Read The Fucking Manual

In- und Exklusion nicht-technischer User in Open Source Software Communities[1]

Daniel Guagnin

Beitrag zur Veranstaltung »Ambivalenzen der Kommunikation und Kollektivität im Internet II« der Sektion Wissenschafts- und Technikforschung

Im Rahmen des Themas Ambivalenzen der Kommunikation und Kollektivität im Internet[2] beschäftigt sich der folgende Beitrag mit dem Spannungsverhältnis zwischen der grundsätzlichen Öffnung der Gestaltung von Software durch ihre Nutzer/-innen[3] und der produktiven Schließung der Softwareentwicklung.[4]

Zwischen Advocacy und Hedonismus

Die zentrale Forderung der Freie Software Bewegung ist, dass jede Nutzerin den Software-Quellcode studieren, verändern, verteilen und für jeglichen Zweck nutzen darf. Dadurch wird die klassische Trennung von Experten und Laien als Entwicklerinnen und Nutzerinnen von Technik aufgehoben. An die Stelle einer strikten a priori Trennung treten *epistemische Regime*, die die Wissensproduktion, und somit die Softwareentwicklung regulieren.

In Anlehnung an die Problembeschreibung der Legitimitätskrise der wissenschaftlichen Expertise in Harry Collins und Robert Evans' (2002) Aufsatz *Third Wave of Science Studies* betrachte ich die Kritik der Free Software Foundation an proprietärer Software als Folge eines Legitimitätsproblems der Software-Hersteller. Durch Freie-Software-Lizenzen, die allen Nutzerinnen Modifikationsrechte einräumen, ent-

[1] Ich verwende an dieser Stelle den Begriff Open Source Software aufgrund des Wiedererkennungswertes. Korrekterweise sollte es Free (Libre) Open Source Software heißen, um den libertären Charakter der damit verbundenen sozialen Bewegung einzufangen.

[2] Das Papier basiert auf einem Vortrag zur so betitelten Sitzung der Sektion Wissenschafts- und Technikforschung auf dem Kongress der Deutschen Gesellschaft für Soziologie in Bamberg 2016.

[3] Aus Gründen der Leserlichkeit werden im Folgenden die männliche und die weibliche Form abwechselnd verwendet, an keiner Stelle ist ein bestimmtes Geschlecht gemeint.

[4] Der vorliegende Beitrag basiert auf vorläufigen Ergebnissen meines Dissertationsprojektes und ist somit als Working Paper zu verstehen.

steht folglich ein entsprechendes Erweiterungsproblem im Sinne der Erweiterung des Kreises der Beitragenden, dies wirft die Frage auf, in welchem Maße Nutzer in die Gestaltung der Software involviert werden sollten und *können*.

Daher gehe ich hier der Frage nach, wie Freie (Libre) /Open Source Software (FLOSS) Gemeinschaften vor dem Hintergrund der grundsätzlichen Öffnung für eine Beteiligung aller Nutzer eine produktive Schließung der Entwicklung organisieren. Und ferner, welche Konsequenzen dies für das gemeinsame Wissensprodukt hat.

Durch die Öffnung der Entwicklung de jure einerseits und die praktische Schließung andererseits ergibt sich ein grundsätzliches Spannungsfeld zwischen Inklusion und Exklusion von Laien in FLOSS Communities. Zur Veranschaulichung dieser Spannung greife ich drei Zitate aus Diskussionen in Mailinglisten verschiedener GNU/Linux Communities heraus. Dabei geht es mir um die *Kontrastierung* verschiedener Praktiken des Umgangs mit Nichtwissen, die bei der Aushandlung der Legitimität des Akronyms RTFM – üblicherweise gelesen als *Read The Fucking Manual* – sichtbar werden. Dieses Akronym findet manchmal Verwendung als Antwort auf eine vermeintlich zu triviale Frage, die nach Ansicht der Antwortenden durch einen kurzen Blick in einschlägige Dokumentationsquellen von den Fragenden selbst hätte beantwortet werden können.

