Open-Source-Projekte als utopischer Gegenentwurf, Entwicklungsmethode und Innovationsstrategie

Jan-Felix Schrape

Beitrag zur Ad-hoc-Gruppe: „Open-Bewegungen": Die Kritik der Geschlossenheit

‚Open' ist zu einem ubiquitären Beiwort der digitalen Moderne avanciert – von ‚Open Science' über ‚Open Innovation' bis hin zu ‚Open Government'. Ein zentraler Ausgangspunkt für die derzeitige Popularität des Offenheitsparadigmas liegt in dem Bedeutungszuwachs von Open-Source-Projekten in der Softwareentwicklung seit der Jahrtausendwende, der in den Sozialwissenschaften angesichts klassischer Sichtweisen, die ‚intellectual property rights' als wesentliche Treiber in Innovationsprozessen ansehen, zunächst mit Erstaunen zur Kenntnis genommen (Lessig 1999) und danach von einigen Autor/-innen als Beleg für die Emergenz eines neuen Produktionsmodells gedeutet wurde, das auf freiwilliger Kollaboration beruht, den Stellenwert korporativer Akteur/-innen schmälert und eingespielten Formen ökonomischer Koordination überlegen sein könnte (Lakhani, Hippel 2003). Insbesondere die durch Benkler (2002) popularisierte Idee der ‚commons-based peer production' als technisch effektivierte „collaboration among large groups of individuals [...] without relying on either market pricing or managerial hierarchies" (Benkler, Nissenbaum 2006: 394) erfuhr intensive Reflexion und Anwendung auf angrenzende Kontexte (zum Beispiel Rifkin 2014; Bennett et al. 2014).

Gerade in der Beobachtung von Open-Source-Gemeinschaften zeigt sich allerdings inzwischen sehr deutlich, dass sich mit wachsender Projektgröße regelmäßig prägnante hierarchische Entscheidungsmuster herausbilden, führende IT-Konzerne mit steigender Marktrelevanz der Entwicklungsvorhaben häufig einen erheblichen Einfluss auf deren Orientierung erlangen und viele aktive Projekte vorrangig durch die Beiträge von in Unternehmen angestellten Softwareentwickler/-innen getragen werden. In der oft als typisches Beispiel herangezogenen Linux-Kernel-Community etwa wurden zuletzt über 80 Prozent der Aktualisierungen von Programmierer/-innen durchgeführt, „who are being paid for their work" (Corbet et al. 2015: 11). Angesichts dieser Verschränkungen reichen die nach wie vor üblichen eher pauschalen Verweise auf Open-Source-Communitys als Alternative zur kommerziellen Softwareentwicklung nicht mehr aus, um dem breiten Spektrum an unterschiedlich ausgerichteten Projekten gerecht zu werden.

Vor diesem Hintergrund entfaltet dieser Beitrag auf der Basis von Marktdaten, Hintergrundgesprächen sowie Dokumentenanalysen einen systematisierenden Überblick über die sich wandelnden Relationen zwischen Open-Source-Projekten und kommerziellem IT-Markt (eingehender: Schrape 2016). Dabei wird deutlich, dass die freie Softwareentwicklung ihre Formatierung als Gegenentwurf inzwischen weitgehend verloren hat, sich aber ebenso wenig als bloßes Addendum traditioneller Koordinationsmuster fassen lässt: Eingebettet in die sich erweiternden kommunikationstechnischen Möglichkeiten haben ab Ende der 1980er Jahre neuartige lizenzrechtliche Konstruktionen für quelloffene Software den institutionellen Rahmen für eine auf Dauer gestellte Form kollektiver Invention aufge-

spannt, die zuerst in subversiven Nischen Anwendung fand, anschließend korporative Aneignung erfahren hat und heute in den allgemeinen Kanon der Softwarebranche übergegangen ist (Abbildung 1).

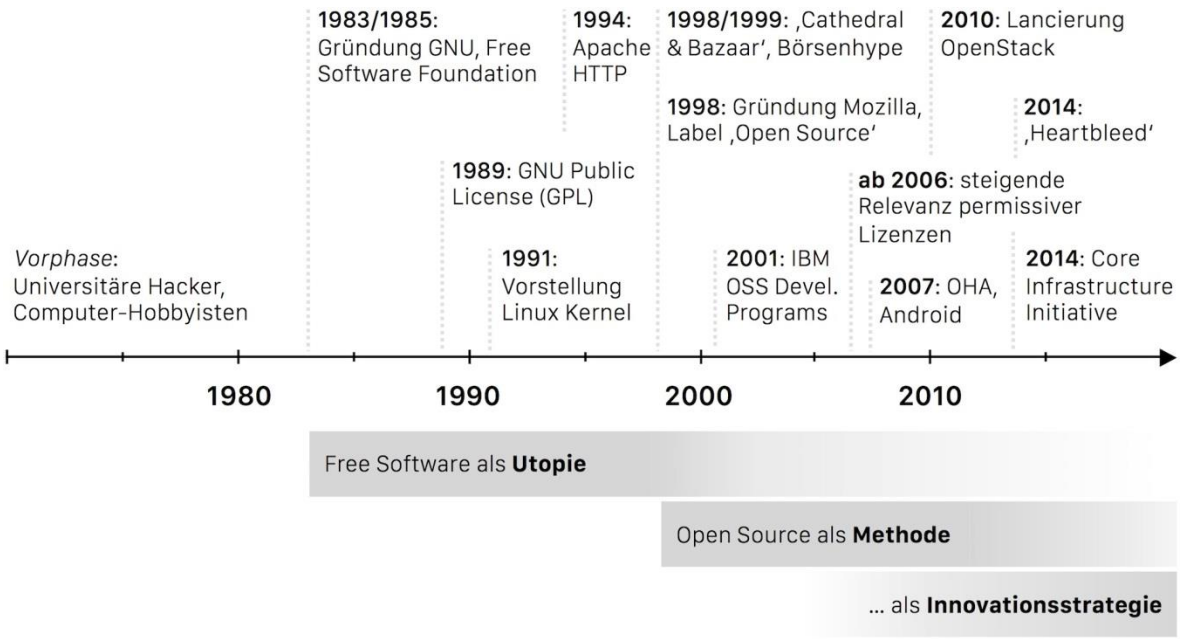

Abbildung 1: Phasen der Open-Source-Softwareentwicklung

Freie Software als utopischer Gegenentwurf

Die Herausbildung des ‚free software movements' in den 1980er Jahren lässt sich als eine direkte Reaktion auf die zuvor angestoßene Kommodifizierung von Software verstehen: Während Programmcode bis in die 1960er Jahre hinein weder von Anbietern noch Kunden und Kundinnen als von der Hardware unabhängiges Gut wahrgenommen wurde, sondern „as a research tool to be developed and improved by all users" (Gulley, Lakhani 2010: 6), wurde Software in den 1960er und 1970er Jahren auch aufgrund entsprechender kartellrechtliche Verfahren immer deutlicher als separates Produkt sichtbar.

