# Niels Taubert
# Autorschaft von Open-Source-Software

## Zur Aktualität eines alten Konzepts[1]

Als Beobachter der derzeitigen Entwicklung der Informations- und Kommunikationsmedien könnte man schnell auf die Idee kommen, Autorschaft sei ein überholtes, ja anrührend antiquiertes Konzept, das aus den guten alten Zeiten der gedruckten Publikation stammt. Die sich verfestigenden Praktiken des »copy, shake and paste«, »remix« und »sampling«, des Auseinanderbrechens, Neuarrangierens und Wiederzusammenfügens von *content* aus der Menge digitaler Daten und Objekte, scheinen nicht kompatibel zu sein mit der Figur des Autors – eine Person, die ein monolithisch-festgefügtes Textwerk produziert, dessen Inhalt verantwortet und der im Gegenzug für ihren schöpferischen Akt die Urheberschaft mit den entsprechenden Rechten zukommt. Auf den ersten Blick scheint es, als hätten die digitalen Medien all das verflüssigt: Die Digitalität eines Werks nimmt diesem seine Abgeschlossenheit und an diese Stelle tritt chronische Unfertigkeit. Als digitales Objekt bildet es eine Vorstufe für weitere Be- und Verarbeitungsschritte, ohne dass grundsätzlich absehbar wäre, wann dieser Prozess zu einem Ende kommt. Zudem findet eine Entgrenzung des Kreises an Personen statt, die sich am Umbau des Werks beteiligen, es in neue Kontexte rücken oder es mit anderen Werken fusionieren. Um im Bild der Gegenüberstellung von gedruckten und digitalen Medien zu bleiben: Aus dem einsamen Poeten, der allein in seinem Kämmerlein ein Buch verfasst, ist der wuselnde Wissensarbeiter geworden. Hyperaktiv als Gruppe in der *cloud* und in der Tendenz sprunghaft geht er in abgegrenzten Zeiträumen seinem jeweils aktuellen *Projekt* nach – wer mag da noch von Autorschaft, Verantwortung und Werkherrschaft sprechen?

All das ist natürlich Klischee. Bereits vom gedruckten Text ist das Phänomen der Mehrautorschaft bekannt, und in den Naturwissenschaften wird seit den wissenschaftlichen Revolutionen das Prinzip der Arbeitsteilung praktiziert, durch das ein gemeinsamer Bestand an Wissen produziert wird. Und sieht man sich die digitalen Medien und ihre Nutzung an, finden sich Indizien dafür, dass Autorschaft nicht verschwindet, sondern im Gegenteil ausufert: Auf Nachrichtenseiten wird auch im digitalen Format der Name des Urhebers genannt. In Blogs werden Beiträge ebenso mit einem Namen versehen wie die Postings, die darauf reagieren. Und selbst ein Dienst wie Twitter kennzeichnet die kurzen Mitteilungen mit Kürzeln, die auf eine Person verweisen, der dann ›gefolgt‹ werden kann.

[1] Ich danke den Herausgebern für konstruktive Hinweise und Kritik.

Der Aufsatz nimmt diese Beobachtung zum Ausgangspunkt und stellt die Frage nach der Bedeutung von Autorschaft an einem Beispiel, das einen Grenz- bzw. Sonderfall bildet: Autorschaft im Fall von Open-Source-Software. Sonderfall ist das Beispiel insofern, weil Open-Source-Softwareentwicklung in doppelter Weise an Digitalität gekoppelt ist. Die entwicklungsbezogene Kommunikation basiert überwiegend auf der Nutzung digitaler Medien, und das Produkt selbst – die Software – ist digital. Grenzfall ist sie, weil das Produkt in einer formalen Programmiersprache abgefasst ist und sich dieser Typus von Werk ganz offensichtlich von dem unterscheidet, was landläufig als ›Literatur‹ bezeichnet wird. Und Grenzfall ist es auch mit Blick auf den beteiligten Personenkreis, der mehrere Dutzend, in Einzelfällen sogar Hunderte Entwickler einschließen kann. Auf diesem Hintergrund stellt sich die Frage, in welchem Sinne hier von Autorschaft gesprochen werden kann und welche Rolle sie im Rahmen von Softwareentwicklungsprozessen spielt.

Der Aufsatz ist wie folgt organisiert: Zu Beginn wird der empirische Gegenstand, Open-Source-Software und Open-Source-Softwareentwicklung, vorgestellt und dessen konstitutive Merkmale werden benannt. In einem zweiten Schritt wird beschrieben, welche verschiedenen Formen der Nennung von Personennamen in diesem Bereich anzutreffen sind. Hieran schließt sich drittens eine knapp gehaltene Aufarbeitung zum Begriff ›Autor‹ in den Literaturwissenschaften, Rechtswissenschaften und der Wissenschaftsforschung an, die im vierten Schritt zur Beantwortung der Funktion von Autorschaft im Bereich von Open-Source-Software mündet. Es zeigt sich, dass sich in der Rolle des Autors rechtliche und motivationale Funktionen kreuzen und Autorschaft zudem mit normativen Erwartungen verbunden ist. Der abschließende fünfte Abschnitt fasst zusammen, welche Modifikationen am Begriff ›Autor‹ vorgenommen werden müssen, um ihn für eine Erklärung des Phänomens fruchtbar zu machen.

## 1. Open-Source-Software

Open-Source-Software ist ein Phänomen, das langsam in die Jahre kommt. Vor nunmehr 30 Jahren, oder genauer: am 27. September 1983, wurde das GNU-Projekt gegründet, das darauf zielt, ein für jedermann uneingeschränkt nutzbares Unix-Betriebssystem zu entwickeln (siehe zur Geschichte von Unix Stallmann 2002, S. 17; Raymond 1999, S. 13 ff.). Der ›Vater‹ der zu diesem Zeitpunkt als »free Software movement« bezeichneten Bewegung, Richard Stallmann, entwarf einige Zeit später eine spezielle Lizenz, die gemeinsam mit der Software vertrieben wird: die GNU General Public License (GPL), deren erste Version im Jahr 1989 aufgelegt wurde. Diese Lizenz räumt den Empfängern umfassende Rechte ein, zu denen im Einzelnen das Recht zur uneingeschränkten Nutzung, Vervielfältigung,

Verbreitung sowie zur Modifikation des Programms zählt.[2] Letztgenanntes Recht schließt die Veränderung, Weiterentwicklung oder das Zusammenfügen (›mergen‹) von verschiedenen Programmen ein. Damit auch praktisch vom Modifikationsrecht Gebrauch gemacht werden kann, wird das Programm nicht nur in der von Computern verarbeitbaren binären Form im Internet bereitgestellt, sondern auch als Programmquellcode. Hierbei handelt es sich um die in einer höheren Programmiersprache abgefasste Form des Programms, die für Menschen lesbar ist.

Zwei Konsequenzen der GNU GPL sollen im Vergleich zu gewöhnlichen proprietären Lizenzen hervorgehoben werden: Im Fall der proprietären Software verfügt typischerweise ausschließlich der Hersteller über das Recht zur Veränderung des Programms, der Quellcode wird als Betriebsgeheimnis behandelt. Durch Open-Source-Softwarelizenzen erhält dagegen jedermann das Recht zur Modifikation der Software. Die erste Konsequenz besteht also darin, dass in der Sozialdimension der Kreis von Personen radikal ausgeweitet wird, der sich an der Entwicklung beteiligen kann. Daneben beinhaltet die Lizenz eine sogenannte Fortgeltungsklausel, die erzwingt, dass für sämtliche Weiterentwicklungen dieselben Lizenzbedingungen zu gelten haben wie für die ursprüngliche Version. Einmal durch die GNU GPL geschützte Software bleibt in sämtlichen zukünftigen Versionen dauerhaft Open-Source-Software. Mit Blick auf die Weiterentwicklung ist daher zu sagen, dass die Ausweitung des potenziell beteiligten Personenkreises auf jedermann in der Zeitdimension dauerhaft stabilisiert wird. Die zweite Konsequenz der Lizenz bezieht sich auf die Möglichkeiten einer Kommodifizierung. Zwar schließen Open-Source-Softwarelizenzen in der Regel nicht den Verkauf von Software aus. Praktisch führt aber die großzügige Einräumung von Rechten an jedermann dazu, dass kaum Knappheit entstehen kann. Dies gilt umso mehr, da die Vervielfältigung und der Vertrieb von Computerprogrammen auf dem Weg des Downloads aus dem Internet zu sehr geringen Kosten möglich sind. Daher stellt Open-Source-Software faktisch ein öffentliches Gut dar (vgl. Brand/Holtgrewe 2005; Tuomi 2005, S. 443), das stets in ausreichendem Maße verfügbar ist.[3]

Open-Source-Software wird auf sehr verschiedene Art und Weise entwickelt. Die Entwicklung kann von einer einzelnen Person vorangetrieben werden, die sich dazu entscheidet, das Werk unter eine entsprechende Lizenz zu stellen. Daneben gibt es Projekte von Softwarehäusern, die Open-

[2] Zu den rechtlichen Aspekten von Open-Source-Softwarelizenzen siehe ausführlicher Lerner/Tirole 2002; Bonaccorsi/Rossi 2003a, S. 9, 2003b, S. 1248; O'Mahony 2003; Taubert 2006, S. 25 ff.

[3] Geschäftsmodelle basieren daher auch nicht auf dem Verkauf von Software, sondern primär auf Dienstleistungen, die mit der Software verknüpft sind (z. B. Auswahl und Zusammenstellung von Softwarepaketen, Anfertigung einer gedruckten Dokumentationen usw.; vgl. Raymond 1999, S. 157 ff.).