> **Ubuntu Linux (1):** [S]aying RTFM [...] is really not cool, nor following the CoC (Code of Conduct, Anm. d. Verf.)[5]
>
> **Debian Linux (2):** When the code is public, RTFM is the proper answer. One might add: document it properly afterwards[6]
>
> **Arch Linux (3):** RTFM helps the noob[7]

Während die Aussage im ersten Fall auf einen Code of Conduct verweist, gegen den die Antwort RTFM zu verstoßen scheint, wird RTFM im zweiten Fall als eine Aufforderung zur aktiven Partizipation an der als unzureichend erachteten Dokumentation gerahmt. Dies steht vor dem Hintergrund der Offenheit des Software-Quellcodes[8], der als letzte Instanz nötigenfalls die grundlegenden Informationen liefern soll. Stillschweigend vorausgesetzt ist dabei, die Fähigkeit Software-Quellcode lesen zu können. Im dritten Fall wird RTFM als ein hilfreicher Hinweis verstanden, der dem Noob (also dem Neuling bzw. Laien) Defizite in seinem Wissen aufzeigt.

Hier tritt der Widerspruch zwischen einem universalistischen Anspruch der Freie Software Bewegung, der eine größtmögliche Offenheit für Laien anstrebt, und einem eher selbstzentrierten Modus des „scratching your own itch" zu Tage.

Mit anderen Worten offenbart sich hier ein Spannungsfeld zwischen Advocacy als Moment der Offenheit auf der einen, und einem auf sich selbst gerichteten Hedonismus[9] als Moment der Geschlossenheit auf der anderen Seite: Entgegen dem Universalismus der Freie Software Advokaten verstehen die Hedonisten die Fähigkeit sich notwendiges Wissen gegebenenfalls selbst anzueignen als ein sinn-

[5] https://lists.ubuntu.com/archives/ubuntu-users/2011-February/239679.html (3.1.2017)

[6] http://www.mail-archive.com/debian-vote@lists.debian.org/msg08500.html (3.1.2017) zitiert in (Coleman 2013, 111)

[7] https://lists.archlinux.org/pipermail/arch-general/2012-May/026420.html (3.1.2017)

[8] Das heißt die menschenlesbaren Computerbefehle, die die Funktionsweise der Software beschreiben.

[9] Den Begriff Hedonismus verwende ich an dieser Stelle um die Freude und Lust am Programmieren zu betonen, die sich jedoch seltener durch die Beantwortung einfacher Fragen ergibt.

volles Unterscheidungskriterium für die Selektion von Mitgliedern und akzeptieren die damit einhergehende Schließung der Community als Teil einer meritokratischen Logik.

Expertise zwischen Legitimitäts- und Erweiterungsproblem

Im Sinne eines *programme, or be programmed* (Rushkoff 2011) wird die Möglichkeit der Laien selbst zu Expertinnen zu werden, und sich der Technik zu ermächtigen, verstanden als ein Recht von dem jede und jeder Gebrauch machen sollte, um unabhängig von den Technik-Herstellerinnen zu werden und selbstbestimmt Computer zu nutzen.[10] Experten und Laien verstehe ich dabei als soziales Verhältnis im Sinne Ingo Schulz-Schaeffers (2000) *Sozialtheorie der Technik*. Demnach gibt es eine grundsätzliche Wissensdifferenz zwischen den Entwicklerinnen von Technik (der Software), den Schöpfern und Hütern der Regeln, die der Technik zugrunde liegen einerseits, und den Nutzerinnen der Technik, die auf diese Regeln als Ressource ihres Handelns zurückgreifen, ohne sie notwendigerweise verstehen zu müssen andererseits. Analog zur Legitimitätskritik wissenschaftlicher Autoritäten (Collins, Evans 2002; Dickel, Franzen 2016) formiert sich in den 1980ern in Form der frühen Free Software Hacker Widerstand gegen die sich etablierende Dichotomie zwischen Entwicklerinnen und Nutzerinnen von Software. Im Zuge der Trennung von Software und Hardware entwickelt sich Software zu einem geschlossenen Wissensprodukt (vgl. Holtgrewe, Werle 2001). Die Erfindung der Freie-Software-Lizenzen wendet sich gegen die Macht der Entwickler und fordert daher Einsichtnahme in und Modifikationsrechte von Softwarecode: „The nonfree program controls the users, and the developer controls the program; this makes the program an instrument of unjust power."[11]

Ebenfalls analog zur Krise wissenschaftlicher Autorität ergibt sich aus der Lösung des *Legitimitätsproblems* – in diesem Falle durch die Erfindung der Freie-Software-Lizenzen – jedoch ein *Erweiterungsproblem* im Sinne einer Erweiterung des Kreises von Beitragenden und der dadurch verstärkten Problematik der Selektion von Beiträgen und Beitragenden: Einerseits ist Software als technisches Produkt stärker als wissenschaftliches Wissen auf eine produktive Schließung angewiesen, denn es muss eine lauffähige Programmversion geben. Andererseits nimmt auch die Anzahl interessierter Laien zu, deren Fragen zu beantworten zusätzliche Zeit kostet, die bei der Entwicklung fehlt.