Für die Ausdifferenzierung einer eigenständigen Softwarebranche spielte zudem die Verbreitung von Minicomputern ab 1960 eine wichtige Rolle: Zum einen waren sie im Betrieb günstiger als Mainframes und daher nicht auf eine möglichst effiziente Nutzung ausgelegt; zum anderen beförderten erweiterte Ein- und Ausgabeschnittstellen die Herausbildung neuer Softwaregenres. Im amerikanischen akademischen Milieu boten institutsöffentlich erfahrbare Minicomputer überdies einen Nährboden für informelle Projektgruppen, die mit ihren Arbeiten wiederum die Basis für die sich ab 1975 entlang der ersten Heimcomputer herausbildende Amateur-Computing-Szene schufen (Levy 1984). Das geteilte Problem der in diesen Kontexten entwickelten Architekturen lag indes in ihrer mangelnden rechtlichen Absicherung: Sie wurden meist gemeinfrei veröffentlicht und waren kaum vor Einzelaneignung geschützt. Unix etwa wurde durch AT&T ab 1983 – sobald es kartellrechtlich möglich war – kommerzialisiert.

Ein Schwierigkeit, die vice versa für gewerbliche Anbieter mit der Hacker- und Hobbyisten-Kultur einherging, bestand darin, dass Programme in diesen Kreisen zwar gerne weitergegeben, aber nur

selten käuflich erworben wurden: „Hardware must be paid for, but software is something to share. […] Who can afford to do professional work for nothing?“ (Gates 1976). Infolgedessen wurden Softwareprodukte in den frühen 1980er Jahren oft nur noch als Binärdatei ohne les- und veränderbaren Quellcode verkauft. Gleichzeitig erhöhten mehrere Gesetzesnovellen in den USA deren Schutz und Ausschließbarkeit. Als gesellschaftsethische Replik auf diese Schließungsprozesse kündigte der MIT-Mitarbeiter Richard Stallman (1983) an, unter dem rekursiven Akronym GNU (‚GNU's Not Unix') ein freies Betriebssystem entwickeln zu wollen: „I consider that the golden rule requires that if I like a program I must share it with other people who like it. […] So that I can continue to use computers without violating my principles, I have decided to put together a sufficient body of free software […].“

Obgleich der GNU-Kernel per se bis heute nicht für den praktischen Einsatz geeignet ist, erwies sich Stallmans Projekt als Keimzelle für die freie Softwareentwicklung: 1985 gründete sich in seinem Kontext die Free Software Foundation; ab 1988 wurden erste industrielle Großspender wie Sony oder Hewlett-Packard angeworben. Die bedeutsamste Neuerung bestand jedoch in der Definition rechtlich belastbarerer Lizenzmodelle, die wie die 1989 erstmals publizierte General Public License (GPL) erzwingen, dass auch Derivate freier Software stets quelloffen bleiben müssen: „Each time you redistribute 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.“ (FSF 1989). Ab 2001 waren Verstöße gegen die GPL Gegenstand mehrerer Gerichtsverfahren in den USA und Europa (Jaeger 2010), wobei „the court of public opinion“ im Usenet bzw. im Web für die Etablierung der in der GPL angelegten Reziprozitätsprinzipien eine ebenso tragende Rolle gespielt hat (O'Mahony 2003: 1189).

Der Erfolg des GNU-Projektes selbst blieb aufgrund seines Zuschnitts auf kostenintensive Workstations und seiner ideologischen Konnotationen indes begrenzt. Auf beide Problemstellungen bot das Linux-Kernel-Projekt eine Antwort: Linux wurde 1991 durch den Studenten Linus Torvalds als freier Betriebssystemkern für die günstigeren Mikrocomputer vorgestellt und war aus diesem Grund für eine größere Zahl an Entwickler/-innen attraktiv. Zudem zeichnete sich Torvalds (2002) von vornherein durch eine liberalere Haltung als die Free Software Foundation aus: „This world would be a much better place if people had less ideology and a whole lot more 'I do this because it's fun […]'.“ Ein weiterer Grund für das Florieren der Linux-Kernel-Entwicklung bestand in der Etablierung des World Wide Web, wodurch sowohl der Zugriff auf das Projekt als auch dessen Koordination erheblich erleichtert wurde. Dennoch blieb auch Linux in den ersten Jahren ein lediglich in Expertenkreisen bekanntes Vorhaben.

Dies änderte sich mit dem vielrezipierten Buch „The Cathedral and the Bazaar“ (1999), das von dem Softwareentwickler Eric S. Raymond 1997 zunächst als Essay vorgestellt wurde. Seine Kernthese lautete: Während in klassischen Produktionsmodellen der Source Code eines Programms lediglich für fertige Versionen veröffentlicht wird und die Entwicklergruppen hierarchisch organisiert sind (cathedral), sei der Quellcode in Projekten wie Linux oder dem von ihm selbst initiierten Fetchmail stets einsehbar, ihre Gruppen seien horizontal strukturiert und geprägt durch modulare Selbstorganisation ohne zentrales Management (bazaar). Kritische Beobachter stellten allerdings früh fest, dass in beiden Fällen zwar viele Vorschläge aus der Community kamen, die finalen Änderungen aber jeweils nur durch eine Person – Torvalds oder Raymond – freigeben wurden (Bezroukov 1999). Anders formuliert: „The only entity that can really succeed in developing Linux is the entity that is trusted to do the right thing. And as it stands right now, I'm the only person/entity that has that degree of trust.“ (Torvalds 1998: 36).