Source-Software analog zu konventionellen Programmen produzieren, und es finden sich Projekte, die von der öffentlichen Hand finanziert werden. Eine in diesem Feld wichtige Organisationsform nutzt das durch die Lizenzen bereitgestellte Potenzial der Ausweitung des Modifikationsrechts in besonders effektiver Weise. Gemeint sind sogenannte Community-getriebene Softwareentwicklungsprojekte, die an der Entwicklung interessierte Personen an einem Ort zusammenbringen. Entstanden ist diese Organisationsform im Rahmen des Linux-Projekts zu Beginn der 1990er Jahre. Die Innovation bestand darin, mit der damals leitenden Vorstellung einer weitgehenden Formalisierung des Entwicklungsprozesses zu brechen und an die Stelle einer strikten sachlichen, zeitlichen und sozialen Organisation des Prozesses eine in kurzen Zyklen strukturierte, inkrementelle Entwicklung zu setzen, durch die der Entwicklungs- und Anwendungskontext iterativ rückgekoppelt wird.[4]

Welches sind nun die Merkmale von Community-getriebenen Projekten? Ein erstes wichtiges Element bildet die *intensive Nutzung von Informations- und Kommunikationstechnologien*. Die entwicklungsbezogene Kommunikation ist vorrangig computervermittelt, und dies in mehrerlei Hinsicht: Die Entwickler nutzen zur Diskussion über Zielsetzungen und zum Austausch von Information eine Mailingliste. Daneben gibt es meist eine Webseite, die das Projekt beschreibt und auf der sich die aktuelle Version des Programms findet. Eingesetzt werden daneben sogenannte Versionsverwaltungsprogramme,[5] die die Dokumente des Projekts wie Programmdateien und Dokumentationen bereithalten. Zusätzlich verfügen die Projekte häufig über sogenannte Bugtrack-Systeme, die zur Sammlung und Verwaltung von Fehlermeldungen der Nutzer dienen. Und schließlich findet sich häufig noch eine sogenannte *wishlist*, auf der die Anwender des Programms Wünsche formulieren können, welche Funktionen sie gern implementiert sehen würden.

Ein zweites Merkmal bildet die *Selbstselektion der Teilnehmerschaft*. Selbstselektion meint, dass die Entscheidung über die Art, den Umfang und die Dauer der Beteiligung bei den Teilnehmern liegt und nicht etwa beim Projekt. Daher verfügen die Entwicklungsprojekte im Unterschied zu formalen Organisationen nicht über die Möglichkeit, sich von der Umwelt

[4] Die Absetzung von den bis dato gültigen Grundsätzen der professionellen Softwareentwicklung zeigt sich am deutlichsten in der Frühzeit des Linux-Projekts (Taubert 2006, S. 98 ff.). Sie war zunächst ausschließlich negativ, indem die Vorstellung der Beherrschbarkeit der Entwicklung durch Planung abgelehnt wurde. In den folgenden Jahren blieb es freilich nicht dabei und es entstand ein Entwicklungsmodell, das den spezifischen Gegebenheiten Community-getriebener Projekte Rechnung trägt.

[5] Beispiele hierfür sind das mittlerweile veraltete Programm Concurrent Version Systems (CVS) und Git.

selbst abzugrenzen und diese Grenzziehung zu kontrollieren. Aufgrund des Fehlens von klaren Mitgliedschaftsregeln, die vom Projekt gehandhabt werden, entsteht ein gewisses Maß an Diffusität, wer sich an den Rändern selbst noch als Teil des Projekts versteht und wer nicht mehr oder noch nicht.

Das dritte Charakteristikum bildet eine *schwache Rollendifferenzierung*. Die wenigen Differenzen, die auszumachen sind (vgl. hierzu Gacek/Arief 2004, S. 36; Crowston et al. 2006, S. 1; Gläser 2006, S. 270),[6] orientieren sich an der Art der Beiträge zum Projekt und an unterschiedlichen Kompetenzen, die selektiv auf die Art der Beteiligung wirken. Im Zentrum des Projekts steht ein sogenannter Projekteigentümer (›owner‹), der entweder das Projekt gegründet hat oder vom Gründer als Nachfolger benannt wurde. Diese Rolle ist in vielen Projekten die einzige, die auf formalen Kriterien basiert. Zu nennen ist hier das Recht, im Namen des Projekts neue Programmversionen (›release‹) zu veröffentlichen, sowie die Berechtigung, Befugnisse in Bezug auf die eingesetzte Informations- und Kommunikationsinfrastruktur zu vergeben.[7] Zweitens findet sich die Rolle des ›core developer‹, die über einen längeren Zeitraum definiert wird, während dessen der Entwickler Beiträge zum Projekt leitet. Neben dieser Rolle und der des sporadisch beteiligten Programmierers findet sich noch die des Nutzers. Dabei handelt es sich um einen Kreis von Personen, die die aktuelle (Entwickler)Version des Programms verwenden und diese hinsichtlich ihrer Alltagstauglichkeit und Stabilität testen. Mit Blick auf diese Rollen muss gesagt werden, dass es sich um Idealtypen handelt. Der Umfang der Aktivität variiert stark und bewegt sich auf einem Kontinuum.

Diese Bemerkung führt zu einem vierten Charakteristikum, der *hohen Fluktuation der Beteiligten*. Im Unterschied zu formalen Organisationen sind Community-getriebene Projekte in einem sehr begrenzten Maße in der Lage, Mitglieder durch monetäre Ressourcen zu binden. Die Abwesenheit solcher Mittel führt gemeinsam mit der Selbstselektion der Teilnehmerschaft zu einer fortlaufenden Veränderung der Zusammensetzung des Projekts. Wenngleich sich der Kern mit dem *owner* und den *core developers* in der Zeitdimension meist als recht stabil erweist, lässt sich an den Rändern ein reges Kommen und Gehen beobachten: User berichten über einen mehr oder weniger langen Zeitraum über unvorhergesehene Programmreaktionen und beenden danach ihr Engagement; Entwickler programmieren eine einzelne Funktion und verschwinden danach wieder aus

[6] Mithilfe von Aktivitätsmessungen ist bereits früh erkannt worden, dass sich die Projekte aus einem kleinen Kreis Hochaktiver und einer großen Zahl sporadisch Beteiligter zusammensetzen (vgl. Koch/Schneider 2002, S. 30f.).

[7] Bereits das Management dieser Rechte (wie die Vergabe von Schreibrechten im Versionsverwaltungsprogramm) kann an einen anderen Entwickler delegiert worden sein.

dem Projekt. Fluktuation bezieht sich aber auch auf die eben beschriebene Rollendifferenzierung. Regelmäßig anzutreffen ist, dass Beteiligte zunächst in der Rolle des Nutzers ihr Engagement mit dem Bericht von Fehlern beginnen, und dann, nach allmählichem Erwerb von Programmierkompetenzen, in der Lage sind, Probleme selbständig einzugrenzen oder sogar einen ›patch‹ zu programmieren.

## 2. Namensnennung in Open-Source-Softwareentwicklungsprojekten

Nach diesen einführenden Bemerkungen zum Gegenstand soll nun der Frage nachgegangen werden, ob es im Bereich der Open-Source-Softwareentwicklung ein Phänomen gibt, das sich als Autorschaft bezeichnen lässt. Und sofern diese Frage bejaht wird: Welche Erscheinungsformen nimmt diese an? Der folgende Schritt dient der Annäherung an das Phänomen, indem zunächst untersucht wird, zu welchen Gelegenheiten und an welcher Stelle die Nennung von Personennamen anzutreffen ist. Dieses stark deskriptiv geprägte Vorgehen dient dazu, die Breite des Phänomens auszuleuchten, und bildet die Vorarbeit, um im anschließenden Abschnitt einen angemessenen Begriff von ›Autor‹ zu entwickeln.

Die hier dargestellten Ergebnisse basieren auf einem thematisch umfangreicheren Projekt, in dem Strukturgenese und Entscheidungsprozesse von Open-Source-Softwareentwicklungsprojekten untersucht wurden.[8] Das dem Projekt zugrunde liegende empirische Material: die Kommunikation einer Entwickler-Mailingliste, 12 qualitative Interviews mit Open-Source-Softwareentwicklern sowie Texte mit selbstbeschreibendem Charakter, wird hier verwendet, um der Frage nach der Funktion von Autorschaft nachzugehen. Fallbeispiel bildet ›KMail‹, ein recht typisches Community-getriebenes Projekt, das auf die Entwicklung eines E-Mail-Clients zielt (vgl. die Darstellung von KMail auf der Webseite sowie Taubert 2006, S. 123–126; 2008, S. 76 f.). Ein solches Programm dient zum Verfassen, Senden, Empfangen, Lesen und Verwalten persönlicher EMail-Korrespondenz, bekannte Beispiele sind Mozilla Thunderbird, Microsoft Outlook, Mac OSX Mail und Opera. Die Besonderheit von KMail besteht darin, dass es Teil des Personal-Information-Managers ›Kontact‹ ist, das sich darüber hinaus aus Adressbuch, Terminverwalter, Merkzettel-Verwaltung, Nachrichtensammler, Newsreader, Newsticker, Synchronisationstool und Wetteranzeige zusammensetzt.[9] Das Programm-

[8] Siehe zur methodischen Anlage der Untersuchung, zur Darstellung des Samplings, der Datenerhebung und Auswertung ausführlicher Taubert 2006, S. 120–123 und 2008, S. 75–77.

[9] Siehe hierzu die Webseite: http://userbase.kde.org/Kontact [Zugriff am 17.01.2014].

paket ist wiederum Teil der grafischen Benutzeroberfläche KDE, das mehrere solcher Pakete umfasst und eine Arbeitsumgebung mit Standardprogrammen für Unix-Betriebssysteme bereitstellt.[10] Diese zweistufige Integration spiegelt sich auch in der Projektstruktur wider: Die Programme sind aufeinander abgestimmt und die Entwicklung folgt einer gemeinsamen zeitlichen Planung, die für das Gesamtprojekt maßgeblich ist.[11]

Einen Anlaufpunkt für Interessierte bildet die *Projektwebseite* von KMail, hier trifft man auf die erste Form von Namensnennung. Mit Blick auf die dargebotenen Inhalte hat die Webseite einen breiten Adressatenkreis. Sie stellt das Programm von seiner praktischen Seite her dar, offeriert Informationen über Funktionalität und Verwendungsmöglichkeiten und verlinkt unterstützende Ressourcen. Daneben wird mit Screenshots illustriert, die Nutzung spezieller Funktionen wie die Verschlüsselung mittels »Pretty Good Privacy« wird beschrieben und es werden Hinweise auf Trainings- und Schulungsangebote gegeben. Die Informationen richten sich vorrangig an den Kreis der derzeitigen Nutzer und adressieren daneben Personen, die sich über das Programm und dessen Leistungsmerkmale informieren möchten. Noch vor dem Punkt »FAQ, Hints and Tips« findet sich der Menüpunkt »The Development Team« mit einem Link. Dieser verweist auf eine separate Webseite, auf der insgesamt 50 Personen mitsamt E-Mail-Adressen aufgelistet werden. Darunter folgt eine zweite Liste, überschrieben mit »Danksagung«, mit weiteren sieben Personen.

