Ausführung von modifizierten Programmversionen auf ihrer Hardware aber durch die Voraussetzung von Signaturen verhinderten.[666]
Die Reichweite der Regelung hat zu zahlreichen Diskussionen und Kritik Anlass gegeben und wird oftmals als zu weitgehend empfunden.[667] Es bleibt daher abzuwarten, in welcher Form sie letztlich in der offiziellen Version der GPL v3 enthalten sein wird.

8.Künftige Relevanz der GPL v2

Im Hinblick auf die für März 2007 angekündigte offizielle Version 3 der GPL bleibt letztlich noch die Frage zu klären, welche Relevanz der GPL v2 in Zukunft zukommen wird.

Dafür, dass die Bedeutung der GPL v2 auch nach März 2007 nahezu ungebrochen fortbestehen wird, spricht zunächst einmal, dass sämtliche Programme, die bislang unter der GPL v2 veröffentlicht wurden, auch weiterhin unter der GPL v2 lizenziert bleiben. Durch die

[665] Ausweislich der Entwurfsbegründung will die FSF durch die Regelung klarstellen, dass sich Lizenzierungspflicht lediglich auf diejenigen Bibliotheken bezieht, auf die das Programm zwangsläufig angewiesen ist und die dadurch nicht ohne weiteres durch andere ausgetauscht werden können, vgl. Draft S. 7 Fn. 20: *„We clarify that the shared libraries and dynamically linked subprograms that are included in Corresponding Source are those that the work is „specifically" designed to require, making it clearer that they do not include libraries invoked by the work that can readly substituted by other existing implementations."*. An dieser Stelle ist im Übrigen auch die inoffizielle deutsche Übersetzung ungenau, da hierin nicht entsprechend an die Wortwahl in Ziff. 5 Abs. 2 GPL v3 angeknüpft, sondern die maßgebliche Passage in Ziff. 1 GPL v3 mit *„konstruktionsbedingter"* Abhängigkeit übersetzt wird und sich insofern von der Regelung in Ziff. 5 Abs. 2 GPL v3 zu unterscheiden scheint, in der auf die „spezielle" Benutzung Bezug genommen wird.
[666] Hintergrund dieser Änderung ist nicht zuletzt in der „Umgehung" der GPL v2 zu sehen, welche die Firma Tivo dadurch ermöglicht hat, dass Sie zwar die Quellen der Open Source Programme entsprechend den Voraussetzungen offen gelegt hat, modifizierte Versionen des Programms sich jedoch nicht auf der Tivo vertriebenen Hardware ausführen ließen, weil hierfür eine spezielle Signatur erforderlich gewesen wäre, die jedoch nicht veröffentlicht wurde; vgl. *Diedrich*, Streit um die neue GPL, http://www.heise.de/open/artikel/78967.
[667] *Wiebe / Heidinger*, MR 2006, 258, 260; ebenso Torvalds, http://www.heise.de/open/news/meldung/76110.

in Ziff. 9 GPL v2 enthaltene Regelung,[668] nach der die der GPL v2 unterstellten Programme zugleich auch jeder späteren Version unterstehen, erhält künftig der Lizenznehmer die Möglichkeit auszuwählen, ob er die GPL unter der Version 2 oder 3 abschließen will. Im Hinblick darauf, dass die Annahme der GPL häufig erst in der Vornahme einer zustimmungsbedürftigen Handlung durch den Nutzer liegt und mithin in konkludenter Form erfolgt,[669] wird es sich letztlich nicht vermeiden lassen, dass der Nutzer sich faktisch erst innerhalb des Prozesses dazu äußert, welcher Version er sich verbunden fühlt.[670] Dies führt dazu, dass sich die neu aufgenommenen Patent- und DRM-Regelungen wohl nicht hinsichtlich solcher Programme durchsetzen lassen, bei denen dem Nutzer die Wahl zwischen der Version 2 und 3 zusteht. Die hierdurch zu befürchtenden Nachteile scheinen jedoch aufgrund der, im Wesentlichen gleichwertigen Ausprägung des Copyleft-Effekts, begrenzt zu sein, was sich aber schon durch eine von der bisherigen Auffassung zur GPL v2 abweichende Entscheidung aus der Rechtsprechung ändern könnte.