Mit GNU und Linux bildeten in den 1980/90er Jahren zwei Flaggschiffprojekte freier Software heraus, deren Erfolg durch die elektronische Effektivierung der Kommunikation befördert wurde. In ihrem Kontext bildeten sich rechtliche Instrumente, welche die kollektiven Arbeitsresultate vor Proprietari-

sierung schützen, wie auch informelle Konventionen heraus, deren Anerkennung sich im Web unkomplizierter überprüfen ließ als zuvor. Daneben verfestigten sich erste anschlussfähige Narrative, die freie Softwareentwicklung als einen revolutionären Produktionsmodus ohne Machtasymmetrien beschrieben und zeitweilig ohne weitere Rückfragen sozialwissenschaftlich weiterverarbeitet wurden (zum Beispiel Benkler 2002, 2013; Tapscott, Williams 2006).

Open Source als Entwicklungsmethode

Im nachfolgenden Jahrzehnt konnte sich die quelloffene Entwicklung als Methode zunehmend in der Softwarebranche etablieren, was sich neben der fortgesetzten Verbreitung des Internets primär auf drei Dynamiken zurückführen lässt.

Zum ersten lagerte eine wachsende Zahl an IT-Firmen die Entwicklung von Software in den quelloffenen Bereich aus, darunter Netscape Communications als ein besonders aufsehenerregender Fall: Nachdem es absehbar erschien, dass Microsoft den Netscape Navigator durch den in Windows integrierten Internet Explorer aus dem Markt drängen würde, kündigte das Unternehmen Anfang 1998 an, den Code seines Browsers in das quelloffene Projekt Mozilla zu überführen, das bis 2003 durch Netscape/AOL finanziell wie auch personell unterstützt wurde. Dabei ging es Netscape (1998) in erster Linie um die Erschließung neuer Kundenkreise „by [...] building a community that addresses markets and needs we can't address on our own [...]."

Zum zweiten kam 1998 eine Gruppe um Eric Raymond zu dem Schluss, dass der politisch belegte Begriff ‚Free Software' für die Verbreitung quelloffener Methoden in kommerziellen Kontexten hinderlich sein könnte, schuf das neue Label ‚Open Source', das die Überlegenheit des Entwicklungsmodells betonen sowie gesellschaftsethische Aspekte ausblenden sollte, und gründete mit Hilfe von Szeneprotagonist/-innen wie Tim O'Reilly die Open Source Initiative. Allerdings unterstützt die Free Software Foundation diese Kursänderung bis heute nicht: „For the Open Source movement, non-free software is a suboptimal solution. For the Free Software movement, non-free software is a social problem and free software is the solution." (Stallman 2001: 57).

Zu den Vermarktungsbemühungen der Open Source Initiative kamen *zum dritten* die durch den allgemeinen Dotcom-Boom beförderten Börsenerfolge einiger ‚open source companies' im Jahr 1999 hinzu, darunter zuvorderst die Hardwareanbieter VA Linux und Cobalt Networks sowie der Linux-Distributor Red Hat. Deren Börsengänge gehörten zu den erfolgreichsten Debüts aller Zeiten und erregten eine entsprechend große mediale Aufmerksamkeit. Kurz darauf beschrieb etwa der Spiegel (33/1999: 78) Linux als „ernsthafte Konkurrenz zum Microsoft-Monopol". Damit war ‚Open Source' als Schlagwort im öffentlichen Bewusstsein angekommen.

Tabelle 1: Meistgenutzte quelloffene Softwarelizenzen

	2015 (%)	2010 (%)	Ausrichtung	Publikation
GNU Public License 2.0	23	47	strongly protective	1991
MIT License (X11)	23	6	permissive	1988
Apache License 2.0	16	4	permissive	2004
GNU Public License 3.0	9	6	strongly protective	2007

BSD License 2.0 (3-clause)	6	6	permissive	1999
Artistic License 1 / 2	4	9	permissive	2000 / 2006
GNU Lesser GPL 2.1 / 3.0	6	9	weakly protective	1999 / 2007
Microsoft Public License	2	2	permissive	2007

Quelle: Black Duck Knowledgebase.

Diese ineinandergreifenden Entwicklungsstränge führten im Verbund mit der weiteren Ausweitung des IT-Marktes zu einem raschen Wachstum freier Softwareprojekte: Während 1999 einige hundert quelloffene Vorhaben existierten, waren 2015 auf den Plattformen GitHub und SourceForge mehrere Millionen Repositorien zu finden. Angesichts dieser steigenden Projektanzahl und der Definition neuer Lizenzmodelle unterlag die Open-Source-Entwicklung in den 2000er Jahren einer deutlichen Diversifizierung (Tabelle 1): Neben strenge ‚Copyleft'-Lizenzen, die garantieren, dass auch Derivate freier Software stets unter gleichen Bedingungen distribuiert werden (*strongly protective*), sind Lizenzen getreten, welche die Einbindung freier Software in proprietäre Produkte gestatten, sofern ebendiese Komponenten quelloffen bleiben (*weakly protective*), oder wieder die Publikation von Fortentwicklungen unter restriktiveren Vorzeichen erlauben (*permissive*). Diese Vielfalt erweitert die strategischen Optionen für kommerzielle Stakeholder: Google etwa entschied sich im Falle des Betriebssystems Android von vornherein dazu, den Großteil des Codes unter permissive Lizenz zu stellen.

Daneben lässt sich in zweierlei Hinsicht eine Korporatisierung von Open-Source-Projekten beobachten: Zum einen werden zentrale Vorhaben wie der Linux Kernel heute primär durch Unternehmensspenden finanziert oder operieren wie Android unter der expliziten Federführung kommerzieller Anbieter. Zum anderen speist sich die Entwicklerbasis großer Projekte verstärkt aus Firmenkontexten: Kolassa et al. (2014) kommen für den Linux Kernel und 5000 weitere marktrelevante Vorhaben zu dem Schluss, dass 2000 bis 2011 über 50 Prozent aller Beiträge in der westlichen Kernarbeitszeit geleistet wurden; die Linux Foundation (Corbet et al. 2015) beobachtet, dass der Anteil unabhängiger Programmierer/-innen an der Kernel-Entwicklung (2009: 18 Prozent; 2014: 12 Prozent) stetig abnimmt. Allein Entwickler/-innen aus den Unternehmen Intel, Red Hat, Samsung und IBM waren 2015 für über 25 Prozent aller Aktualisierungen verantwortlich.