Diese Liste stellt zunächst einen Zusammenhang zwischen dem genannten Personenkreis und dem Programm her und verweist darauf, dass die Personen an der Entwicklung von KMail beteiligt waren. Auf drei weitere Aspekte soll hingewiesen werden: Erstens kann der Liste entnommen werden, dass die Entwicklung des Programms eine kollektive Leistung darstellt. Dies ergibt sich bereits aus der beachtlichen Zahl namentlich genannter Personen. Zweitens beschränkt sich die Webseite darauf, die Namen der Entwickler zu nennen, offeriert aber keine darüber hinausgehenden Informationen wie z. B. zur Rolle, die ein Entwickler für das Projekt spielt. Ebenso im Dunkeln bleibt drittens der jeweils erbrachte Leistungsumfang. Hier könnte man zwar aufgrund der nicht-alphabetischen Reihung der Personennamen auf die Idee kommen, es könnte ein Zusammenhang zwischen Leistungsumfang und Reihenfolge bestehen.[12]

[10] Eine Beschreibung der Benutzeroberfläche findet sich unter: http://de.kde.org/infos/wasistkde/ [Zugriff am 17.01.2014].

[11] Die Planung findet in Entwicklungszyklen statt, die durch ›release schedules‹ strukturiert werden. Mit ihnen werden Schritte vorgegeben, die im Zuge der Veröffentlichung neuer Versionen durchlaufen werden müssen; vgl. http://techbase.kde.org/Schedules/Release_Schedules_Guide [Zugriff am 17.01.2014].

[12] Eine solche Form der Unterscheidung des Leistungsumfangs ist in der Wissenschaft von ko- oder multiautorierten Publikationen bekannt.

Diese Überlegungen bleiben aber Spekulation, da nicht ausgeschlossen werden kann, dass Position und Reihenfolge auf einen anderen Faktor wie beispielsweise den Eintrittszeitpunkt in den Kreis der Entwickler zurückzuführen sind.

Eine zweite Form von Namensnennung findet sich im *KMail-Handbuch*, wobei es sich nicht um ein gedrucktes Buch handelt, sondern um eine elektronische Dokumentation, die auf einer Webseite hinterlegt ist.[13] Sieht man sich den Text an, wird deutlich, dass der Kreis der adressierten Personen Ähnlichkeiten zu dem der Webseite aufweist. Wenn die Voraussetzungen der Verwendung des Programms in Sinne »Erster Schritte« erklärt, die Funktionalität erläutert und Voreinstellungen dargestellt werden, wird deutlich, dass wiederum Anwender die Adressaten sind. Die Nennung von Namen nimmt hier die folgende Gestalt an:[14]

**KMail_1** (KMail_1 t-online.de)
**KMail_2** (KMail_2 mediaone.net)
**KMail_3** (KMail_3 kde.org)
**KMail_4** (KMail_4 KMail4.com)
Deutsche Übersetzung: KMail_5
Überarbeitung: KMail_6
Überarbeitung ab Version 1.3: KMail_7
Krypto-Übersetzung: KMail_8
Version 2.1.96 (KDE 4.7) (2011-07-15)
Copyright © 1999, 2000, 2001, 2002 KMail_2

Beispiel 1: Auszug aus dem KMail-Handbuch[15]

Im Vergleich mit dem erstgenannten Fall fällt auf, dass diese Liste nicht egalitär verfährt, sondern vier Personennamen typografisch durch Fettdruck hervorhebt. Die Nennung einer E-Mail-Adresse und das Fehlen spezifischer Leistungen, die den anderen Personen beigefügt sind, lassen die Erstgenannten als Hauptautoren erscheinen und gleichzeitig als Ansprechpartner fungieren, während sich die Letztgenannten in der Rolle von Mitarbeitern präsentieren.

Diese einleitende und sichtbare Form der Namensnennung gibt allerdings nur sehr unvollständig Auskunft über die an der Erstellung des Handbuchs beteiligten Personen, wie der letzte Abschnitt »Dokumentation« zeigt:[16]

[13] Vgl. http://docs.kde.org/stable/de/kdepim/kmail/index.html [Zugriff am 17.01.2014].

[14] Sämtliche Entwicklernamen und E-Mail-Adressen wurden anonymisiert. Dabei wurden etwaige typografische Hervorhebungen beibehalten. Entwickler aus dem KMail-Projekt sind dabei mit »KMail« gekennzeichnet und fortlaufend nummeriert (bspw. KMail_1), alle anderen Entwickler (»developer«) mit »DEV« bezeichnet und ebenfalls durch Nummerierung unterschieden (bspw. DEV_1).

[15] Quelle: http://docs.kde.org/stable/de/kdepim/kmail/ [Zugriff am 17.01.2014].

[16] Dieser Abschnitt findet sich unter: http://docs.kde.org/stable/de/kdepim/kmail/documentation.html [Zugriff am 17.01.2014].

Hier werden deutlich ausführlicher insgesamt zwölf Personen namentlich erwähnt, die an 13 unterschiedlichen Arbeitsschritten – wie z. B. »Kapitel zu OpenPGP«, »Aktualisierung auf KMail 1.7« oder »deutsche Übersetzung ab Version 1.3« – beteiligt waren. Daneben werden noch weitere vier Personen genannt, die für das Handbuch wichtige Informationen bereitgestellt haben.

Die wesentlichen Unterschiede zwischen diesen beiden Auflistungen von Namen bestehen in der Länge der Liste und in der Spezifikation der Leistungen. Hinzu kommt, dass für sämtliche in der Dokumentation genannte Personen E-Mail-Adressen angegeben werden. Neben diesen beiden sehr sichtbaren Fällen von Namensnennung sollen noch zwei weitere Typen vorgestellt werden, die weniger auffällig sind, da sie nicht in der Projektdarstellung auftauchen, sondern der unmittelbar projektbezogenen Kommunikation entstammen.

```
From:   KMail_9 <KMail_9 () gmx ! nt>
Date:   2001-05-20 11:45:51
[Download message RAW]

Moin KMail_1!

KMail_1 schrieb am Sonntag, den 20. Mai 2001:

> Mhh, but why is Arial then listed in Kmail`s font choser?
> Also in the preview Arial seems to work. All font look
> different (=correct) in the preview.

Ok, it might be that arial „black" is listed before „arial"
on your system.
Try the appended patch.

KMail_9

[„patch.kcharset" (text/plain)]

Index: kcharsets.cpp
===================================================================
RCS fiel: /home/kde/kdelibs/kdecore/kcharsets, v
retrieving revision 1.77
diff - u -2 -d -p -b -re.77 kcharsets.cpp
[…]
```

Beispiel 2: E-Mail-Antwort auf den Bericht eines Anwenders über Probleme bei der Darstellung[17]

Die erste Form findet sich auf der öffentlich zugänglichen *Entwickler-Mailingliste* von KMail.[18] Auch hier trifft man eine intensive Nennung von

[17] Zitiert aus dem Archiv der Mailingliste von KMail: http://lists.kde.org/?l=kmail-devel&m=99035928319054&w=2 [Zugriff am 17.01.2014].

[18] Das Archiv der Mailingliste findet sich unter: http://lists.kde.org/?l=kmail-devel&r=1&w=2 [Zugriff am 17.01.2014].

Namen an, allein schon aufgrund des verwendeten Mediums. Der E-Mail-Verkehr erfordert technisch die Spezifikation von genau einem Absender und mindestens einem oder mehreren Empfängern. Dieser Punkt ist hier allerdings nicht wesentlich. Entscheidend ist vielmehr, dass auf dieser Liste nicht nur entwicklungsbezogene Fragen diskutiert, fehlerhafte Programmreaktionen erörtert und Entscheidungen über den weiteren Projektverlauf getroffen werden. Daneben tauchen mit großer Regelmäßigkeit auch Beiträge zum Projekt selbst auf. Solche E-Mails kombinieren einen natürlichsprachigen Ausdruck mit kleineren Beiträgen zum Programm, verfasst in einer formalen Programmiersprache. Ein Beispiel hierfür bildet die oben stehende gekürzte E-Mail, der ein Bericht eines Anwenders über Probleme bei der Darstellung von Schrifttypen vorausgegangen war (siehe Abbildung 2).

Für die Typologie von Namensnennungen von Bedeutung ist, dass hier ein Beitrag zur Lösung eines Problems geleistet wird. Dabei erlauben die Öffentlichkeit der Kommunikation und die Nennung von Namen die Zurechnung von Leistungen gegenüber einer Person und die Beurteilung der Qualität eines Beitrags durch andere Entwickler.

Eine letzte Form von Namensnennung findet sich im primären Produkt, dem Programm selbst. Dies gilt zumindest für den in einer Programmiersprache verfassten *Quellcode des Programms*. Öffnet man eine zu KMail gehörende Datei im Quellcode, sieht man zunächst nicht etwa auf Laien kryptisch wirkende Befehle und Anweisungen, sondern einen Vorspann, der Informationen zur Datei bereithält:

```
/*
* This file is part of Kmail
* Copyright (c) 2009 KMail_8 <KMail_8@gmail.com>
* Based on KMMsgBase code by
* Copyright (c) 1996-1998 KMail_9 <Kmail_9@kde.org>
* This program is free software; you can redistribute 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.
[…]
*/
```

Beispiel 3: Auszug aus dem Quellcode von KMail[19]

Nach der Erwähnung der Zugehörigkeit der Datei zu KMail werden zwei Namen in einen rechtlichen Kontext gerückt. Es wird eine Person als Inhaber der Urheberrechte an der Datei angeführt und eine andere genannt, die über die Urheberrechte an einem Werk verfügt, auf dem die Datei basiert.

[19] Siehe exemplarisch: http://www.filewatcher.com/p/kdepim-4.8.1.tar.bz2.15814559/kdepim-4.8.1/kmail/codecmanager.cpp.html [Zugriff am 17.01.2014].

# 3. Autorschaft in Literaturwissenschaft, Rechtswissenschaft und Wissenschaftsforschung

Nachdem bislang phänomenorientiert und sehr provisorisch von Namensnennungen in der Open-Source-Softwareentwicklung gesprochen wurde, soll in diesem Abschnitt geklärt werden, was mit ›Autorschaft‹ gemeint ist. Damit soll ein Hintergrundverständnis geschaffen werden, das es gestattet, das zur Diskussion stehende Phänomen einzuordnen, und die Voraussetzung gelegt werden, um die Frage nach der Funktion von Autorschaft in diesen Projekten zu beantworten.

Zuständig für Autorschaft fühlen sich gleich mehrere Disziplinen und Forschungsfelder. Daher ist es wenig überraschend, dass es keinen einheitlichen Begriff von ›Autor‹ gibt. Mit Blick auf den interessierenden Gegenstand ist es erhellend, sich mit drei Konzepten zu beschäftigen: mit dem literaturwissenschaftlichen Begriff, der den literarischen Autor meint, mit dem Begriff in der Wissenschaftsforschung, der sich auf den schreibenden und publizierenden Wissenschaftler bezieht, sowie mit der begrifflich breiter gefassten Figur des Urhebers in der Rechtswissenschaft. Im Folgenden werden die drei Begriffskonzepte kurz einander gegenübergestellt.