Daneben ist zu berücksichtigen, dass der Entwurf der GPL v3 auch in der Entwicklergemeinde nicht unumstritten ist und sich auch IT-Unternehmen im Hinblick auf die darin enthaltenen DRM und Patentregelungen künftig vor einer Veröffentlichung von unternehmensinternen Entwicklungen eingehend mit der Frage beschäftigen werden, unter welcher Lizenzversion sie den Quellcode anbieten. Ausdrücklich gegen die GPL v3 hat sich bereits die Mehrzahl der Entwickler des Linux-Kernels ausgesprochen,[671] weshalb wesentliche Teile von GNU/Linux auch künftig ausschließlich unter der GPL v2 zu erhalten sein werden. Diesem Beispiel sind sowohl MySQL[672] als auch Sun[673] gefolgt und haben für ihre Entwicklungen Ziff. 9 GPL v2 dahin gehend abgeändert, dass die darin enthaltene *„any later version“* - Klausel gegen einen ausdrücklichen Bezug auf GPL v2 ersetzt wurde.

Zusammenfassend ist daher festzuhalten, dass die GPL v2 nicht unmittelbar nach der Veröffentlichung der GPL v3 an Bedeutung verlieren wird. Vielmehr wird ihre Relevanz wohl erst in einigen Jahren dadurch spürbar abnehmen, dass zunehmend neue Open Source Projekte gegründet werden, die ihre Entwicklungen ausschließlich unter der GPL v3 anbieten.

---

[668] Ziff. 9 Abs. 2 GPL v2: *“...which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation.”*.
[669] Vgl. Kapitel 5 S. 98 ff.
[670] Es dürfte eine Ausnahme darstellen, dass der Nutzer ein eindeutiges konkludentes Verhalten vornimmt, das ausschließlich auf die Annahme einer bestimmten Version schließen lässt.
[671] Vgl. die Umfrageergebnisse unter den Kernelentwicklern http://www.uwsg.indiana.edu/hypermail/linux/kernel/0609.2/1882.html.
[672] Vgl. http://www.golem.de/0612/49629.html
[673] Vgl. http://www.itreseller.ch/news/nw_single.cfm?NW_ID=26952.

Im Hinblick auf die bereits derzeit bestehenden größeren Open Source Projekte, wird hingegen oftmals die Möglichkeit fehlen, für diese einen Lizenzwechsel vorzunehmen, da hierfür die Zustimmung sämtlicher Miturheber erforderlich wäre, welche, nicht zuletzt aufgrund der durch die kollaborativen Entwicklungsstrukturen gegebenen großen Anzahl an Rechtsinhabern und deren lockeren Bündnisse zueinander, nur in Ausnahmen zu erreichen sein dürfte.[674]

[674] Ähnlich in der Einschätzung *Kreutzer* in: Ifross, Die GPL kommentiert und erklärt, Ziff. 9 Rn. 20; von einer praktischen Undurchführbarkeit gehen *Wiebe / Heidinger*, MR 2006, 258, 261, aus.

Anhang

GNU General Public License

Version 2, June 1991

This page mirrors the text from http://www.gnu.org/licenses/gpl.html.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The “Program”, below, refers to any such program or work, and a “work based on the Program” means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term “modification”.) Each licensee is addressed as “you”.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

1. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
2. 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. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. 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.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

1. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
2. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
3. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place

counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
[one line to give the program's name and an idea of what it does.]
Copyright (C) [yyyy]  [name of author]
```

```
This program is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the Free
Software Foundation; either version 2 of the License, or (at your option)
any later version.

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.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc., 51
Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) [year] [name of author]
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under certain
conditions; type 'show c' for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

[signature of Ty Coon], 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit

linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

* * *

Copyright notice above.

51 Franklin Street, Fifth Floor, Boston, MA 02110, USA

Verbatim copying and distribution of this entire article is permitted in any medium without royalty provided this notice is preserved.

GNU General Public License v3

Discussion Draft 2 of Version 3, 27 July 2006

This page mirrors the text from http://gplv3.fsf.org/gpl-draft-2006-07-27.html.

THIS IS A DRAFT, NOT A PUBLISHED VERSION OF THE GNU GENERAL PUBLIC LICENSE.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other program whose authors commit to using it. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to make requirements that forbid anyone to deny you these rights or to ask you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be associated erroneously with the original version.