Zwar lassen sich nach wie vor Projekte wie die Linux-Distributionen Arch oder Parabola finden, die sich an den generischen Maximen freier Software ausrichten. In viele langfristig aktive Open-Source-Projekte sind heute allerdings etablierte Unternehmen involviert, die sich auf die dortigen Rahmenbedingungen eingestellt haben und diese Kontexte nutzen, um außerhalb formaler Kooperationsbeziehungen mit externen Programmierer/-innen zu kollaborieren. Insofern beschreibt der Blogger Bulajewski (2011) das Bild von Open-Source-Projekten als Gemeinschaften „of volunteer programmers collaborating together in a gift economy" zurecht als Illusion.

Open Source als Innovationsstrategie

Insbesondere für das Segment der Unternehmenssoftware, in dem über 80 Prozent der weltweiten Erlöse mit Software und Services generiert werden, diagnostizieren Marktforscher inzwischen „a widespread use of open-source technology in mission-critical IT portfolios" (Driver 2014). Überdies kann

quelloffenen Architekturen in vielen Bereichen der basalen informationstechnischen Infrastrukturen seit mehreren Jahren Marktführerschaft zugesprochen werden (Forrester 2014). Vor diesem Hintergrund sind heute die meisten großen IT-Konzerne in Open-Source-Projekte involviert.

Microsoft – das Unternehmen, das Open Source lange als „intellectual-property destroyer" einstufte (Computerworld 3/2001: 78) – hat 2012 die Tochterfirma MS Open Technologies lanciert, gehörte zwischenzeitig zu den Top-Beiträgern im Linux-Kernel-Projekt (Corbet et al. 2012) und hat ab 2014 das Framework .NET sowie weitere Komponenten unter freie Lizenz gestellt – „to achieve a strategic objective, such as promoting industry standards, advancing interoperability, or attracting and enabling our external development community" (Microsoft 2015: 13). Welche genauen Anteile ihrer Entwicklungsausgaben marktdominante Konzerne wie Microsoft in Open-Source-Projekte investieren, lässt sich freilich kaum gesondert abschätzen, da quelloffene Elemente inzwischen für viele herstellerspezifische Produkte von elementarer Bedeutung sind. *Apples* Betriebssystempakete OS X und iOS etwa basieren auf dem freien unixoiden Kernel Darwin und tragen über 200 weitere Open-Source-Komponenten mit sich.

IBM investierte bereits zur Jahrtausendwende mehrere 100 Mio. US-Dollar in Linux-Entwicklungsprogramme, um Microsofts Dominanz im Enterprise-Bereich entgegenzusteuern und um ein Servicegeschäft um proprietäre Technologien im Verbund mit quelloffener Software aufzubauen (Capek et al. 2005). Heute ist IBM in über 100 Open-Source-Projekte involviert, darunter die Cloud-Computing-Plattform OpenStack, an deren Entwicklung auch *Intel* und *Hewlett-Packard* beteiligt sind. Ihr Involvement resultiert jedoch ebenfalls nicht aus Idealismus, sondern aus unternehmerischem Kalkül: „Such actions are comparable to giving away the razor (the code) to sell more razor blades (the related consulting services […])." (Lerner 2012: 43). Aus ähnlichen Gründen beteiligen sich Unternehmen wie *Oracle*, *SAP* oder *Adobe* an quelloffenen Projekten. Vor allen Dingen kleineren Firmen dient ein Open-Source-Involvement überdies als „marketing tool to increase brand recognition" (Dahlander, Magnusson 2008: 638).

Eine spezielle Variante korporativen Open-Source-Engagements stellt die Entwicklung von Android durch die Open Handset Alliance dar: In der Literatur oft in eine Linie mit dem Linux Kernel gestellt (Herstatt, Ehls 2015), wird das Projekt de facto allein von *Google* gesteuert: „Because it fully controls the development of the OS, Google can determine the technological specifications to which Android partners must abide." (Spreeuwenberg, Poell 2012). Mit der Lancierung von Android ging es Google (2015) mit offenkundigem Erfolg vorrangig darum, den nahtlosen Zugriff auf eigene Dienste auf möglichst vielen Geräten zu ermöglichen: Während Google 2007 rund 99 Prozent seines Jahresumsatzes (16,6 Mrd. USD) mit Werbung generierte, war der Verkauf digitaler Inhalte 2014 für 11 Prozent des Umsatzes (66 Mrd. USD) verantwortlich.

Darüber hinaus haben sich Ende der 1990er Jahre eine Reihe ‚open source companies' herausgebildet, die ihr Kernprodukt – den Softwarecode – kostenfrei abgeben und mit Supportleistungen ein Geschäft aufbauen wollten. Mit Ausnahme des Linux-Distributors *Red Hat* (Umsatz 2014: 1,8 Mrd. USD), der früh mit dominanten Hardwareanbietern kooperiert hat, sind die meisten dieser im Fahrwasser des New Economy Hypes lancierten Firmen jedoch rasch wieder eingegangen. Zwar sind im Open-Source-Umfeld zuletzt erneut einige Startups entstanden (zum Beispiel Hortonworks); in ihrer Außendarstellung verzichten diese Unternehmen aber in der Regel auf ‚Open Source' als primäres Differenzierungsmerkmal und zeichnen sich durch eine nur noch geringe Verbundenheit mit Stallmans originären Reziprozitätsidealen aus (Bergquist et al. 2012). Vice versa sind es heute vor allem Konzerne wie IBM oder Microsoft, die in ihrer Öffentlichkeitsarbeit auf ausgewählte Maximen freier Software verweisen.

Imagepflege ist allerdings nur einer der Gründe, warum große IT-Firmen als Sponsoren für ein breites Portfolio an quelloffenen Entwicklungsvorhaben auftreten. In vielen Fällen eröffnet ein finanzielles Engagement den Unternehmen zudem die Möglichkeit, deren Ausrichtung im Sinne ihrer Partikularinteressen mitzugestalten. Eine exemplarische Zusammenschau aktiver Open-Source-Projekte zeigt, dass viele Communitys nicht mehr ohne die Unterstützung großer IT-Konzerne auskommen (Tabelle 2): Korporativ initiierte Projekte wie Android stehen naturgemäß am eindeutigsten unter der finanziellen Ägide eines oder mehrerer Unternehmen; ebenso werden Infrastrukturvorhaben wie der Apache HTTP Server heute primär durch korporative Akteure getragen, wobei große Sponsoren zumeist einen Sitz im Steuerungsrat der jeweiligen Stiftung erhalten. Aber auch gesellschaftsethisch fundierte Communitys rekurrieren auf Unternehmensspenden: Die Free Software Foundation beispielsweise generierte 2013 über 80 Prozent ihres Umsatzes (1,2 Mio. USD) durch korporative Zuwendungen.