## *3.1 Autorschaft in den Literaturwissenschaften*

In der Literaturwissenschaft wird der Autor vor allem in seinem Verhältnis zum Werk und zum Leser diskutiert. Die leitende Frage ist dabei die nach dem Textverständnis: Wie viel Kontextwissen über die Eigenschaften und die Biografie des Autors muss einbezogen werden, um einen Text interpretieren (Jannidis et al. 2000, S. 11), respektive den Sinn entschlüsseln zu können? Über lange Zeit hinweg galt der Autor schlicht als derjenige, der ein Werk hervorgebracht hat. Diesem Verständnis nach ist der Autor charakterisiert durch die ungewöhnliche oder auch außeralltägliche Fähigkeit, Werke zu schaffen, die von hoher Bedeutung sind. Entsprechend exklusiv wird das Attribut Autor verwendet. Der enge Zusammenhang von Autor und Werk und die zentrale Bedeutung des Autors haben Konsequenzen mit Blick auf die Frage, wie Verständnis des in einen Text hineingelegten Sinns erlangt werden kann. Sicheres Verständnis setzt voraus, ein Werk auf dem Hintergrund des Wissens über den Autor, seiner Lebensumstände und seiner Intentionen zu interpretieren.

Dabei ist es jedoch nicht geblieben: Ausgehend von zwei Beiträgen zur Debatte hat eine deutliche Dezentrierung des Autors stattgefunden: Dies waren Roland Barthes' 1967 veröffentlichter Essay *La mort de l'auteur*[20]

[20] Ich beziehe mich hier auf die deutsche Übersetzung des Textes in Barthes 2000.

und Michel Foucaults 1969 am Collège de France gehaltener Vortrag *Qu'est-ce qu'un auteur?*[21]. Barthes wendet sich in seinem Aufsatz gegen das eingangs skizzierte Verständnis und die aus seiner Sicht stattfindende Überhöhung des Autors. Um den Autor von seinem Sockel zu stoßen, führt er zwei Argumente ins Feld: Erstens stellt er mit Blick auf die *Produktion von Texten* fest, die Leistung des Autors werde in der Gegenwart dramatisch überhöht.[22]

> »Heute wissen wir, dass ein Text nicht aus einer Reihe von Wörtern besteht, die einen einzigen, irgendwie theologischen Sinn enthüllt (welcher die ›Botschaft‹ des *Autor*-Gottes wäre), sondern aus einem vieldimensionalen Raum, in dem sich verschiedene Schreibweisen [ècritures], von denen keine einzige originell ist, vereinigen und bekämpfen. Der Text ist ein Gewebe von Zeichen aus unzähligen Stätten der Kultur. Wie die ewigen, ebenso erhabenen wie komischen Abschreiber Bouvard und Pécuchet, deren abgrundtiefe Lächerlichkeit genau die Wahrheit der Schrift bezeichnet, kann der Schreiber nur eine immer schon geschehene, niemals originelle Geste nachahmen. Seine einzige Macht besteht darin, die Schriften zu vermischen und sie miteinander zu konfrontieren.« (Barthes 2000, S. 190)

Das zweite Argument bezieht sich auf den Leser. Setzt man die eben geschilderte Art und Weise der Textproduktion voraus, gilt für die Rezeption, dass es nicht um die Entschlüsselung eines im Text liegenden Sinns gehen kann, sondern lediglich darum, die Sinnbezüge zu ordnen und diesen zu folgen.

> »Die vielfältige Schrift kann nämlich nur *entwirrt*, nicht *entziffert* werden. Die Struktur kann zwar in allen ihren Wiederholungen und auf allen ihren Ebenen nachvollzogen werden (so wie man eine Laufmasche ›verfolgen‹ kann), aber ohne Anfang und ohne Ende. Der Raum der Schrift kann nur durchwandert, aber nicht durchstoßen werden. Die Schrift bildet unentwegt Sinn, aber nur, um ihn wieder aufzulösen.« (ebd., S. 191)

Mit seinem Essay geht es Barthes nicht nur darum, den Leser aufzuwerten, mehr noch: Er zielt auf eine Veränderung der Rezeptionspraxis. Es solle

[21] Im Folgenden beziehe ich mich auf die deutsche Übersetzung des Vortrags Was ist ein Autor in Foucault 2003.

[22] Barthes vermengt hier ein systematisches und ein historisches Argument. Zu Beginn seines Essays spricht er davon, das Dazwischentreten von Schrift zerstöre bereits den Autor und dessen Intentionen (Barthes 2000, S. 185), um dann direkt anschließend zu argumentieren, mit dem Auftreten des scripteur habe sich die moderne Schreibpraxis verändert (ebd., S. 188). Diese Unklarheit fällt allerdings nicht weiter ins Gewicht, da es hier um die Rekonstruktion der Figur des modernen Autors bei Barthes geht.

darauf verzichtet werden, im Text nach einem Autor zu suchen, dem die Leistung der Schöpfung eines Textes zugeschrieben wird, was zu einer endgültigen Fixierung des Sinns eines Textes führe (ebd., S. 191). Zudem will er Kritik üben und eine überholte und unzutreffende Vorstellung über das Verhältnis von Autor, Werk und Text durch eine angemessenere ersetzen.

Ähnlich kritisch zum traditionellen Verständnis des Autors positioniert sich Foucault. Seine Zielrichtung unterscheidet sich aber von der vorangegangenen, da er sich nicht für das Textverständnis interessiert, sondern für die Existenzweisen von Diskursen und die Rolle des Autors darin. Es verbindet ihn allerdings mit Barthes, dass er den Autor nicht einfach als den Urheber eines Werks begreift. Für beide gilt, dass der Autor seinen Ursprung nicht im Akt der Werkschöpfung, sondern in einem Zuschreibungsprozess hat. Mit Blick auf Diskurse arbeitet Foucault vier Funktionen heraus, von denen die klassifizierende Funktion die erste ist: Autorennamen machen es möglich, Texte oder Gruppen von Texten zu bestimmen, von anderen zu unterscheiden und sie zu Werken zusammenzufassen – die Figur des Autors hat also einen Orientierungswert. Dies gilt umso mehr, da der Autorname nicht nur bezeichnet, sondern als ein Äquivalent einer Beschreibung fungiert (Foucault 2003, S. 243). Die zweite Funktion besteht in der Verknüpfung mit dem »rechtlichen und institutionellen System« (ebd., S. 251). Die Figur des Autors erlaubt im Rahmen des Urheberrechts eine Aneignung von Texten durch eine Person und damit das Entstehen einer spezifischen Eigentumsform. Drittens stellen Zuschreibungen eine »Projektion der Behandlung [dar], die man dem Text angedeihen lässt« (ebd., S. 248). Sie sind nicht völlig enthoben von den Eigenschaften und Eigentümlichkeiten des Textes, entscheiden aber über die Bedeutsamkeit, die ihm letztlich beigemessen wird. Die vierte Funktion von Autorschaft besteht darin, sich auf unterschiedliche *Egos*, wie etwa den Schriftsteller oder einen fiktionalen Sprecher, beziehen zu können: »die Funktion Autor vollzieht sich gerade in diesem Bruch – in dieser Trennung und dieser Distanz« (ebd., S. 250). Es gibt mehrere Positionen im Raum, die durchaus von verschiedenen Individuen eingenommen werden können.

Beide Texte wurden in den Literaturwissenschaften breit rezipiert und haben zu einer bis heute anhaltenden Debatte geführt. Kritik entfacht sich insbesondere an der behaupteten engen Verknüpfung des modernen Konzepts von Autorschaft mit dem Urheberrecht einerseits und dem bürgerlichen Individuum andererseits (Chartier 2003) sowie an der These der Verzichtbarkeit der Person des Autors für die Interpretation eines Textes, die von beiden nahegelegt wird (Arndt 2002, S. 129). Trotz der sehr grundsätzlichen Differenzen in der Konzeption dessen, was ein Autor ist, gibt es allerdings auch Kontinuität: Dem Begriff haftet Exklusivität an. Der Gebrauch in der älteren Literaturwissenschaft bezieht sich nicht allgemein auf Personen, die publizistisch tätig sind – der schreibende Jedermann ist hier nicht gemeint –, sondern der Begriff ist reserviert für Personen, denen

aufgrund außerordentlicher Leistungen ein hohes Maß an öffentlicher Anerkennung und Beachtung zukommt. Etwas anders gewendet findet sich Exklusivität aber auch bei Barthes und Foucault. Barthes' Kritik bezieht sich auf die Überhöhung von Autoren zu ›Autor-Göttern‹, ein Phänomen, das sich sicherlich nur auf einen sehr kleinen Teil der schreibenden Zunft beziehen lässt. Deutlicher tritt die Exklusivität des Begriffs noch bei Foucault hervor: Der Aspekt der Autor-Funktion, Texte zu klassifizieren, eine bestimmte Art von Rezeption zu erfordern, ist nur bei einem überschaubaren Kreis von Autorennamen denkbar. Denn dieser Vorgang setzt ein allgemeines Wissen über die Regeln der Rezeption ebenso voraus wie über die Bedeutung, mit der ein Text mit einem bestimmten Autorennamen zu versehen ist.

Bereits Johann Wolfgang von Goethe hat sich mit den Worten »mein Lebenswerk ist das eines Kollektivwesens, und dies Werk trägt den Namen Goethe«[23] gegen diese unzutreffenden Leitungszurechnungen gewandt. Trotz dieser Zweifel aufseiten mancher Autoren sowie der Kritiken von Barthes und Foucault hat sich deren Bestreben, die Figur des Autors zu ersetzen, handlungspraktisch nicht durchsetzen können.

## *3.2 Autorschaft in der Wissenschaftsforschung*

Das gerade angesprochene Problem des kollektiven Ursprungs von Texten bzw. der darin geäußerten Ideen und Gedanken ist auch in der Wissenschaft bekannt. Damit ist keineswegs nur das Sonderproblem gemeint, dass in großen Forschergruppen zum Teil mehr als Hundert Personen in den Autorenlisten von Publikationen geführt werden.[24] Das Problem ist grundsätzlicher: Von Isaac Newton ist beispielsweise eine Einschätzung seiner eigenen Leistung überliefert, die Parallelen zum obigen Zitat von Goethe aufweist: »If I have seen further it is by standing on ye shoulders of giants.«[25] Auch er rechnet die Leistung der Produktion des Werks nicht

[23] Johann Wolfgang von Goethe im Gespräch mit Frédéric Soret 1832, zitiert nach Haischer 2005, S. 229.