Das Erweiterungsproblem wird gelöst durch epistemische Regime – „[Zusammenhänge] von Praktiken, Regeln, Prinzipien und Normen des Umgangs mit Wissen und unterschiedlichen Wissensformen" (Wehling 2007) – die somit die Wissens- und Softwareproduktion regulieren.

Epistemische Regime der Software Produktion

Gabriella Coleman beschreibt die Entstehung der GNU/Linux Distribution Debian als eine Reaktion auf die üblicherweise willkürlichen Selektionspraktiken, die ein ehemals für die *Berkeley Software Distribution*[12] programmierender Entwickler wie folgt skizziert:

[10] Und nicht zuletzt auch, um die negativen Netzwerkeffekte und Pfadabhängigkeiten der Hegemonie der kommerziellen Hersteller proprietärer Software zu verringern.

[11] https://www.gnu.org/philosophy/free-sw.en.html (7.1.2017).

[12] Die Berkeley Software Distribution ist eine freie Weiterentwicklung des Betriebssystems UNIX, mehr zur Entstehung findet sich bei Grassmuck (2004: 214).

> There was a process by which you wrote some code and submitted in the "I-am-not-worthy but I-hope-that-this-will-be-of-use-to-you supplication mode" to Berkeley, and if they kinda looked at it and thought, "Oh, this is cool", then it would make it in, and if they said, "Interesting idea, but there is a better way to do that", they might write a different implementation of it. (Coleman 2013, 129)

Mit dem Debian-Projekt schafft Ian Murdock 1993 eine formale Struktur, die transparente Regeln für eine verfasste Meritokratie festschreibt und damit auf transparente Kriterien setzt. Dabei entstehen eine Reihe von für das Projekt konstitutiven Dokumenten darunter die Debian Free Software Guidelines und eine eigene Verfassung. Debian ist somit eine reine Community-Distribution ohne federführendes Unternehmen im Hintergrund, und stellt daher einen interessanten Fall für die Untersuchung epistemischer Regime in der offenen Softwareproduktion dar.

Als Kontrastfälle betrachte ich Ubuntu Linux, das 2004 vom Unternehmen Canonical mit dem expliziten Anspruch GNU/Linux zu einem massentauglichen, also laienfreundlichen System auszubauen gestartet ist, und damit das Monopol von Microsoft auf dem Markt in Frage zu stellen gedachte. Initiiert wurde dies vom millionenschweren Weltraumtouristen Marc Shuttleworth. Eine konträre Intention liegt der Linux Community von Arch Linux zugrunde. Arch Linux (gegründet 2001) grenzt sich bewusst von einer Mode der Benutzerfreundlichkeit ab und bietet ein minimales Basissystem mit maximalen Konfigurationsmöglichkeiten. Der Begriff User-friendly wird dabei ersetzt durch User-centric, das bedeutet Arch Linux ist dezidiert konzipiert für „those contributing to it".[13]

Im Folgenden werde ich schlaglichtartig die vorgestellten drei Linux Communities beschreiben, dafür betrachte ich zunächst drei Kategorien: *Kollektives Selbstbild*, *Beitragsmöglichkeiten*, und *Strukturen und Verfahren*. Im Anschluss beschreibe ich in aller Kürze das Installationsprogramm der jeweiligen Linux Varianten, um zu zeigen, wie die verschiedenen Konfigurationen der epistemischen Regime sich im gemeinsamen Produkt, der Software, widerspiegeln.

Meine Empirie stützt sich maßgeblich auf qualitative Interviews, die ich mit Entwicklerinnen und Nutzern der betrachteten Communities durchgeführt und mit einer qualitativen Inhaltsanalyse nach Jochen Gläser und Grit Laudel (2008) untersucht habe. Ergänzend greife ich punktuell auf Kommunikate aus den Mailinglisten der Communities zurück.

Kollektives Selbstbild

Unter der Kategorie *Kollektives Selbstbild* betrachte ich hier insbesondere die Schwelle des Einstiegs in die Community, und was von Mitgliedern als Voraussetzungen für die Nutzung und eine aktive Beteiligung verstanden wird. Des Weiteren findet sich hier die Adressierung des Systems und damit die vorgestellten beziehungsweise adressierten Nutzertypen.