Some computers are designed to deny users access to install or run modified versions of the software inside them. This is fundamentally incompatible with the purpose of the GPL, which is to protect users' freedom to change the software. Therefore, the GPL ensures that the software it covers will not be restricted in this way.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in places where they do, we wish to avoid the special danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

TERMS AND CONDITIONS

0. Definitions.

In this License, each licensee is addressed as “you,” while “the Program” refers to any work of authorship licensed under this License. A “modified” work includes, without limitation, versions in which material has been translated or added. A work “based on” another work means any modified version, formation of which requires permission under applicable copyright law. A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means doing anything with it that requires permission under applicable copyright law, except executing it on a computer, or making modifications that you do not share. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well. To “convey” a work means any kind of propagation that enables other parties to make or receive copies, excluding sublicensing.

A party's “essential patent claims” in a work are all patent claims that the party can give permission to practice, whether already acquired or to be acquired, that would be infringed by making, using, or selling the work.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source version of a work.

The “System Libraries” of an executable work include every subunit such that (a) the identical subunit is normally included as an adjunct in the distribution of either a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the object code runs, or a compiler used to produce the object code, or an object code interpreter used to run it, and (b) the subunit (aside from possible incidental extensions) serves only to enable use of the work with that system component or compiler or interpreter, or to implement a widely used or standard interface for which an implementation is available to the public in source code form.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, except its System Libraries, and except general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes scripts used to control those activities, interface definition files associated with the program source files, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by complex data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source also includes any encryption or authorization keys necessary to install and/or execute modified versions from source code in the recommended or principal context of use, such that they can implement all the same functionality in the same range of circumstances. (For instance, if the work is a DVD player and can play certain DVDs, it must be possible for modified versions to play those DVDs. If the work communicates with an online service, it must be possible for modified versions to communicate with the same online service in the same way such that the service cannot distinguish.) A key need not be included in cases where use of the work normally implies the user already has the key and can read and

copy it, as in privacy applications where users generate their own keys. However, the fact that a key is generated based on the object code of the work or is present in hardware that limits its use does not alter the requirement to include it in the Corresponding Source.

The Corresponding Source may include portions which do not formally state this License as their license, but qualify under section 7 for inclusion in a work under this License.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running it is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of “fair use” or other equivalent, as provided by copyright law.

This License permits you to make and run privately modified versions of the Program, or have others make and run them on your behalf. However, this permission terminates, as to all such versions, if you bring suit against anyone for patent infringement of any of your essential patent claims in any such version, for making, using, selling or otherwise conveying a work based on the Program in compliance with this License.

Propagation of covered works other than conveying is permitted without limitation. Sublicensing is not allowed; section 10 makes it unnecessary. Conveying is permitted under the conditions stated below.

3. No Denying Users' Rights through Technical Measures.

Regardless of any other provision of this License, no permission is given for modes of conveying that deny users that run covered works the full exercise of the legal rights granted by this License.

No covered work constitutes part of an effective technological “protection” measure under section 1201 of Title 17 of the United States Code. When you convey a covered work, you

waive any legal power to forbid circumvention of technical measures that include use of the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing the legal rights of third parties against the work's users.

4.[1.] Verbatim Copying.

You may copy and convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all license notices and notices of the absence of any warranty; and give all recipients, along with the Program, a copy of this License and the central list (if any) required by section 7. The recipients of these copies will possess all the rights granted by this License (with any added terms under section 7).

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.

5.[2.] Conveying Modified Source Versions.

You may copy and convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4 above, provided that you also meet all of these conditions:

a) The modified work must carry prominent notices stating that you changed the work and the date of any change.

b) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License must apply, unmodified except as permitted by section 7 below, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

c) If the modified work has interactive user interfaces, each must include a convenient feature that displays an appropriate copyright notice, and tells the user that there is no warranty for the program (or that you provide a warranty), that users may convey the modified work

under this License, and how to view a copy of this License together with the central list (if any) of other terms in accord with section 7. Specifically, if the interface presents a list of user commands or options, such as a menu, a command to display this information must be prominent in the list; otherwise, the modified work must display this information at startup. However, if the Program has interactive interfaces that do not comply with this subsection, your modified work need not make them comply.