[24] In großen Forschungskooperationen mit mehreren Hundert Autoren ist es zum Problem geworden, Leistungen individuell zuzurechnen. Hier sind mindestens zwei Lösungen entstanden, um dem Problem zu begegnen: Die Nennung der jeweiligen Rolle, die die genannten Autoren beim Zustandekommen der Publikation gespielt haben, und die Nennung eines Autorenkollektivs anstelle einer Liste individueller Namen. Da durch diese zweite Form Leistungen, Verantwortung und Reputation nicht mehr einzelnen Personen zugerechnet werden können, müssen diese Funktionen auf anderem Wege ausgeübt werden. Zu einer Diskussion dieses Problems siehe Biagioli 2003.

[25] Brief an Robert Hooke, 5. Februar 1675/76, zitiert nach Westfall 1996, S. 143.

primär sich selbst zu, sondern betont den Ursprung im Kollektiv. Infolge der Differenzierung von Wissenschaft und Literatur (siehe hierzu Lepenies 1978) wurde der kollektive Ursprung von wissenschaftlichem Wissen anerkannt und es hat sich, im Unterschied zu literarischen Texten, eine institutionalisierte Form der Zurechnung von Leistungen entwickelt. Neben der Nennung von Autoren, die am unmittelbaren Zustandekommen einer Publikation und der darin berichteten Forschungsergebnisse beteiligt waren, kommt dies vor allem in der Referenzierung bereits publizierter Forschungsergebnisse zum Ausdruck. Diese institutionalisierte Praxis bildet eine Lösung für das Problem des kollektiven Ursprungs von wissenschaftlichem Wissen. In einem Text wird nur der Teil der Leistungen, der nicht auf andere Publikationen verweist, dem Autor zugerechnet. Der Eindruck, das gesamte in einer Publikation erwähnte Wissen sei Produkt des Autors, wird dadurch vermieden.

Diese Referenzierungspraxis wirft allerdings die Frage auf, weswegen sich Wissenschaftler den Mühen eines solchen komplexen Leistungszurechnungssystems unterwerfen. Was sind also die Bedingungen, unter denen sich dieses System hat stabilisieren können? In der Reflexionstheorie – der Wissenschaftsforschung – wird die Antwort im Rahmen von zwei Kontexten gesucht: einerseits im Reputationssystem und andererseits in den normativen Erwartungen, die mit der Rolle des wissenschaftlichen Autors verbunden sind, hier insbesondere die Autorenverantwortung. Das Reputationssystem ist seit den Arbeiten der Gruppe um Robert K. Merton zu einem regelmäßig untersuchten Gegenstand geworden.[26] Seitdem ist immer wieder betont worden,[27] dass Wissenschaftler nicht aufgrund monetärer Motive publizieren. Anstelle von ›Geld gegen Leistung‹ findet sich eine andere Form des Austauschs. Hagstrom (1982, S. 21 f.) argumentiert, dass Publikationen freiwillige, nicht von Kollegen einforderbare Leistungen darstellen und daher den Charakter von Gaben haben. Allerdings bleibt es nicht bei einem einseitigen Akt. Reziprozität wird hergestellt, sobald die Leistungen als Beiträge zum Fach anerkannt und zitiert werden (ebd., S. 22 f.). Eine solche Zitation stellt dabei nichts anderes dar als die Bewertung eines Beitrags zum Fach als beachtlich.[28] Verfestigt sich diese Leis-

[26] Im Zentrum der Untersuchungen von Merton und seiner Gruppe steht die Frage, ob die Leistungszurechnung durch nichtwissenschaftliche Faktoren, wie z. B. vorangegangene Leistungen oder sozialstrukturelle Faktoren, beeinflusst bzw. ›verzerrt‹ werde (Merton 1968; Cole 1972).

[27] Zuletzt geschah dies im Zusammenhang mit der Diskussion um Open Access (siehe hierzu Taubert/Weingart 2010, S. 170).

[28] Dies gilt auch für solche Publikationen, die zitiert werden, um kritisch auf sie Bezug zu nehmen. Auch in der kritischen Erwähnung kommt ein gewisses Maß an Beachtungswürdigkeit zum Ausdruck – schließlich könnten sie auch schlicht ignoriert werden.

tungszuschreibung, wird sie also von anderen Wissenschaftlern übernommen, entsteht im Rahmen eines kollektiven und dezentralen Prozesses Reputation. Die Aussicht auf eine solche Anerkennung und deren Verfestigung bildet gewissermaßen den Treibstoff, der die Veröffentlichung und Zirkulation von Forschungsergebnissen sicherstellt.

Verbunden mit der Rolle des Autors in der Wissenschaft sind allerdings auch normative Erwartungen, die insbesondere im Fall des Normverstoßes hervortreten.[29] Wissenschaft ist zu einem hohen Maße vertrauensbasiert und gründet darauf, dass Wahrheitsansprüche in aller Regel nicht im Einzelnen nachgeprüft werden. Mit der Rolle des Autors verbunden ist daher die Erwartung, dass die Forschung den methodischen, theoretischen und argumentativen Standards des Fachs entspricht, der Wissenschaftler bei der Durchführung der Forschung und bei der Interpretation der Ergebnisse sorgfältig vorgeht, die verwendete Literatur entsprechend genau referenziert und damit eigene und fremde Leistungen unterscheidbar macht. Zwar ist die Wissenschaft insofern fehlerfreundlich, als dass Irrtümer und sich als falsch erweisende Wahrheitsansprüche in der Regel nicht mit dem Entzug von Reputation sanktioniert werden. Empfindlich reagiert die Fach-Community allerdings im Fall mangelnder Sorgfalt und erst recht bei vorsätzlichen Normverstößen wie Fälschungen von Forschungsergebnissen oder Plagiaten. Solche drastischen Verstöße können den Ruf des betreffenden Wissenschaftlers durchaus beschädigen und verweisen auf die Existenz einer Autorenverantwortung.

Ein Unterschied zum Autorbegriff in den Literaturwissenschaften soll abschließend hervorgehoben werden. In den Literaturwissenschaften ist der Begriff wie bereits bemerkt an hohe Voraussetzungen geknüpft und hat folglich eine geringe Reichweite. Er beschränkt sich auf Personen, deren Werk in einer größeren Öffentlichkeit oder zumindest im Kreis der literarisch Interessierten bekannt ist. Demgegenüber ist der Autorbegriff in der Wissenschaftsforschung durch ein hohes Maß an Einschluss gekennzeichnet. Er inkludiert sämtliche Personen, die als Verfasser einer Publikation in Erscheinung getreten sind und beschränkt sich nicht auf von der Fach-Community hoch anerkannte Autoren, die häufig zitiert werden. Auf eine Formel gebracht lässt sich sagen: Während der literaturwissenschaftliche Autorbegriff voraussetzungsvoll ist, indem er öffentliche Bekanntheit voraussetzt, geht es im Fall des wissenschaftlichen Autors lediglich darum, dass ein Name im Metatext einer Publikation auftaucht und damit als Zuschreibungsadresse für Verantwortung und Reputation vor dem Publikum der *Peers* verfügbar wird.

[29] Siehe hierzu Franzen et al. 2007 und die dort genannten Formen wissenschaftlichen Fehlverhaltens der Erfindung und Fälschung von Forschungsergebnissen sowie des Plagiats. Diese Formen verweisen jeweils auf spezifische normative Erwartungen, die sich an den wissenschaftlichen Autor richten.

## *3.3 Autorschaft im Urheberrecht*

Im Unterschied zum US-amerikanischen Copyright verfügt das deutsche Urheberrecht über keinen Begriff des Autors. Eine Auseinandersetzung mit dem an dieser Stelle stehenden Begriff des Urhebers ist aber für den hier verfolgten Zweck erhellend, aus diesem Grund soll das Konzept ebenfalls knapp dargestellt werden. Historisch entwickelt hat sich das Urheberrecht in engem Zusammenhang mit der durch den Druck entstandenen Möglichkeit einer massenhaften Vervielfältigung schriftlicher Werke. Das damit auftretende Ordnungsproblem wurde zunächst als Problem des Investitionsschutzes wahrgenommen: Für die Produktion eines Buchs mussten hohe Investitionen für Beschaffung und Aufbereitung des Manuskripts getätigt werden, die sich Nachdrucker sparen konnten. Anfänglich wurde dieses Problem durch die Einräumung von Privilegien gelöst. Der für das heutige Urheberrecht konstitutive Gedanke des Schutzes der Beziehung zwischen Autor und Werk entwickelte sich langsam und gewann erst im 18. und 19. Jahrhundert Gestalt. Zwar wird in den Kommentaren zum Urheberrecht mit Blick auf diese Frühphase der Begriff des Autors benutzt (vgl. z. B. Rehbinder 2010, S. 9 f.), und auch in der häufig verwendeten Formel »*post mortem auctoris*«, mit der die über den Tod des Autors hinausreichende Fortgeltung des Urheberschutzes bezeichnet wird, ist der Autor präsent. Bereits aber das erste moderne deutsche Urheberrechtsgesetz (1837 in Preußen) bezieht sich nicht nur auf Schriftwerke, sondern schließt dramatische und musikalische Werke mit ein (vgl. ebd., S. 13). Aus diesem Grund ist im Urheberrechtsgesetz der Begriff des Autors zugunsten des umfassenderen Begriffs des Urhebers aufgegeben worden.

Zur Rekonstruktion der Figur des Urhebers ist es daher sinnvoll, zunächst von den Werken auszugehen, die Urheberschaft konstituieren, um dann zur Person zu kommen, die das Werk in die Welt gesetzt hat. Das heutige Urheberrecht zielt auf den Schutz von Werken in Literatur, Wissenschaft und Kunst und kennt eine Vielzahl von Werkarten. In § 2 UrhG werden die folgenden exemplarisch erwähnt: Sprachwerke, wie Schriftwerke, Reden und Computerprogramme, Werke der Musik, pantomimische Werke, Werke der Tanzkunst, Werke der bildenden Künste; Lichtbildwerke; Darstellungen wissenschaftlicher oder technischer Art, wie Zeichnungen, Pläne, Karten, Skizzen, Tabellen und plastische Darstellungen. Diese Aufzählung ist bewusst offengehalten, um zukünftig entstehende Werkarten mit dem Urheberrecht schützen zu können. Für die Definition des Begriffs ›Werk‹ muss für das deutsche Urheberrecht konstatiert werden, dass sie unterbestimmt ist und es an Klarheit fehlt. Dies führt zu Abgrenzungsproblemen gegenüber anderen Ergebnissen geistiger Arbeit. Die Definition im Gesetz – »Werke im Sinne dieses Gesetzes sind nur persönliche geistige Schöpfungen« (§ 2 UrhG) – beinhaltet drei Bestandteile: (1) persönlicher Ursprung, (2) geistiger Gehalt und (3) Formgebung (Schack 2010, S. 97 ff.).