Ubuntu Linux

Entsprechend dem Motto Linux for Human Beings, mit dem Ubuntu startete, wird von Mitgliedern der Community berichtet, dass der Einstieg in die Community relativ niederschwellig ist. Teils ist dies dem oben erwähnten freundlichen Umgangston geschuldet, der in einem Code of Conduct verankert ist, wenngleich es punktuell auch hier Abweichungen gibt.

In den Interviews findet auch Erwähnung, dass ein guter nicht-technischer Einstiegspunkt sogenannte Local Meet-Ups sind, ähnlich etwa einer Tupper Party, wo neue Nutzer angesprochen werden,

[13] https://wiki.archlinux.org/index.php/Arch_Linux#User_centrality (7.1.2017).

und ihnen Hilfestellung gegeben wird. Dabei sind hier durchaus auch wörtlich „Telly-User“ (Fernseh-Nutzerinnen) adressiert.

Debian Linux

Hier fällt der Einstieg bisweilen berichtetermaßen schwierig, nicht zuletzt weil es starke Konventionen gibt, an die man sich halten sollte. Dementsprechend gibt es ein aufwendiges Bewerbungsverfahren um Debian Developer zu werden und damit aktives Mitglied zu werden. Ein wichtiger Teil dieses Verfahrens ist ein Nachweis der Kenntnis der wichtigsten Regelwerke, der Debian Free Software Guidelines und der Debian-Verfassung.

Adressierte Nutzerinnen des Systems sind zwar prinzipiell auch Einsteiger, aber die starke Orientierung an Free Software Prinzipien bringt für Laien manche Ecken und Kanten mit sich.[14] Auch die Fokussierung auf ein stabiles Universal-System, das auf einer großen Anzahl von Rechnerarchitekturen lauffähig ist, zeigt, dass die Hauptverwendung eher im professionellen Betrieb als im Heim-PC-Bereich liegt.

Arch Linux

Die Adressierung und Zielsetzung lässt sich kaum treffender zusammenfassen, als dies auf der eigenen Website beschrieben ist:

> Whereas many GNU/Linux distributions attempt to be more *user-friendly*, Arch Linux has always been, and shall always remain *user-centric*. The distribution is intended to fill the needs of those contributing to it, rather than trying to appeal to as many users as possible.[15]

Beitragsmöglichkeiten

Wenngleich die grundsätzlichen Beitragsmöglichkeiten in den unterschiedlichen Communities zunächst ähnlich erscheinen, so gibt es doch Unterschiede bei genauerer Betrachtung. Obwohl Nutzung und Beitrag idealiter verschmelzen, so differenzieren sich doch die konkreten Möglichkeiten des Beitragens unterschiedlich aus.

Zu den üblichen Beitragsmöglichkeiten gehören das Melden von Fehlern (Bugreporting), das Einreichen von Veränderungsvorschlägen (Patches), aber auch das Schreiben der Dokumentation und der Anleitungen einzelner Softwarekomponenten.

Ubuntu Linux

In Ubuntu fällt auf, dass ein Entwickler explizit anmerkt, dass als valider Fehlerbericht durchaus auch die subjektive Feststellung gilt, dass eine Funktionalität schwer verständlich ist, beispielsweise eine nicht intuitive Menüführung. Zudem wurde in Interviews angemerkt, dass im Unterschied zu anderen Communities hier ein bemerkenswerter Teil der Anleitungen und Wiki-Beiträge von Laien beigetragen werden. Eine prominente Stellung hat darüber hinaus die Mund-zu-Mund-Propaganda, die als sinnvoller und wichtiger sowie niederschwelliger Beitrag für die Community betrachtet wird.

[14] Nicht zuletzt weil im niedrigpreisigen Hardwaresegment üblicherweise günstige Hardware verbaut ist, für die das Angebot von Treibern, die den Free Software Prinzipien genügen, gering ist und dadurch häufig bei der Installation etwas „Handarbeit“ notwendig ist. Debian bietet zwar auch ein sogenanntes non-free Repository, das jedoch manuell eingebunden werden muss. Zur kontroversen Diskussion innerhalb der Community über das Angebot von non-free Software siehe auch Lazaro (2009).

[15] https://wiki.archlinux.org/index.php/Arch_Linux_ (7.1.2017).