To the extent that identifiable sections of the modified work, added by you, are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you convey them as separate works, not specifically for use in combination with the Program.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6.[3.] Conveying Non-Source Forms.

You may copy and convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give any third party a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of

source.

[b1) Convey the object code in a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to provide access to copy the Corresponding Source from a network server at no charge.]

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b or 6b1.

d) Convey the object code by offering access from a designated place, and offer equivalent access to the Corresponding Source in the same way through the same place at no extra charge. You need not require recipients to copy the Corresponding Source along with the object code.

[If the place to copy the object code is a network server, the Corresponding Source may be on a different server that supports equivalent copying facilities, provided you have explicitly arranged with the operator of that server to keep the Corresponding Source available for as long as needed to satisfy these requirements, and provided you maintain clear directions next to the object code saying where to find the Corresponding Source.]

e) Convey the object code using peer-to-peer transmission provided you know that, and inform other peers where, the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

The Corresponding Source conveyed in accord with this section must be in a format that is publicly documented, with an implementation available to the public in source code form, and must require no special password or key for unpacking, reading or copying.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

7. Additional Terms.

You may have received the Program, or parts of it, under terms that supplement the terms of this License. These additional terms may include additional permissions, as provided in subsection 7a, and additional requirements, as provided in subsection 7b. When you convey copies of a covered work, unless the work also permits use under a previous version of this License, it must list, in one central place in the source code, the complete set of additional terms governing all or part of the work.

1. Additional Permissions.

   Additional permissions make exceptions from one or more of the requirements of this License. A license document containing a clause that permits relicensing or conveying under this License shall be treated as a list of additional permissions, provided that the license document makes clear that no requirement in it survives such relicensing or conveying.

   Any additional permissions that are applicable to the entire Program are treated as though they were included in this License, as exceptions to its conditions, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional terms.

2. Additional Requirements.

   Additional requirements are terms that further constrain use, modification or propagation of covered works. This License affects only the procedure for enforcing additional requirements, and does not assert that they can be successfully enforced by the copyright holder. Only these kinds of additional requirements are allowed by this License:

   1. terms that require preservation of specified reasonable legal notices or author attributions; or
   2. terms that require that the origin of the material they cover not be misrepresented, or that modified versions of that material be marked in specific reasonable ways as different from the original version; or
   3. warranty or liability disclaimers that differ from the disclaimers in this License; or

4. terms that prohibit or limit the use for publicity purposes of specified names of licensors or authors, or that require that certain specified trade names, trademarks, or service marks not be used for publicity purposes without express permission, other than in ways that are fair use under applicable trademark law; or
5. terms that require, if a modified version of the material they cover is a work intended to interact with users through a computer network, that those users be able to obtain copies of the Corresponding Source of the work through the same network session; or
6. terms that wholly or partially terminate, or allow termination of, permission for use of the material they cover, for a user who files a software patent lawsuit (that is, a lawsuit alleging that some software infringes a patent) not filed in retaliation or defense against the earlier filing of another software patent lawsuit, or in which the allegedly infringing software includes some of the covered material, possibly in combination with other software; or
7. terms that are precisely equivalent in type and extent to a requirement expressly stated in this License, or that deny permission for activities that are clearly not permitted, expressly or otherwise, by this License.

All other additional requirements, including attorney's fees provisions, choice of law, forum, and venue clauses, arbitration clauses, mandatory contractual acceptance clauses, requirements regarding changes to the name of the work, and terms that require that conveyed copies be governed by a license other than this License, are prohibited.

3. Terms Added or Removed by You.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. Some additional permissions require their own removal in certain cases when you modify the work.

Additional requirements are allowed only as stated in subsection 7b. If the Program as you received it purports to impose any other additional requirement, you may remove that requirement.

You may place additional permissions, or additional requirements as allowed by subsection 7b, on material, added by you to a covered work, for which you have or can give appropriate copyright permission. Adding requirements not allowed by subsection 7b is a violation of this License that may lead to termination of your rights under section 8.

If you add terms to a covered work in accordance with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

8.[4.] Termination.