Das erste Kriterium verweist insbesondere im deutschen Urheberrechtsgesetz auf das Individuum. Deutlich arbeitet Rehbinder dies heraus, der zwischen allgemeinmenschlichen Fähigkeiten und Anlagen individueller Natur unterscheidet. Das Urheberrechtsgesetz rechtfertigt den Schutz des Werks und die besondere Beziehung zu seinem Urheber nun im Rückgriff auf diese individuellen Anlagen und die Einmaligkeit der Persönlichkeit, die in einer originellen Gestalt des Werks ihren Ausdruck finden (Rehbinder 2010, S. 24). Nur Werken, die hinsichtlich ihres Inhalts oder ihrer Form originell sind und die nicht nur auf allgemeinmenschlichen Fähigkeiten basieren, komme ein Schutz zu. Das zweite Kriterium des geistigen Gehalts spricht dagegen an, dass es sich um eine kreative Leistung handeln muss, um vom Urheberrecht geschützt zu werden. Dem Werk muss also eine schöpferische Idee zugrunde liegen. Und drittens meint Formgebung, dass die geistige Schöpfung einen nach außen hin sichtbaren Ausdruck gefunden haben muss. Damit ist nicht zwangsläufig die Fixierung verlangt, vielmehr reichen auch vergängliche Formen wie die Performance oder die freie Rede aus.

Mit der Unterscheidung zwischen allgemeinmenschlicher Natur und individuellen Anlagen ist bereits eine Crux im Urheberrecht angesprochen. Da beides in einem mehr oder minder großen Umfang in einem Werk repräsentiert sein kann, stellt sich die Frage, welche Anforderungen an Originalität gestellt werden sollen. Hierzu sind zwei Dinge zu bemerken: Erstens legt das Urheberrecht für unterschiedliche Werkarten verschieden hohe Anforderungen für die Zubilligung eines Schutzes fest. Am deutlichsten tritt dies am Fall von Computerprogrammen hervor, die seit 1985 durch das Urheberrecht geschützt werden. Der Gesetzgeber fühlte sich aufgrund der hohen Investitionskosten aufgefordert, Computerprogrammen bereits bei geringer Schöpfungshöhe urheberrechtlichen Schutz zuzubilligen. Für sie ist es ausreichend, dass »sie individuelle Werke in dem Sinne darstellen, dass sie das Ergebnis der eigenen geistigen Schöpfung ihres Urhebers sind« (§ 69I UrhG). Im Unterschied zu anderen Sprachwerken sind damit praktisch alle auch sehr trivialen Computerprogramme urheberrechtlich geschützt. Zweitens lässt sich eine gewisse Diskrepanz zwischen der ursprünglichen Intention des Urheberrechts und der Rechtsprechung feststellen. Während das Urheberrechtsgesetz von seiner Anlage her und der Tendenz nach den rechtlichen Schutz exklusiv handhabt, hat die Rechtsprechung den Bereich schützenswerter Werke dagegen recht weit ausgelegt. Dies wird in den Kommentaren zum Urheberrecht allgemein bemerkt (Schack 2010, S. 102) und zum Teil auch beklagt (Rehbinder 2010, S. 31 u. 69).

Welche Eigenschaften zeichnen nun den Urheber in der rechtswissenschaftlichen Diskussion aus? In den Kommentaren wird er als Person charakterisiert, deren Chancen zur Verwirklichung ihrer Interessen gering sind: Da sich Geisteswerke im Unterschied zu körperlichen Werken nur

schwer schützen lassen und leicht Kopien hergestellt werden können, liegt Nichtausschließbarkeit vor. Gelangt das Werk in den Umlauf, kann die Allgemeinheit leicht darauf zugreifen und davon Gebrauch machen, ohne dass der Urheber es verhindern kann (Rehbinder 2010, S. 45). Um dies zu vermeiden und um den Urheber in die Lage zu versetzen, sich gegen einen nicht-autorisierten Werkgenuss zur Wehr setzen zu können, schafft das Urheberrecht ein zeitlich befristetes Monopol über die Verfügungsrechte am Werk zugunsten des Autors. Der Urheber ist also im Urheberrechtsgesetz vor allem Träger von Rechten, die sich auf seine materiellen, vermögensrechtlichen und seine persönlichkeitsrechtlichen Interessen beziehen. Zu letztgenannten zählen u. a. das Veröffentlichungsrecht, das Bearbeitungsrecht und der Schutz vor Entstellung des Werks.

## 4. Funktionen von Autorschaft in der Open-Source-Softwareentwicklung

Bereits die Beschreibung der Formen von Namensnennung im zweiten Abschnitt lässt vermuten, dass das Phänomen bei der Entwicklung von Open-Source-Software eine zentrale Rolle spielt. Um die Namensnennung wird ein erheblicher Aufwand betrieben und sie zieht sich durch die Darstellung des Projekts und die Projektkommunikation und ist auch im Arbeitsergebnis selbst anzutreffen. Diese Präsenz des Phänomens führt zur Frage nach dessen Funktion; und die unterschiedlichen Kontexte, in denen Namen auftauchen, lassen vermuten, dass die Frage nicht einheitlich mit Verweis auf eine einzige Funktion zu beantworten ist. Für eine Antwort werden ausgehend von den Überlegungen des letzten Abschnitts zum Begriff des Autors die Formen der Namensnennung mit drei verschiedenen Typen von Material kontextualisiert: der Kommunikation innerhalb von Open-Source-Softwareentwicklungsprojekten, Interviews mit den beteiligten Entwicklern aus dem Projekt sowie Texten mit selbstbeschreibendem Charakter.

### *4.1 Autorschaft und die Konstitution von Open-Source-Software*

Für das Verständnis der ersten Funktion braucht es keine Kontextualisierung, da sich im bisherigen Argumentationsgang bereits alle Bestandteile finden, sodass diese hier nur noch zusammengefügt werden müssen. Im Fall der Namensnennung im Programmquellcode hat sich gezeigt, dass der Autor als Träger bestimmter Rechte auftritt, eine Figur, die aus der Urheberrechtsdiskussion im vorangegangenen Abschnitt bekannt ist. Im Quellcode reklamiert der Autor diese Rechte durch die Nennung des Copy-

right-Zeichens, um dann direkt im Anschluss und unter Zuhilfenahme der GNU General Public License die Rechte zu Nutzung, Vervielfältigung, Modifikation und Vertrieb an jedermann weiterzugeben. Eine solche Generalisierung der Rechte setzt voraus, dass es eine Person gibt, die über die Rechte verfügen und über eine Vergabe entscheiden kann. Die Funktion des Autors – hier verstanden als Träger von Rechten – liegt also in der *Konstituierung von Open-Source-Software*. Der Autor bildet den Transmissionspunkt, durch den dieses besondere Regime von Rechten auf das Programm und dessen Entwicklung angewandt wird. Auf einen weiteren, wesentlichen Punkt dieses Vorgangs soll an dieser Stelle noch hingewiesen werden. Die Einräumung von Rechten an jedermann ist eine einseitige Leistung, die nicht durch eine Entlohnung oder andere direkte Gegenleistungen entgolten wird. Von der Struktur her ähnelt dieser Vorgang den weiter oben dargestellten Austauschprozessen der Publikation von Forschungsergebnissen in der Wissenschaft. Auch hier leisten Wissenschaftler ohne unmittelbare Gegenleistung Beiträge zu einem gemeinsamen Wissensstand.

## 4.2 *Autorschaft und die fachliche Anerkennung von Leistungen*

In der Literatur zur Open-Source-Softwareentwicklung wird diese Einseitigkeit der Leistungserbringung angemerkt und es wird festgestellt, dass in der Regel keine monetären Anreize für eine Beteiligung an Community-getriebenen Projekten bestehen (Lerner/Tirole 2001, S. 822; von Krogh/von Hippel 2003, S. 1152).[30] Mit Blick auf die gerade aufgezeigte Parallele zum wissenschaftlichen Autor ist daher zu fragen, ob auch im Fall von Open-Source-Software ähnliche, ›weichere‹ Anreize bestehen, wie z. B. die Anerkennung von Leistungen durch Peers. Die Verknüpfungen von Beiträgen mit Namen im Programmquellcode und auf der Mailingliste einerseits sowie deren öffentliche Einsehbarkeit andererseits stellen zumindest die Voraussetzungen her, die eine Evaluierung und Zurechnung von Leistungen ermöglichen würden. In den Interviews, die dieser Untersuchung zugrunde liegen, findet sich Evidenz dafür, dass die Community der Open-Source-Softwareentwickler analoge Strukturen kennt.

> »Jetzt auch auf der Cebit habe ich gerade die ›Intevation‹-Leute aus Osnabrück, DEV_1, kennengelernt, der GnuPG programmiert, und DEV_2, der SourceForge entwickelt hat. Und solche Leute lernt man alle kennen und einige haben dann auch Respekt vor mir – im Prinzip schon. (lacht)« (KMail_12)

[30] Siehe auch Lakhani/Wolf 2003, S. 3; Hertel et al. 2003; Osterloh et al. o. J.; und den Forschungsstand kritisch zusammenfassend Holtgrewe/Brand 2007, S. 29 f.

Das Zitat zeigt, dass die Bekanntschaft und die Hochachtung anderer Entwickler als positiv und anstrebenswert wahrgenommen wird und der Wert dieser Hochachtung abhängig ist von den Leistungen, die der *andere* Entwickler erbracht hat. Genannt werden hier ausschließlich renommierte Entwickler, die als Urheber bekannter Programme vorgestellt werden. Die soziale Struktur der Community der Open-Source-Softwareentwickler ist also eine Meritokratie. Status und Stellung einer Person basieren auf anerkannten Leistungen, sei es für die Entwicklung eines spezifischen Projekts, sei es für Open-Source-Software allgemein. (vgl. Feller/Fitzgerald 2000, S. 64; Roberts et al. 2006, S. 992; Aberdour 2007, S. 59)[31] Voraussetzung einer solchen Meritokratie ist, dass innerhalb der Community ein Bestand an Wissen darüber existiert, welcher Entwickler in welchem Kontext Leistungen erbracht hat.[32] In diesen Zusammenhang ordnet sich Autorschaft ein: Obschon es schwierig und mit einem großen zusätzlichen Aufwand verbunden ist, bei der kollektiven Produktion eines Computerprogramms zu dokumentieren und sichtbar zu machen, welche Leistungen von wem stammen, wird genau dies unternommen. Die zweite Funktion der Namensnennung im Bereich der Open-Source-Softwareentwicklung liegt in der Herstellung der Bedingungen der Möglichkeit der Anerkennung von Leistungen und daran anschließender fachlicher Reputation durch die Bereitstellung von Zuschreibungsadressen.[33]

[31] Eine Bestätigung findet diese Diagnose auch in der Selbstbeschreibung der Open-Source-Softwareentwickler, die nicht nur selbst den Begriff der Meritokratie verwenden, sondern auch Ähnlichkeiten zwischen der Gruppe der Hacker und wissenschaftlichen Communities sehen. Das New Hacker's Dictionary erläutert zum Begriff Hacker: »It is better to be described as a hacker by others than to describe oneself that way. Hacker consider themselves something of an elite (a meritocracy based on ability), though one to which new members are gladly welcome.« (Raymond 1998, S. 234). Der Ehrentitel des Hackers wird verliehen durch Peers, die ebenfalls über Programmierkompetenzen verfügen und die Qualität der Arbeit beurteilen können: »Accordingly, when you play the hacker game, you learn to keep score primarily by what other hackers think of your skills (this is why you aren't really a hacker until other hackers consistently call you one)« (Raymond 1999, S. 242).