Debian Linux
Bei Debian sticht ins Auge, dass hier (im Gegensatz zu einem ansprechenden Webinterface bei Ubuntu) der Bugreport über die Mailingliste erfolgt, und hierbei wiederum Konventionen zu beachten sind. Also spielt hier durchaus auch der technische Teil des sozio-technischen Arrangements eine Rolle für die Selektion der Beiträge bzw. der Beitragenden.

Außerdem haben hier üblicherweise die einzelnen Paketbetreuer die Hoheit über Änderungen an ihrem Teil des Debian-Universums, somit ist hier eine dezentrale Struktur vorzufinden und die jeweiligen Zuständigen gehen individuell sehr unterschiedlich mit Beitragsangeboten um.

Arch Linux
Bei Arch Linux möchte ich hier hervorheben, dass ein häufig referenzierter Bezugspunkt, das sogenannte Arch User Repository, kurz AUR darstellt. Hierbei handelt es sich um eine sozio-technische Infrastruktur, in der jede Nutzerin für alle Nutzerinnen ein Softwarepaket zur Verfügung stellen kann. Der User wird also hier mit einer Entwicklertätigkeit bedacht und die User-Rolle somit als Entwickler gerahmt.

Strukturen und Verfahren

Als letzte Kategorie wird nun ein Blick auf die *Strukturen und Verfahren* der Entscheidungsfindung gelegt. Hier offenbaren sich die Aufgaben und Rechte unterschiedlich zugewiesener Rollen. Zudem geben die Entscheidungskriterien Aufschluss über die Verfassung der jeweiligen epistemischen Regime.

Ubuntu Linux
Gegründet und initiiert von der Firma Canonical spielt das Unternehmen in der Entscheidungsfindung eine nicht unerhebliche Rolle. Zwar hält der Chef sich als „Selbsternannter wohlwollender Diktator auf Lebenszeit" offiziell weitgehend zurück, aber innerhalb und außerhalb der Community gibt es unterschiedliche Ansichten darüber, wie groß sein Einfluss und der seines Unternehmens tatsächlich sind. Zwar gibt es Entwicklungen, die erst ab einem gewissen Reifegrad der Community für Feedback frei gegeben werden, andererseits gibt es aber auch ein Usability Lab und dezidierte Community Manager, also Aspekte die durchaus geeignet erscheinen die Integration von Laien zu fördern.

Debian Linux
In Debian herrscht explizit technische Vernunft: Dezentral versuchen diejenigen Zuständigen, mit technisch sinnvollen Argumenten einen Konsens zu finden. Wo das nicht ausreicht, hilft das *Technical Committee*. Darüber hinaus gibt es eine Art (virtuelle) Generalversammlung, in der jede Entwicklerin eine Stimme hat. Wohlgemerkt nur diejenigen Entwickler, die einen Status als *Debian Developer*, also den dafür notwendigen Bewerbungsprozess erfolgreich absolviert haben. Die User selbst haben jedoch keine formale Rolle und auch keine Abstimmungsrechte.

Arch Linux
Hier gibt es eine kleine Entwickler-Elite, die die Richtungsentscheidungen trifft und die wichtigsten Programme (den sogenannten *Core*) verwaltet, den Rest verwalten die *Trusted User*, also wiederum User die in ihrer formalen Rolle mit Entwickler-Tätigkeiten betraut werden.

Eingeschriebene Werte im Installationsprogramm

Wie spiegeln sich nun die hier aufgezeigten Eckpunkte der unterschiedlichen epistemischen Regime in der produzierten Software wieder? Da bislang um Linux zu nutzen kaum ein Nutzer auf ein vorinstal-

liertes System zurückgreifen kann, sondern als ersten Schritt das System auf den Rechner bringen muss, vergleiche ich hier die üblichen Standard-Installationsprogramme.[16]

Ubuntu Linux
In Ubuntu fällt auf, dass die Installation üblicherweise von einer sogenannten Live-CD erfolgt, das ist eine CD von der aus man ein komplett lauffähiges System erst niederschwellig testen und anschließend direkt auf die Festplatte installieren kann. Die Grafik ist optisch sehr ansehnlich, und das Installationsprogramm weist gleich zu Beginn darauf hin, dass es von Vorteil ist, wenn der Rechner während der Installation am Strom angeschlossen ist. Hier wird schnell klar, dass der Laie auf einem recht niederschwelligen Wissensstand abgeholt wird, und befähigt werden soll, ein Betriebssystem zu installieren.