Tabelle 2: Populäre Projekte auf Open Hub (2015)

Projekt	Commits*	Dachorganisation	Primäre Finanzierungsquelle
Android	81.119	Google, Open Handset Alliance	
Linux Kernel	65.402	Linux Foundation	Mitglieder (u.a. HP, Intel, IBM)
OpenStack	62.370	OpenStack Foundation	Mitglieder (u.a. HP, IBM, Red Hat)
Mozilla Firefox	60.226	Mozilla Foundation	Provisionen (2013: 95% Google)
KDE	36.220	KDE e.V.	Patronagen (u.a. Google, SUSE)
Debian Linux	26.374	Debian Project	Spenden, Partner (u.a. HP, 1&1)
LibreOffice	21.384	Document Foundation	Spenden (u.a. Google, Red Hat)
GNU CC	8.382	Free Soft. Foundation	Patronagen (u.a. Google, IBM)
Eclipse	3.515	Eclipse Foundation	Mitglieder (u.a. Google, IBM)
Apache HTTP Server	2.356	Apache Foundation	Spenden (u.a. Google, Microsoft)

Quelle: Open Hub, Jahresberichte der Stiftungen; *Aktualisierungen 5/2014–5/2015

Viele große Open-Source-Vorhaben stehen heute also in einem finanziellen Wechselverhältnis mit führenden IT-Firmen, die im Rahmen ihrer Markt- und Innovationsstrategien gezielt in spezifische Projektportfolios investieren. Kombiniert mit ihrem Involvement in die konkrete Code-Entwicklung sichern sich die jeweiligen Unternehmen so einen nicht zu unterschätzenden Einfluss auf relevante Entwicklungsvorhaben und tragen zugleich zu einer Erhöhung der Planungssicherheit in den Communitys bei.

Spielarten quelloffener Softwareprojekte

In den letzten 20 Jahren ist die Open-Source-Entwicklung insofern zu einem integralen Bestandteil der Softwareindustrie geworden; sie hat dabei allerdings ihre Formatierung als Gegenentwurf zur kommerziellen Herstellung weithin verloren. Derzeit lassen sich vier idealtypische Varianten von Open-Source-Projekten unterscheiden (Abbildung 2).