[32] Analoges gilt für die Wissenschaft. Luhmann spricht davon, dass »ein beträchtliches, stets auf dem laufenden zu haltendes Reputationswissen […] zu den overhead costs der Wissenschaft« gehört (Luhmann 1970, S. 235).

[33] Der Erwerb von fachlicher Reputation setzt sich weiter um in den Einfluss auf Entscheidungen, die im Rahmen des betreffenden Projekts gefällt werden (vgl. ausführlicher Roberts et al. 2006, S. 985; Taubert 2008).

## 4.3 *Autorschaft und Anerkennung durch Anwender*

Während der wissenschaftssoziologische Begriff von Autorschaft den Fokus auf das Verhältnis des Autors zu seinen Peers legt, bezieht sich der literaturwissenschaftliche Begriff auf das Verhältnis von Autor und Publikum. Hier stellt sich die Frage, ob dieses Publikumsverhältnis im Fall von Open-Source-Software ebenfalls von Bedeutung ist. Genauer gefragt: Existiert also eine Form der Anerkennung durch einen Kreis von Personen, die nicht zu der Gruppe der Peers gehören? Evidenz dafür findet sich an erster Stelle auf der Mailingliste des Projekts, über die regelmäßig E-Mails wie die folgende verschickt werden:

```
Hello,
I want to send a warm thank you to you and to all the kmail
team. I knew this program at it's [sic] early stage and now
it is a really good Software.
Thanks again,
KMail_13 (User)
```

Beispiel 4: E-Mail eines Users an KMail

Die E-Mail gibt ein Beispiel dafür, dass auch das Verhältnis zwischen Entwicklern und Anwendern durch ein gewisses Maß an Reziprozität gekennzeichnet ist. Die Entwickler stellen den Nutzern ein Programm zur Verfügung und erhalten im Gegenzug Anerkennung für ihre Leistung, die hier die Form von Lob und Dank annimmt. Im Unterschied zur fachlichen Anerkennung geht es dabei nicht um die Würdigung der Qualität einer Programmierleistung. Dies ist schon dadurch ausgeschlossen, dass es den Anwendern in der Regel an den fachlichen Kompetenzen fehlt, um die Leistung beurteilen zu können. Anerkennung durch die Benutzer bezieht sich stattdessen auf die *Verwendbarkeit* und *Nützlichkeit* des Programms zur Bewältigung alltäglicher Aufgaben. Die Bedeutung dieser Form von Anerkennung liegt darin, dass sie es den Entwicklern ermöglicht, ihr Engagement für das Projekt als sinnvoll – im Sinne von nützlich – zu interpretieren:

> »Gerade auch, wenn man sich da vorstellt, dass Linux jetzt irgendwie 2 Prozent Desktop-Markt hat und man sich dann ausrechnet, 70 Prozent läuft unter KDE. Und wenn dann 50 Prozent von KDE[-Anwendern; Anm. NT] KMail benutzt, dann sind das Hunderttausende von Leuten, die dieses Programm benutzen. Das ist schon beeindruckend. Dass man da irgendwas geschrieben hat, wird benutzt. Das ist schon ziemlich gut.« (KMail_10)

Mit Blick auf das Verhältnis von Entwicklern und Anwendern stellt Autorschaft ebenfalls Adressierbarkeit her, und dies geschieht vorrangig durch

die Nennung der Namen auf der Webseite des Projekts. Diese Liste der Autoren macht sichtbar, dass sich das Kollektiv aus individuellen Personen zusammensetzt und dass sowohl das Kollektiv als auch Einzelpersonen zum Adressaten von Lob und Dank gemacht werden können.[34] Diese Form individualisierter Anerkennung setzt voraus, dass Leistungen mit Personen in einer Form in Verbindung gebracht werden, die auch von Anwendern nachvollzogen werden kann. Eine solche Möglichkeit wird durch die Nennung von Personennamen im KMail-Handbuch geschaffen.

## 4.4 *Autorschaft und Verantwortung für den Projektfortschritt*

Eine letzte Auffälligkeit von Autorschaft im hier interessierenden Fall ist, dass bei bestimmten Typen von Namensnennungen nicht nur eine Verknüpfung von Leistungen und Personen stattfindet, sondern daneben auch noch eine E-Mail-Adresse angegeben wird. Dies ermöglicht eine direkte Kontaktaufnahme mit der betreffenden Person. Die Typologie der Namensnennung im zweiten Abschnitt hat gezeigt, dass E-Mail-Adressen insbesondere in Kontexten angegeben werden, die für die Weiterentwicklung des Projekts von Bedeutung sind. Das paradigmatische Beispiel bildet hier die Namensnennung im Quellcode des Programms. An dieser Stelle soll die These vertreten werden – wiederum in Analogie zur Wissenschaft –, dass mit der Rolle des Autors normative Erwartungen verbunden sind, die sich als Autorverantwortung bezeichnen lassen. Sie beziehen sich nicht ausschließlich auf die Qualität des Beitrags, sondern auch auf die Weiterentwicklung der Software und den Fortgang des Projekts. Diese Verantwortung findet sich gewissermaßen ›im Großen‹ auf der Ebene des Projekts und tritt deutlich in einem Text mit selbstbeschreibendem Charakter hervor. Eric Raymond, ein Entwickler, der eine Vielzahl von Texten zur Kultur der Open-Source-Softwareentwickler verfasst hat, diskutiert den Fall der Ablösung des ›owner‹ eines Projekts durch seinen Nachfolger und beschreibt die Pflicht des alten Projektleiters wie folgt: »It is well understood in the community that project owners have a duty to pass projects to competent successors when they are no longer willing or able to invest needed time in development or maintenance work.« (Raymond 1999, S. 90) Diese normative Erwartung existiert allerdings nicht nur auf der Ebene des ge-

[34] Im Einzelfall kann es auch zu einer Verfestigung dieser Form der Anerkennung und dem Entstehen von Popularität kommen. Ein solcher Fall ist der Entwickler von Linux – Linus Torvalds –, dessen Bild in der Öffentlichkeit überaus positiv ist. Dieses Bild ist aber durch Massenmedien vermittelt und basiert nicht auf der unmittelbaren Anerkennung des Nutzens eines Programms durch Anwender.

samten Projekts, sondern auch in Bezug auf die einzelnen Programmkomponenten. Der Autor einer Komponente bildet den Ansprechpartner, wenn ein anderer Entwickler Modifikationen in größerem Umfang plant. Ein Entwickler, der über Erfahrung in mehreren Projekten verfügt, hat u. a. kurzzeitig an der Entwicklung des Apache-Webservers mitgewirkt. Die folgende Textpassage zeigt, dass die Fertigstellung seines Beitrags nicht das Ende seiner Beteiligung am Projekt bedeutete:

> »Aber es haben zwei, drei Leute Patches geschickt. Die Gruppenimplementierung war Grund mehrerer Patches und die ursprüngliche Notierung kam durch nen Patch, dann hab ich die modifiziert und dann hat irgendjemand anderes noch mal was Portableres geschickt […]. Und irgendwie jetzt vor Kurzem hat jemand einen Patch geschickt, mit dem man noch mehr Konfigurationsdateien verwenden kann, also dann kann man halt auf allen Webservern verschiedene Formen der Authentifizierung benutzen. So was habe ich dann eingebaut.« (DEV_3)

An dieser Passage interessieren nicht so sehr die technischen Details, sondern vielmehr der Umstand, dass der ursprüngliche Entwickler eines Programmbestandteils auch im weiteren Verlauf als Ansprechpartner für Beiträge (Patches) zu diesem Teil des Programms fungiert, und dies auch in einer Situation, in der der Entwickler seine Arbeit für das Projekt eigentlich als abgeschlossen betrachtet. Die vierte Funktion von Autorschaft bildet also die Verantwortung für die Kontinuierung des Projekts und der Entwicklungsarbeit.

## 5. Schluss

Ausgehend von den vielfältigen Formen der Namensnennung wurde die Frage nach der Funktion von Autorschaft im Bereich der Open-Source-Softwareentwicklung gestellt. Herausgearbeitet wurde, dass an den Autor mindestens vier Funktionen anknüpfen: Erstens konstituiert der Autor Open-Source-Software, da er als Träger von Rechten diese nutzt, um sie zu generalisieren. Zweitens wird mit Autorschaft die Voraussetzung hergestellt, um Programmierleistungen fachlich anzuerkennen. Diese Anerkennung und ihre kondensierte Form der Reputation bilden ein wesentliches Motivationsmittel für die Entwickler, sich über längere Zeiträume hinweg an einem Projekt zu beteiligen. Drittens ermöglicht Autorschaft auch eine Anerkennung durch Nutzer. Diese zweite, nicht-fachliche Form ist ebenfalls als Motivationsmittel von Bedeutung, da sie es erlaubt, die geleistete Programmiertätigkeit als nützlich zu interpretieren. Und viertens besitzt Autorschaft im Fall von Open-Source-Software auch eine normative Komponente, die sich als Verantwortung für den Fortgang des Projekts bezeichnen lässt.