Debian Linux
Der *Debian Installer* kommt üblicherweise mit einer aufs Wesentliche reduzierten Grafik, in der Textboxen mit Rechtecken und einfachen Strichen dargestellt werden, was in seiner Ästhetik an frühere DOS-Grafiken erinnert. Hinweise zur Notwendigkeit der Stromzufuhr fehlen hier, vielmehr erfolgt recht bald im Prozess der Aufruf zur Partitionierung der Festplatte. Wenngleich auch eine geführte Partitionierung angeboten wird, setzt schon der Begriff der Partition hier einen gewissen Wissensstand voraus.[17]

Arch Linux
Hier besteht der *Installer* lediglich aus einer Befehlszeile und Hinweisen auf die Dokumentation, in der die notwendigen Befehle zu finden sind. Diese Befehle darf der User selbst eintippen, er hat somit die volle Kontrolle und wird faktisch selbst zum Installations-Skript.

Vorläufiges Fazit: *Regimes of Extension*

Die Freie Software Bewegung formuliert ein Legitimitätsproblem der Trennung zwischen Herstellerinnen und Nutzerinnen, das mit den Freie-Software-Lizenzen de jure aufgebrochen wird. Damit wird die Dichotomie zwischen Experten und Laien als Hersteller und Nutzerinnen von Technik idealiter aufgelöst. Dadurch ergibt sich in der Konsequenz ein Erweiterungsproblem, insofern dass für die produktive Schließung des Produkts eine Selektion der Beiträge und auch die Verarbeitung der Support-Anfragen von interessierten Laien zunimmt.

Unterschiedliche Communities lösen dieses Erweiterungsproblem auf verschiedene Weise durch die Ausbildung von *epistemischen Regimen*. Betrachtet habe ich hier GNU/Linux Communities, die ein sehr ähnliches Produkt entwickeln, nämlich ein Betriebssystem für Computer. Die unterschiedlichen Regime legitimieren aber unterschiedliche Selektionskriterien für valide Beiträge und akzeptierte Mit-

[16] Es gibt bei jedem System verschiedene Installations-Wege, ich greife hier die klassischen und in der Community präferierten Installationsprogramme heraus, um die damit verbundenen Setzungen herauszustellen.

[17] Zugegebener Maßen gibt es mittlerweile durchaus auch einen grafisch aufgehübschten Installer, der aber im Aufbau dem textbasierten weitgehend gleich ist, siehe auch https://www.debian.org/releases/stable/amd64/ch06s01.html.de#gtk-using (7.1.2017).

glieder der Community. Die ins Regime eingeschriebenen Werte spiegeln sich wider sowohl in der kommunikativen Praxis der Community (RTFM), als auch in der produzierten Software.

In Ubuntu wird durch verschiedene Setzungen ein Regime geführt, das geeignet erscheint in großem Maße Laien in die Community zu integrieren. Dies hat den Effekt, dass vergleichsweise viele Laien auch bei der Wissensproduktion beteiligt sind – insbesondere bei Dokumentation und Außen-Werbung (Advocacy). Es herrscht also eine Diversität der Nutzer von Laie bis Experte mit Überschneidungsbereichen zwischen Entwicklerinnen und Nutzerinnen. Jedoch kommt es zum Teil zur Schließung mancher Entwicklungsprozesse, so besitzt das Unternehmen Canonical eine zentrale Stellung, und trifft teils strategische Entscheidungen ohne die Partizipation der Community.

Bei Arch hingegen kann man eine starke Integration der Nutzerinnen in die Software-Entwicklung beobachten. Jedoch geht bei dieser starken Form der technischen Integration der User *in die Entwicklung* die Integration der *Laien* zum Teil verloren. Integriert werden hier lediglich jene Laien, die bereit sind, die notwendigen Wissens-Hürden zu nehmen und allmählich zu Experten zu werden.

Interessant ist hierbei, dass bei Debian eine formale Trennung von Entwicklerinnen und Nutzern festgeschrieben ist. Informell besteht dennoch eine Mitarbeit der User, aber für den formalen Status eines Debian Developers ist ein bemerkenswerter Bewerbungsprozess notwendig (siehe oben).

