Koordination überbetrieblicher Wissensproduktion – zum Spannungsverhältnis zwischen Unternehmen und Communities in Open Source Projekten mit Unternehmensbeteiligung[1]

Patrick Feuerstein/Heidemarie Hanekop

Abstract

Open Source (OSS) Communities sind eine spezifische, durch Digitalisierung zugleich von Arbeitsprozessen und Produkten ermöglichte Form kollaborativer Innovationsarbeit, die wegen ihres besonderen Potentials zur Organisation verteilter, überbetrieblichen Wissensproduktion zunehmend auch von Unternehmen für kommerzielle Zwecke genutzt wird. Doch obgleich digitale, technische Infrastrukturen für diese Form von Innovationsarbeit zentral sind, ist das Gelingen von Community-basierten Innovationen auf jenseits der Technik liegende, sie stützende soziale Mechanismen von Gemeinschaftlichkeit angewiesen ist. Der Beitrag untersucht anhand einer Fallstudie die durch digitale Infrastrukturen vermittelte Koordination unterschiedlicher Akteure innerhalb einer OSS-Community mit spezifischem Blick auf Konflikte zwischen Unternehmen und Community und deren Bearbeitung durch die Akteure im Spannungsverhältnis von gemeinschaftlichen Governance in der Community und hierarchischer Governance in den Unternehmen.

1. Einleitung

In der Softwarebranche hat sich seit den 1990er Jahren mit der Open Source Softwareentwicklung (OSS) eine besondere Form gemeinschaftlicher Softwareentwicklung etabliert. OSS beruht auf dem digitalen Charakter von Arbeitsprodukten und –prozessen, kombiniert diese jedoch mit spezifischen nicht-technischen, gemeinschaftlichen Formen der Governance in einer spezifischen Weise. OSS wird auch von Unternehmen zunehmend als innovative, digitale Infrastruktur für den Wissensaustausch zwischen vielen heterogenen Akteuren entdeckt und kann daher als wichtiges Innovationsmodell einer zunehmend digitalisierten Arbeitswelt angesehen werden (siehe u.a. Dahlander and Magnusson 2005; David and Shapiro 2008; Giuri et al. 2008; Mateos-Garcia and Steinmueller 2008; O'Mahony 2007; O'Mahony and Lakhani 2011; ; Von Hippel 2005; West and Gallagher 2006; West and Lakhani 2008; West and O'Mahony 2008).

OSS mit Unternehmensbeteiligung in kommerziellen Innovationsprozessen stellt dabei eine interessante Mischung unterschiedlicher (und häufig als widersprüchlich angesehener) Governanceformen dar, treffen gemeinschaftliche Formen der Wissensproduktion innerhalb der Communities auf Unternehmen, die intern hierarchisch koordiniert sind und zugleich kommerzielle Interessen verfolgen (Faraj et al. 2016; Lakhani et al. 2013). Daher ist es überraschend, dass trotz der zunehmenden Beteiligung von Unternehmen an wichtigen Open Source Projekten (Schrape 2015) bisher nur wenige Studien untersucht haben, wie diese unterschiedlichen Mechanismen der Handlungskoordination in konkreten Prozessen des überbetrieblichen Wissensaustauschs wirken und wie die Beteiligten mit den Spannungen zwischen Hierarchie und Gemeinschaft konkret umgehen (für Ausnahmen, siehe z.B. Dobusch and Quack 2011; West and O'Mahony 2008).

Der folgende Beitrag versucht, genau dieser Frage nachzugehen: Unsere Ausgangsthese ist, dass OSS-Communities mit ihrer spezifischen Verschränkung technischer Infrastrukturen, spezifischen Institutionen, Regeln und sozialen Praxen ein Modell für erweiterte Formen der digital vermittelten Wissensproduktion über Organisationsgrenzen hinweg darstellen. Die Implikationen für die Unternehmen, die sich dieser Möglichkeit des Zugangs zu externem Wissen bedienen, so unser Argument weiter, sind dabei jedoch weitreichend und greifen tief in die internen, betrieblichen Unternehmensprozesse ein. So besagt der zweite Teil des hier entwickelten Arguments, dass Community-basierte Innovationsprozesse für die beteiligten Unternehmen mit systematischen Steuerungsproblemen einhergehen, die von diesen organisatorisch und personell bewältigt werden müssen.

Zunächst werden im ersten Schritt die Herausforderungen der Wissensproduktion in verteilten Innovationsprozessen skizziert. Anschließend werden OSS-Communities als Sonderformen gemeinschaftlicher

[1] Die Autoren möchten gerne dem Team und den Teilnehmern der Abschlusskonferenz des Projekts COLLIN, den Herausgebern der Zeitschrift Arbeit sowie den anonymen Gutachtern für die Hinweise zur Verbesserung dieses Artikels danken.

Governance in ihren wesentlichen Eigenschaften skizziert und zentrale Institutionen und Strukturmerkmale herausgestellt. Im vierten Abschnitt schließlich wird anhand der Fallstudie einer OSS-Community mit einem besonderen Fokus auf den unterschiedlichen Governanceformen diskutiert, auf welche Weise deren besondere Eigenschaften zu einer gelingenden Wissensproduktion zwischen den beteiligten Akteuren beitragen. Anschließend wird das Spannungsverhältnis zwischen Unternehmen und Community analysiert und dargestellt, wie die beteiligten Unternehmen und Entwickler in der sozialen Praxis mit den Herausforderungen umgehen, die aus den widersprüchlichen Interessen und Koordinationsmechanismen resultieren.