Offen bleibt die Frage nach dem Begriff des Autors, der dieser Analyse zugrunde liegt. Erkenntnisträchtig waren sämtliche der im dritten Schritt referierten Begriffe. Es dürfte deutlich geworden sein, dass Autorschaft hier rollentheoretisch verstanden wird und sich aus sämtlichen im dritten Schritt referierten Konzepten bedient. Der Autor ist gleichermaßen öffentlicher und fachlich anerkannter Urheber von Leistungen, Träger von Rechten und Adressat von normativen Rollenerwartungen. Auf der Hand liegt dabei, dass die Rolle des Autors keine exklusive Figur ist wie etwa in den Literaturwissenschaften, sondern eher vergleichbar ist mit dem publizierenden Wissenschaftler, der mehr oder minder große Beiträge zu seinem Fach leistet. Die letzte Bemerkung soll sich auf die Art des geschaffenen Werks beziehen. Auch hier erweist sich das literaturwissenschaftliche, stark an ›Text‹ gebundene Verständnis von Autorschaft als zu eng. Um den Begriff des Autors auch auf Open-Source-Software anwenden zu können, muss er sich auf eine breiter gefasste Gruppe von Werken beziehen. Der aus dem Urheberrecht stammende und bereits erwähnte Begriff des Sprachwerks scheint dafür geeignet zu sein.

Eine kollektive Arbeit an einem sich ständig verändernden digitalen Werk bildet gewiss keine günstige Rahmenbedingung für Autorschaft. Das macht das Beispiel der Open-Source-Softwareentwicklung so interessant. Das Beispiel legt aber nahe, dass – solange keine Alternativen entstehen, die in der Lage sind, die in diesem Aufsatz herausgearbeiteten Funktionen zu übernehmen – trotz der auf den ersten Blick sehr ungünstigen Bedingungen auch künftig mit der alt eingeübten und eingelebten Autorschaft zu rechnen ist.

## Literaturverzeichnis

Aberdour, Mark (2007): »Achieving Quality in Open Source Software«, in: *IEEE Software* 24(1), S. 58–64.

Arndt, Andreas (2002): »Subjektivität und Autorschaft«, *in*: *edito* 16, S. 1–13.

Barthes, Roland (2000): »Der Tod des Autors«, *in*: Fotis Jannidis, Gerhard Lauer, Mathias Martiney und Simone Winko (Hg.): *Texte zur Theorie der Autorschaft*, Stuttgart: Reclam, S. 185–197.

Biagioli, Mario (2003): »Rights or Rewards? Changing Framework of Scientific Authorship«, in: ders. und Peter Galison (Hg.): *Scientific Authorship. Credit and Intellectual Property in Science*, New York und London: Routledge, S. 259–273.

Bonaccorsi, Andrea/Rossi, Christina (2003a): *Licensing schemes in the production and distribution of Open Source Software. An empirical investigation*. Institute for Informatics and Telematic: Working Paper, URL: http://ifipwg213.org/sites/flosshub.org/files/bnaccorsirossilicense.pdf [Zugriff am 17.01.2014].

Bonaccorsi, Andrea/Rossi, Christina (2003b): »Why Open Source Software can succeed«, in: *Research Policy* 32(7), S. 1243–1258.

Brand, Andreas/Holtgrewe, Ursula (2005): »KDE im Kontext: Open Source Software Entwicklung und öffentliche Güter«, in: Manfred Moldaschl und Hajo Weber (Hg.): *Wissen und Innovation – Beiträge zur Ökonomie der Wissensgesellschaft*, Marburg: Metropolis.

Chartier, Roger (2003): »Foucault's Chiasmus. Authroship between Science and Literature in the Seventeenth and Eighteenth Century«, in: Mario Biagioli und Peter Galison (Hg.): *Scientific Autorship. Credit and Intellectual Property in Science*, New York und London: Routledge, S. 13–33.

Cole, Stephen (1972): »Wissenschaftliches Ansehen und die Anerkennung wissenschaftlicher Leistungen«, in: Peter Weingart (Hg.): *Wissenschaftssoziologie Bd. I: Wissenschaftliche Entwicklung als sozialer Prozeß*, Frankfurt a. M.: Fischer, S. 165–187.

Crowston, Kevin/Wei, Kangning/Li, Qing/Howison, James (2006): »Core and Periphery in Free/Libre and Open Source Software Team Communications«, in: *Proceedings of the 39th Annual Hawai'i Conference on System Sciences*, Waikoloa, URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1579526 (doi: 10.1109/HICSS.2006.101.) [Zugriff am 17.01.2014].

Feller, Joseph/Fitzgerald, Brian (2000): »A framework analysis of the open source Software development paradigm«, in: *Proceedings of the Twenty-First International Conference on Information Systems*, ICIS 2000, Brisbane, Australia, December 10–13, 2000, herausgegeben von Soon Ang, Helmut Krcmar, Wanda J. Orlikowski, Peter Weill und Janice I. DeGross, Association for Information Systems, S. 58–69.

Foucault, Michel (2003): »Was ist ein Autor?«, in: ders.: *Schriften zur Literatur*, herausgegeben von Daniel Defert und Francois Ewald unter Mitarbeit von Jaques Lagrange, Frankfurt a. M.: Suhrkamp, S. 234–270.

Franzen, Martina/*Rödder, Simone*/Weingart, Peter (2007): »Fraud: causes and culprits as perceived by science and the media«, in: *EMBO reports* 8(1), S. 3–7.

Gacek, Cristina/Arief, Budi (2004): »The Many Meanings of Open Source«, in: *IEEE Software* 21, S. 277–304.

Gläser, Jochen (2006): *Wissenschaftliche Produktionsgemeinschaften: Die soziale Ordnung der Forschung*, Frankfurt a. M.: Campus.

Hagstrom, Warren O. (1982): »Gift as an organizing principle in science«, in: Barry Barnes und David Edge (Hg.): *Science in Context. Readings in the Sociology of Science*, Cambridge: MIT Press, S. 21–34.

Haischer, Peter-Henning (2005): »Ruine oder Monument? Goethes Lebenswerk im Spiegel seiner Gotik-Studien«, in: Werner Frick, Jochen Golz und Edith Zehm (Hg.): *Goethe Jahrbuch Bd. 122*, *Göttingen: Wallenstein*, S. 215–229.

Hertel, Guido/Nieder, Sven/Herrmann, Stefanie (2003): »Motivation of Software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel«, in: *Research Policy* 32(7), S. 1159–1177.

Holtgrewe, Ursula/Brand, Andreas (2007): »Die Projektpolis bei der Arbeit. Open Source-Software-Entwicklung und der ›neue Geist des Kapitalismus‹«, in: Österreichische Zeitschrift für Soziologie 32(3), S. 25–45.

Jannidis, Fotis/Lauer Gerhard/Martiney, Mathias/Winko, Simone (2000): »Einleitung. Autor und Interpretation«, *i*n: dies. (Hg.): *Texte zur Theorie der Autorschaft*, Stuttgart: Reclam, S. 7–29.

Koch, Stefan/Schneider, Georg (2002): »Effort, co-operation and co-ordination in an open source Software project: GNOME«, in: *Information Systems Journal* 12, S. 27–42.

Lakhani, Karim R./Wolf, Robert G. (2003): *Why Hackers Do What They Do: Understanding Motivation Efforts in Free/Open Source Software Projects*. MIT Sloan School of Management, Working-Paper 4425-03, URL: http://ocw.mit.edu/courses/sloan-school-of-management/15-352-managing-innovation-emerging-trends-spring-2005/readings/lakhaniwolf.pdf [Zugriff am 17.01.2014].

Lepenies, Wolf (1978): »Der Wissenschaftler als Autor«, *i*n: *Akzente* 25(2), S. 129–147.

Lerner, Josh/Tirole, Jean (2001): »The Open Source Movement: Key Research Questions«, in: *European Economical Review* 45, S. 819–826.

Lerner, Josh/Tirole, Jean 2002: The *Scope of Open Source Licenses*. MIT Working Paper, URL: http://www.people.hbs.edu/jlerner/OSLicense.pdf [Zugriff am 17.01.2014].

Luhmann, Niklas (1970): »Selbststeuerung der Wissenschaft«, in: ders.: *Soziologische Aufklärung. Bd. 1*, Opladen: Westdeutscher Verlag, S. 232–252.

Merton, Robert K. (1968): »The Matthew Effect in Science«, in: *Science* 159(3810), S. 56–63.

O'Mahoney, Siobhán (2003): »Guarding the Commons: How Community Managed Software Projects Protect their Work«, in: *Research Policy* 32(7), S. 1179–1198.

Osterloh, Margit/Rota, Sandra/Kuster, Bernhard (o. J.): *Open source Software production. Climbing on the shoulders of giants*, URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.3183&rep=rep1&type=pdf (doi: 10.1.1.12.3183) [Zugriff am 17.01.2014].

Raymond, Eric S. (1998): *The New Hacker's Dictionary*, Cambridge und London: The MIT Press.

Raymond, Eric S. (1999): *The Cathedral & the Bazaar. Musings on Linux and Open Source by an Accidental Revolutionary*, Peking et al.: O'Reilly.

Rehbinder, Manfred (2010): *Urheberrecht*, 16., neu bearb. Aufl., München: C.H. Beck.

Roberts, Jeffrey A,/Hann, Il-Horn/Slaughter, Sandra A. (2006): »Understanding the Motivation, Participation, and Performance of Open Source Software-Developers: A Longitudinal Study of the Apache Project«, in: *Management Science* 52(7), S. 984–999.

Schack, Haimo (2010): *Urheber- und Urhebervertragsrecht*, 5., neu bearb. Aufl., *Tübingen: Mohr Siebeck*.

Stallman, Richard M. (2002): *Free Software, Free Society: Selected Essays of Richard M. Stallman*, herausgegeben von Joshua Gay, Boston: GNU Press.

Taubert, Niels (2006): *Produktive Anarchie? Netzwerke freier Softwareentwicklungsprojekte*, Bielefeld: transcript.

Taubert, Niels (2008): »Balancing Requirements of Decision and Action. Decision and Implementation in Free/Open Source Software Projects«, in: *STI-Studies* 4(1), S. 69–88.

Taubert, Niels/Weingart, Peter (2010): »Open Access – Wandel des wissenschaftlichen Publikationssystems«, in Tilmann Sutter und Alexander Mehler (Hg.): *Medienwandel als Wandel von Interaktionsformen*, Wiesbaden: VS, S. 159–181.

Tuomi, Ikka (2005): »The Future of Open Source«, in: Marleen Wynants und Jan Cornelis (Hg.): *How Open is the Future? Economic, Social and Cultural Scenarios Inspired by Free and Open Source Software*, Brüssel: VUB Brussels University Press, S. 429–459.

von Krogh, Georg/von Hippel, Eric (2003): »Editorial. Special Issue on Open Source Software Development«, in: *Research Policy* 32, S. 1149–1157.

Westfall, Richard (1996): *Isaac Newton. Eine Biographie*, Heidelberg et al.: Spektrum Akademischer Verlag.