Eine gewisse Trennung von Entwicklung und Nutzung scheint also nicht vermeidbar. Diese Trennung kann jedoch sehr unterschiedlich organisiert sein. Strikt formell wie in Debian, hier wird dadurch eine formale Struktur geschaffen, die in sich sehr egalitär ist, aber schon eine starke Geschlossenheit aufweist. In Arch fällt Nutzung und Entwicklung im Grunde zusammen und kommt somit der Idealvorstellung einer Aufhebung der Grenze zwischen Entwicklung und Nutzung am nächsten. Jedoch wird der Status des Laien hierbei als temporär betrachtet und das Verbleiben im Laien-Status, also der Zugriff auf die Regeln ohne sie zu verstehen, ist nicht vorgesehen. Am anderen Ende des Spektrums gelingt in Ubuntu eine Integration der Laien in die Nutzung, aber auch in aktive Beiträge bis hin zur Entwicklung. Hierbei spielt ein aktives Community Management eine tragende Rolle und es gibt einen Bereich der strategischen Schließung ohne Partizipationsmöglichkeiten.

Die Konfigurationen der epistemischen Regime reichen also von kleinen unterschiedlichen funktionalen Bedeutungen eines Akronyms (RTFM) bis hin zu verschiedenen Rollenbeschreibungen (was sind die Aufgaben eines Users) und Organisationsformen (zum Beispiel Community Management).

Gewissermaßen steht Arch dabei für die Nische der Freien Software Entwicklung der alten Schule und Ubuntu für die pragmatische Nutzung dieser Nische und ihrer Formierung in ein anschlussfähiges Produkt.[18] Die verschiedenen Communities profitieren am Ende trotz der gegenseitigen Abgrenzung voneinander, die Verbreitung auf größere Nutzerkreise führt potenziell zu einer besseren Linux-Unterstützung von Hardwareherstellern und die technisch-versierten Anwenderinnen begünstigen die technische Weiterentwicklung der Projekte – und die User haben letztlich die Wahl zwischen verschiedene normativen Rahmen.

Darüber hinaus zeigt die Betrachtung der epistemischen Regime, wie in der Produktion von Wissen unterschiedliche Normen die spezifischen „Wahrheiten“ und Ziele beeinflussen und dadurch eine Selektionsleistung erfüllen, die von der Community-Ebene bis hin ins Wissens-Produkt wirksam ist.

[18] Nicht zu vergessen, dass auch andere Unternehmen sich die Nische Open Source zunutze machen, und damit auch finanzielle Erfolge erzielen (vgl. Schrape 2016) – was im Fall Ubuntu aber noch strittig ist.

Literatur

Coleman, E. G. 2013: Coding freedom: The ethics and aesthetics of hacking. Princeton: Princeton University Press.

Collins, H., Evans, R. 2002: The third wave of science studies: Studies of expertise and experience. Social Studies of Science, Vol. 32, Issue 2, 235–296.

Dickel, S., Franzen, M. 2016: The "Problem of Extension" revisited: New modes of digital participation in science. Journal of Science Communication, Vol. 15, Issue 01, 1–15.

Gläser, J., Laudel, G. 2008: Experteninterviews und qualitative Inhaltsanalyse als Instrumente rekonstruierender Untersuchungen. Wiesbaden: VS Verlag für Sozialwissenschaften, 3. überarbeitete Auflage.

Grassmuck, V. 2004: Freie Software: Zwischen Privat- und Gemeineigentum. Bonn: Bundeszentrale für politische Bildung, 2. korrigierte Auflage.

Holtgrewe, U., Werle, R. 2001: De-commodifying software? Open Source Software between Business strategy and social movement. Science Studies, Vol. 14, Issue 2, 43–65.

Lazaro, C. 2009: From source code to text code: Textual norms and practices in the Debian Community. http://www.constantvzw.org/verlag/spip.php?page=article&id_article=119# (letzter Aufruf 7.1.2017).

Rushkoff, D. 2011: Program or Be Programmed: Ten commands for a digital age. Berkeley: Soft Skull Press.

Schrape, J.-F. 2016: Open-Source-Projekte als Utopie, Methode und Innovationsstrategie: Historische Entwicklung – Sozioökonomische Kontexte – Typologie. Glückstadt: Hülsbusch.

Schulz-Schaeffer, I. 2000: Sozialtheorie der Technik. Frankfurt am Main: Campus-Verlag.

Wehling, P. 2007: Wissensregime. In R. Schützeichel (Hg.), Handbuch Wissenssoziologie und Wissensforschung. Konstanz: UVK, 704–712.