You may not propagate or modify the Program except as expressly provided under this License. Any attempt otherwise to propagate or modify the Program is void. If you violate this License, any copyright holder may put you on notice by notifying you of the violation, by any reasonable means, provided 60 days have not elapsed since the last violation. Having put you on notice, the copyright holder may then terminate your license at any time. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as they remain in full compliance.

9.[5.] Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing else grants you permission to propagate or modify the Program or any covered works. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating the Program (or any covered work), you indicate your acceptance of this License to do so, and all its terms and conditions.

10.[6.] Automatic Licensing of Downstream Users.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License, including any additional terms introduced through section 7. You may not impose any further restrictions on the recipients' exercise of the rights thus granted or affirmed, except in the

limited ways permitted by section 7. Therefore, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License. You are not responsible for enforcing compliance by third parties to this License.

If propagation results from a transaction transferring control of an organization, each party to that transaction who receives a copy of the work also receives a license and a right to possession of the Corresponding Source of the work from the party's predecessor in interest.

11. Patents.

You receive the Program with a covenant from each author and conveyor of the Program, and of any material, conveyed under this License, on which the Program is based, that the covenanting party will not assert (or cause others to assert) any of the party's essential patent claims in the material that the party conveyed, against you, arising from your exercise of rights under this License. If you convey a covered work, you similarly covenant to all recipients, including recipients of works based on the covered work, not to assert any of your essential patent claims in the covered work.

If you convey a covered work, knowingly relying on a non-sublicensable patent license that is not generally available to all, you must either (1) act to shield downstream users against the possible patent infringement claims from which your license protects you, or (2) ensure that anyone can copy the Corresponding Source of the covered work, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12.[7.] No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey the Program, or other covered work, so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you accept a patent license that

prohibits royalty-free conveying by those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from conveying the Program.

13.[8.] Geographical Limitations.

If the conveying and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical limitation on conveying, excluding those countries, so that conveying is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.]

14.[9.] Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

15.[10.] Requesting Exceptions.

If you wish to incorporate parts of the Program into other free programs under other licenses, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.]

NO WARRANTY

16.[11.] Disclaimer of Warranty.

There is no warranty for the Program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the Program “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the Program is with you. Should the Program prove defective, you assume the cost of all necessary servicing, repair or correction.

17.[12.] Limitation of Liability.

In no event unless required by applicable law or agreed to in writing will any copyright holder, or any other party who may modify and/or convey the Program as permitted above, be liable to you for damages, including any general, special, incidental or consequential damages arising out of the use or inability to use the Program (including but not limited to loss of data or data being rendered inaccurate or losses sustained by you or third parties or a failure of the Program to operate with any other programs), even if such holder or other party has been advised of the possibility of such damages.

END OF TERMS AND CONDITIONS

* * *

Universität Karlsruhe (TH)
Institut für Informationsrecht

Die Open-Source-Wirtschaft führt schon längst kein Schattendasein mehr. Software, die unter Open Source Lizenzen veröffentlicht wurde, kommt in immer mehr Anwendungsfeldern zum Einsatz und stellt immer häufiger den wirtschaftlichen Kernbereich von Unternehmen dar. Neben den gestiegenen Einsatzbereichen für Open Source Software ist seit geraumer Zeit jedoch auch der Trend zu erkennen, die Einhaltung der im Open Source Bereich bestehenden Lizenzbestimmungen – allen voran der General Public License (GPL) – notfalls auch gerichtlich durchzusetzen. Insofern müssen sämtliche Unternehmen, die in ihrer Geschäftspraxis mit der Open Source Branche in Berührung kommen, sich künftig – mehr als zuvor – um die Einhaltung der maßgebenden Lizenzen sorgen.

Die vorliegende Arbeit nimmt diese Ausgangslage zum Anlass, die weitgehend umstrittene Reichweite der GPL eingehend zu beleuchten und Wege aufzuzeigen, wie es Unternehmen – trotz teilweise bestehender rechtlichen Unsicherheiten – möglich ist in diesem Umfeld tätig zu sein. Dabei werden sowohl die entscheidungsrelevanten rechtlichen wie auch die technischen Kriterien dargestellt und bewertet, um so die Entscheidungsträger zu der unternehmensinternen Strategieentscheidung zu befähigen.

ISSN: 1860-8744
ISBN: 978-3-86644-141-5

www.uvka.de