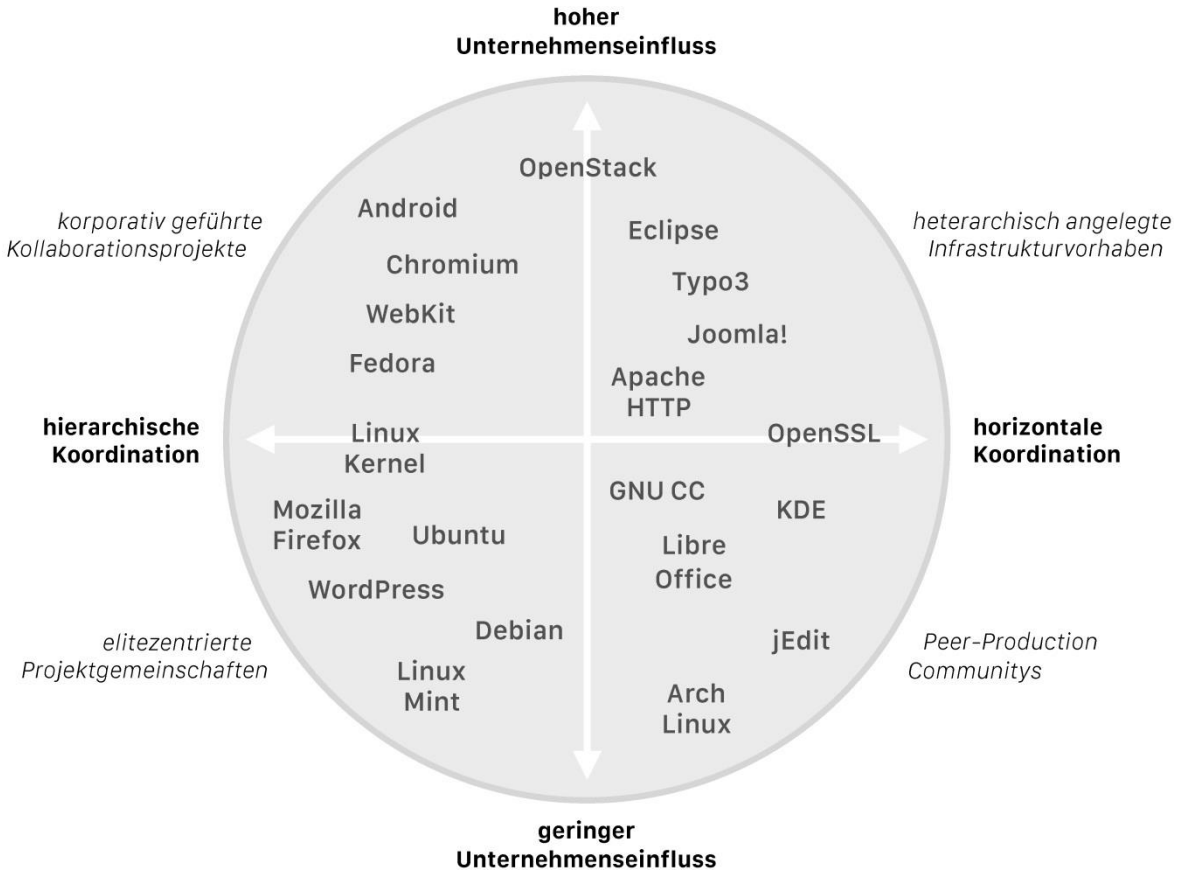

Abbildung 2: Idealtypische Varianten von Open-Source-Projekten

Korporativ geführte Kollaborationsprojekte zeichnen sich durch prägnante Hierarchisierungen auf Arbeitsebene aus. Im Falle von Android und WebKit liegt die strategische Kontrolle eindeutig bei Google bzw. Apple; im Falle von OpenStack haben zentrale Sponsoren ebenfalls einen steuernden Einfluss. Die Projektgemeinschaften bestehen vorrangig aus angestellten Entwickler/-innen, die problemzentriert auf individueller Ebene oder als explizite Unternehmensvertreter/-innen zusammenarbeiten. Eine solche Kollaboration in Open-Source-Projekten trägt zur Überwindung zweier ‚knowledge sharing dilemmas' bei: Zum einen verhindern quelloffene Lizenzen eine Einzelaneignung des Codes; zum anderen stellen sie sich Trittbrettfahrer/-innen entgegen, da stets nachvollziehbar bleibt, welche Firmen sich auf welche Elemente stützen und inwiefern sie an deren Entwicklung teilhaben (Henkel et al. 2014; Gonzalez-Barahona, Robles 2013).

Auch *heterarchisch angelegte Infrastrukturvorhaben* sind eng mit korporativen Kontexten verwoben: Entweder sie fußen (wie Eclipse) auf vormals herstellerspezifischen Architekturen oder sie waren (wie Apache HTTP) durch ein rasches organisches Wachstum gekennzeichnet, da sie Lösungen für zuvor nicht probat adressierte Bereiche boten, und deshalb früh für Unternehmen interessant waren. Heute werden Infrastrukturvorhaben primär durch das Engagement mittlerer und großer IT-Firmen getragen; ihre Communitys werden jedoch nicht durch korporative Kernzirkel angeleitet, sondern operieren unter dem Dach gemeinnütziger Stiftungen. Funktionsträger/-innen werden meist meritokratisch designiert; allerdings können sich von Unternehmen dafür freigestellte Entwickler/-innen in der Regel

intensiver als Freizeitprogrammierer/-innen in die Projekte einbringen und so eher Entscheidungspositionen erlangen (Greenstein, Nagle 2014; Wasserman 2013).

Elitezentrierte Projektgemeinschaften stehen ebenfalls nicht unter der direkten Kontrolle eines gewerblichen Akteurs; auch sie stützen sich aber auf die Beiträge firmenaffiliierter Entwickler/-innen. Ihre Koordination erfolgt entlang von Entscheidungspyramiden, an deren Spitze ihr Gründer als ‚benevolent dictator' (zum Beispiel Linux Kernel), ein langfristig installiertes Führungsteam (zum Beispiel Mozilla) oder ein gewählte/-r Projektleiter/-in (zum Beispiel Debian) steht. Kritische Entschlüsse werden durch diese Steuerungselite gefasst, wobei genannte Projekte auch daneben über Entscheidungsträger/-innen verfügen, die vereinheitlichend tätig werden. In Debian und Mozilla sind die Projektrichtlinien formal fixiert worden; im Falle von Mint und dem Linux Kernel haben sich entlang des Führungsstils ihrer Gründer hingegen lediglich „opaque governing norms [...] without a clear mechanism of accountability" herausgebildet, die in Konfliktfällen der proklamierten Offenheit der Projekte entgegenlaufen können (Kreiss et al. 2011: 252; Stamelos 2014).

Peer-Production-Communitys dienen laut Eigendefinition der marktunabhängigen und gleichberechtigten Kollaboration unter freiwilligen Entwicklern; sie bilden allerdings – wie sich an KDE, GNU oder LibreOffice zeigen lässt – ab einer gewissen Größe bzw. Marktrelevanz gleichermaßen abgestufte Führungsstrukturen aus und verfügen über einen Pool an korporativen Stakeholdern. Intrinsisch verortete Communitys wie Arch hingegen richten ihre Produkte auf spezifische Anspruchsgruppen aus, werden durch kleine Entwicklerteams getragen und konnten daher bis dato auf die Ausbildung ausgeprägter sozialer Strukturierungen verzichten. Sobald aber die Gemeinschaft wächst und sich ihre Interaktionen mit externen Akteuren intensivieren, werden offenkundig trotz aller technischen Effektivierungen auch in gesellschaftsethisch fundierten Vorhaben ‚kathedralartige' Koordinationsmuster notwendig (Corbet 2015; Alleyne 2011).

Der gemeinsame Nenner aller genannten Spielarten besteht in den dahinterliegenden quelloffenen Lizenzmodellen, die ihre Produkte wirksam vor direkter Proprietarisierung schützen. Mit „Rebel Code" (Moody 2002) hat all dies indes nicht mehr viel gemein: Die Verschränkungen mit marktlichen Kontexten sind oft ausgeprägt; trotz der technisch erweiterten Austauschmöglichkeiten bilden sich regelmäßig hierarchische Entscheidungsmuster aus; entgegen dem Eindruck, „that organizations [...] don't matter as much as they used to" (Suddaby 2013: 1009), verlieren korporative Akteure in Open-Source-Projekten keineswegs an Bedeutung, sondern bleiben als deren Initiatoren, Financiers und Arbeitgeber der involvierten Programmierer/-innen prominent im Spiel. Ferner verfügen viele unabhängige Projekte über assoziierte gemeinnützige Organisationen, die als adressierbare Dachentitäten fungieren (Ahrne, Brunsson 2011).

Open Source als soziotechnisch verstetigte kollektive Invention

Das nach wie vor einschlägige Narrativ von der quelloffenen Softwareentwicklung als revolutionäres Produktionsmodell, das auf selbstgesteuerter Kollaboration fußt und eingespielten sozioökonomischen Koordinationsformen wie ‚Markt' oder ‚Hierarchie' überlegen ist, lässt sich in dieser Radikalität insofern in den allgemeinen Strom der übersteigerten Entdifferenzierungserwartungen in der digitalen Moderne einordnen (Dickel, Schrape 2015): Ebenso mit Blick auf andere Spielarten kollektiven Handelns (zum Beispiel in sozialen Bewegungen), der nutzerzentrierten Produktion (zum Beispiel in journalistischen Kontexten) oder den Strukturwandel der Öffentlichkeit zeigt sich auch hinsichtlich Open-Source-Communitys, dass die Onlinetechniken zwar die Kommunikation effektivieren und er-

weiterte Möglichkeiten zur Kollaboration aufgeschlossen haben, aber grundlegende gesellschaftliche Strukturierungen und Rollendifferenzierungen dadurch keineswegs überschrieben werden (Dolata, Schrape 2016). Technische Infrastrukturen können die Koordination vereinfachen und die Arbeitsorganisation unterstützen; die sozialen Ordnungsleistungen, die einen situativen Zusammenschluss in eine festgefügte Projektgemeinschaft überführen, können sie aber nicht ersetzen.

Wie auch die Resultate früherer Episoden der ‚collective invention' (Allen 1983; Osterloh, Rota 2007; Boudreau, Lakhani 2015), in denen Unternehmen ihre Wissensbestände zu Beginn von Innovationsprozessen auch abseits formaler Kooperationen geteilt haben, um kumulativen Fortschritt zu ermöglichen – zum Beispiel auf dem Feld der Hochofentechnologie (1850–1880) oder in der Entwicklung von Flachbildschirmen (1969–1989), erfuhren freie Softwareprojekte zudem rasche Korporatisierung und Kommerzialisierung, sobald sie für den Markt Relevanz erlangt hatten. Von den von Robert Allen und Kollegen diskutierten Beispielen hebt sich das Open-Source-Modell allerdings durch zwei entscheidende Merkmale ab: Zum einen unterliegt der Austausch in onlinebasierten Communitys keiner geografischen Begrenzung mehr; zum anderen haben ab den 1980er Jahren neuartige lizenzrechtliche Konstruktionen, die eine grundsätzliche Quelloffenheit auch für Derivate freier Software garantieren, und die im Web erleichterte Verbreitung informeller Regeln dazu beigetragen, dass Open-Source-Projekte im Gegensatz zu früheren Ausprägungen kollektiver Invention nach der Herausbildung eines dominanten Designs und dessen Verwertung überlebensfähig bleiben.

Insoweit waren ab den 1980er Jahren nicht nur die erweiterten Formen der elektronischen Vernetzung, sondern ebenso die Kristallisation übergreifend akzeptierter Werte und Arbeitskonventionen sowie vor allem anderen die Definition rechtlich tragfähiger Lizenzmodelle für die Stabilisation und den anhaltenden Erfolg quelloffener Entwicklungsvorhaben von herausgehobener Bedeutung. ‚Copyleft'-Lizenzen und ihre Ableitungen haben im Verbund mit den koordinationserleichternden Eigenschaften der Online-Technologien den soziotechnischen Rahmen für eine auf Dauer gestellte Form kollektiver Invention aufgespannt, die in den 1980er und 1990er Jahren zunächst in subversiven, vom allgemeinen Markt abgekoppelten Nischen Anwendung fand, nach der Jahrtausendwende zunehmend von der allgemeinen Softwareindustrie adaptiert wurde und heute zu einem Baustein der Innovationsstrategien aller etablierten Anbieter geworden ist, welche so ihre internen Forschungs- und Entwicklungsaktivitäten um eingegrenzte Kollaborationskontexte mit anderen Marktteilnehmern ergänzen.

Quelloffene Softwarelizenzen sind dementsprechend nicht einfach nur eine vorübergehende „form of institutional jiujitsu" (Benkler 2002: 446) vor einer erhofften Totalauflösung geistiger Eigentumsrechte (vgl. dazu als Überblick: Coleman 2013: 185–205), sondern nach wie vor das strukturelle Alleinstellungsmerkmal und die elementare Geschäftsgrundlage von Open-Source-Communitys, die einen erwartungssicheren Rahmen für den punktuellen Wissensaustausch sowie die projektorientierte Zusammenarbeit zwischen Einzelentwicklern wie auch Unternehmen bieten, auf rechtlich abgesicherte Weise kumulativen Fortschritt ermöglichen und ob ihrer flexibilisierenden Eigenheiten inzwischen nicht mehr nur in der Softwareproduktion, sondern auch auf angrenzenden Feldern wie etwa der Hardware-Entwicklung Anwendung finden. Offener Quellcode mündet allerdings nicht zwangsläufig in transparenteren Koordinationsmustern als in anderen Arbeitszusammenhängen oder in einer Disintermediation langfristig kristallisierter Ressourcen- und Einflussverteilungen. Insofern stehen neuartige soziotechnische Kollaborationsmuster in Open-Source-Projekten und existente institutionelle Arrangements wie ‚Markt' und ‚Hierarchie' weniger in einem rivalisierenden als in einem sich überlagernden und komplementären Verhältnis zueinander.

Literatur

Ahrne, G., Brunsson, N. 2011: Organization Outside Organizations: The Significance of Partial Organization. Organization, 18. Jg., Heft 1, 83–104.

Allen, R. 1983: Collective Invention. Journal of Economic Behavior & Organization, 4. Jg., Heft 1, 1–24.

Alleyne, B. 2011: Challenging Code: A Sociological Reading of the KDE Free Software Project. Sociology, 45. Jg., Heft 3, 496–511.

Benkler, Y. 2002: Coase's Penguin, or, Linux and ‚The Nature of the Firm'. Yale Law Journal, 112. Jg., 369–446.

Benkler, Y. 2013: Practical Anarchism, Peer Mutualism, Market Power, and the Fallible State. Politics & Society, 41. Jg., Heft 2, 213–251.

Benkler, Y., Nissenbaum, H. 2006: Commons-based Peer Production and Virtue. Journal of Political Philosophy, 14. Jg., Heft 4, 394–419.

Bennett, W. L., Segerberg, A., Walker, S. 2014: Organization in the Crowd: Peer Production in Large-scale Networked Protests. Information, Communication & Society, 17. Jg., Heft 2, 232–260.

Bergquist, M., Ljungberg, J., Rolandsson, B. 2012: Justifying the Value of Open Source. ECIS 2012/122.

Bezroukov, N. 1999: A Second Look at the Cathedral and the Bazaar. First Monday, 4. Jg., Heft 12. http://firstmonday.org/article/view/708/618 (12/2016).

Boudreau, K., Lakhani, K. 2015: ‚Open' Disclosure of Innovations, Incentives and Follow-on Reuse. Research Policy, 44. Jg., Heft 1, 4–19.

Bulajewski, M. 2011: The Peer Production Illusion, Part I. MrTeaCup vom 19.11.2011. http://www.mrteacup.org/post/peer-production-illusion-part-1.html (letzter Aufruf 01.12.2016).

Capek, P., Frank, S., Gerdt, S., Shields, D. 2005: A History of IBM's Open-Source Involvement and Strategy. IBM Systems Journal, 44. Jg., Heft 2, 249–257.

Coleman, G. 2013: Coding Freedom. The Ethics and Aesthetics of Hacking. Princeton: PUP.

Corbet, J. 2015: Development Activity in LibreOffice and OpenOffice. LWN.net vom 25.3.2015. https://lwn.net/Articles/637735/ (letzter Aufruf 01.12.2016).

Corbet, J., Kroah-Hartman, G., McPherson, A. 2009–2015: Linux Kernel Development Report. San Francisco: The Linux Foundation.

Dahlander, L., Magnusson, M. 2008: How do Firms Make Use of Open Source Communities? Long Range Planning, 41. Jg., Heft 6, 629–649.

Dickel, S., Schrape, J.-F. 2015: Dezentralisierung, Demokratisierung, Emanzipation. Zur Architektur des digitalen Technikutopismus. Leviathan, 43. Jg., Heft 3, 442–463.

Dolata, U., Schrape, J.-F. 2016: Masses, Crowds, Communities, Movements: Collective Action in the Internet Age. Social Movement Studies, 15. Jg., Heft 1, 1–18.

Driver, M. 2014: Within the Enterprise, Open Source Must Coexist in a Hybrid IT Portfolio. Research Report vom 23.8.2014. Stamford: Gartner.

Forrester Research Inc. 2014: Enterprise and SMB Software Survey, North America and Europe. Market Report. Cambridge: Forrester.

Free Software Foundation. 1989: GNU General Public License (GPL) Version 1.0. Boston: FSF.

Gates, B. 1976: An Open Letter to Hobbyists. Computer Notes, 1. Jg., Heft 9, 3.

Gonzalez-Barahona, J., Robles, G. 2013: Trends in Free, Libre, Open Source Software Communities. Information Technology, 55. Jg., Heft 5, 173–180.

Google Inc. 2015: Annual Report. Form 10-K. https://investor.google.com (12/2016).

Greenstein, S., Nagle, F. 2014: Digital Dark Matter and the Economic Contribution of Apache. Research Policy, 43. Jg., 623–631.

Gulley, N., Lakhani, K. 2010: The Determinants of Individual Performance and Collective Value in Private-collective Software Innovation. Harvard Business School TOMU Working Paper 10/065.

Henkel, J., Schöberl, S., Alexy, O. 2014: The Emergence of Openness: How and Why Firms adopt Selective Revealing in Open Innovation. Research Policy, 43. Jg., Heft 5, 879–890.

Herstatt, C., Ehls, D. 2015: Open Source Innovation. New York: Routledge.

Jaeger, T. 2010: Enforcement of the GNU GPL in Germany and Europe. Journal of Intellectual Property, Information Technology and E-Commerce Law, 1. Jg., Heft1, 34–39.

Kolassa, C., Riehle, D., Riemer, P., Schmidt, M. 2014: Paid vs. Volunteer Work in Open Source. Proceedings 47th Hawaii Conference on System Sciences, 3286–3295.

Kreiss, D., Finn, M., Turner, F. 2011: The Limits of Peer Production: Some Reminders from Max Weber for the Network Society. New Media & Society, 13. Jg., Heft 2, 243–259.

Lakhani, K., Hippel, E. v. 2003: How Open Source Software Works. Research Policy, 32. Jg., Heft 6, 923–943.

Lerner, J. 2012: The Architecture of Innovation. Boston: Harvard Business Press.

Lessig, L. 1999: Open Code and Open Societies. Chicago Kent Law Review, 74. Jg., 1405–1420.

Levy, S. 1984: Hackers: Heroes of the Computer Revolution. Garden City: Anchor Press.

Microsoft Inc. 2015: 2014 Annual Report. http://www.microsoft.com/investor/reports/ (letzter Aufruf 12/2016).

Moody, G. 2002: Rebel Code. New York: Basic.

Netscape Communications. 1998: Netscape Announces Mozilla.org. Press Release 23.2.1998.

O'Mahony, S. 2003: Guarding the Commons. How Community Managed Software Projects Protect their Work. Research Policy, 32. Jg., Heft 7, 1179–1198.

Osterloh, M., Rota, S. 2007: Open Source Software Development: Just Another Case of Collective Invention? Research Policy, 36. Jg., Heft 2, 157–171.

Raymond, E. S. 1999: The Cathedral and the Bazaar. Sebastopol: O'Reilly.

Rifkin, J. 2014: The Zero Marginal Cost Society. New York: Palgrave Macmillan.

Stamelos, I. 2014: Management and Coordination of Free/Open Source Projects. In G. Ruhe, C. Wohlin (Hg.), Software Project Management in a Changing World. New York: Springer, 321–341.

Schrape, J.-F. 2016: Open-Source-Projekte als Utopie, Methode und Innovationsstrategie. Glückstadt: VWH.

Spreeuwenberg, K. /Poell, T. 2012: Android and the Political Economy of the Mobile Internet. First Monday, 17. Jg., Heft 7. http://dx.doi.org/10.5210/fm.v17i7.4050 (112/2016).

Stallman, R. 2002: Free Software, Free Society. Boston: GNU Press.

Stallman, R. 1983: New UNIX Implementation. http://bit.ly/1DSDoXW (letzer Aufruf 12/2016).

Suddaby, R. 2013: Book Review: The Janus Face of Commercial Open Source Software Communities. Organization Studies, 34. Jg., Heft 7, 1009–1011.

Tapscott, D./Williams, A. D. 2006: Wikinomics. New York: Portfolio.

Torvalds, L. 1998: LINUX Manifesto. Interview. Boot Magazine, Jg. 1998, Heft 7–8, 32–37.

Torvalds, L. 2002: Re: [PATCH] Remove Bitkeeper Documentation from Linux Tree. Linux Kernel Mailinglist vom 20.4.2002. http://lwn.net/2002/0425/a/ideology-sucks.php3 (letzter Aufruf12/2016).

Wasserman, A. 2013: Community and Commercial Strategies in Open Source Software. Information Technology 55. Jg., Heft 5, 181–188.