2. Herausforderungen der Wissensproduktion in verteilten Innovationsprozessen

Innovationsprozesse, die über Organisationsgrenzen hinausgehen, potenzieren ein Problem, das die Innovationsforschung auch schon für innerbetriebliche Innovationsprozesse herausgearbeitet hat: das für Innovationen notwendige und gewünschte Wissen ist gewöhnlich über verschiedene Wissensträger verstreut. Damit Innovationen zustande kommen, müssen also verschiedene Akteure zusammengebracht werden, mit allen Folgen, die sich aus den Problemen kollektiven Verhaltens in und zwischen ausdifferenzierten und vermachteten Organisationen, aber auch aus den unterschiedlichen Entstehungskontexten von Wissen folgen können.

So ist einerseits das für Innovationen relevante Wissen in der Regel nicht unabhängig von den beteiligten Akteuren zu stabilisieren. Nach Heidenreich (2003: 27) kann Wissen als „lernbereite Deutungsschemata“ gefasst werden, „die den natürlichen und sozialen Lebensbedingungen der Menschen einen Sinn geben und die ihr praktisches Verhalten regeln“. Hiernach ist Wissen zum einen stets vorläufig, zum anderen aber auch situationsabhängig und kontextgebunden. Die Kontextgebundenheit des Wissens macht sich vor allem daran fest, dass Wissen nicht beliebig expliziert und kommuniziert werden kann, eine Tatsache, auf die Polanyi (1985) mit seiner klassischen Unterscheidung zwischen implizitem und explizitem Wissen hingewiesen hat. Nach Polanyi ist es gerade der (über-)individuelle Background der Akteure, der das Verstehen und sinnvolle Anwenden expliziter, kommunizier- und formalisierbarer Wissensarten, wie z.B. technischer Artefakte und Routinen, überhaupt erst möglich macht (vgl. auch Heidenreich 1995; Tsoukas 1996). In betriebsübergreifenden Innovationsprozessen muss daher Wissen integriert werden, das über verschiedene Organisationen verteilt und an den jeweiligen Entstehungskontext gebunden ist.

Als zweite Herausforderung betriebsübergreifender Innovationsprozesse kann nach Krogh (2011) das Problem kollektiven Verhaltens bei der verteilten Wissensproduktion angesehen werden. Diese Perspektive betont vor allem zwei relevante Dimensionen des Problems betriebsübergreifender Innovationsprozesse. Die erste Dimension weist darauf hin, dass die Kombination unterschiedlicher Wissensbestände als soziale Praxis verstanden werden muss: Wissen kann demnach nicht einfach durch formale Prozeduren und Kommunikationen übermittelt werden, erfolgreiche Vermittlung setzt vielmehr auch das Teilen des jeweiligen Backgrounds der Akteure in einer gemeinsamen (und gemeinsam zu entwickelnden) sozialen Praxis voraus (Krogh 2011, Faraj et.al. 2016). Dieser Auffassung folgend, wird Wissen in verteilten Innovationsprozessen weniger *transferiert* als in sozialen Interaktionen stets *neu geschaffen*. Welche Anreize für die einzelnen Akteure jeweils gegeben sein müssen, damit sie ihr Wissen einbringen, ist daher auch die zweite Dimension des Problems verteilter Innovationsprozesse: können Unternehmen bei internen Innovationsprozessen zumindest noch auf gemeinsame organisatorische Sozialisationsprozesse und eine zumindest z.T. gemeinsame Zielsetzung der Akteure im Rahmen gemeinsamer Ziele der Organisation vertrauen, so müssen in verteilten Innovationsprozessen Akteure zusammengebracht werden, die in ganz unterschiedlichen organisatorischen Kontexten situiert sind und unterschiedliche Zielsetzungen verfolgen können.

3. Open Source Communities als Gemeinschaftsorganisationen

OSS-Communities können als eine besondere (modernisierte) Form gemeinschaftlicher Governance verstanden werden (West and Lakhani 2008, O’Mahony and Lakhani 2011). Gemeinschaft wird in der Governanceforschung allgemein als ein Koordinationsmechanismus begriffen, der auf der Handlungssteuerung der Mitglieder durch ein Gefühl der Zugehörigkeit beruht und sich durch solidarisches Verhalten in Bezug auf die Ziele der Gemeinschaft auszeichnet.

> „Man agiert, weil man anderen in einem spezifischen Merkmal gleicht, und man agiert deswegen auf eine spezifische Weise.“ (Gläser 2007: 87)

Die Zugehörigkeit muss dabei nicht auf festen Regeln und Festlegungen beruhen, sondern kann auch durch die subjektive Wahrnehmung der Akteure hergestellt werden (vgl. Gläser 2007; Streeck and Schmitter 1985). Ist mit der kollektiven Identität der Mitglieder der zentrale Koordinationsmechanismus gemeinschaftlicher Governance – und damit das zentrale Spezifikum gerade auch in Abgrenzung zu Markt, Netzwerk und Organisation – benannt (vgl. Streeck and Schmitter 1985: 122), so reicht diese Eigenschaft zur Charakterisierung der Koordination kollektiven Handelns durch Gemeinschaften nicht aus.

So bestimmen Dolata and Schrape (2014) Gemeinschaften – über die bereits erwähnte kollektive Identität hinaus – über zwei weitere Merkmale. Zum einen verweisen sie auf „Institutionalisierungsdynamiken, die kollektives Handeln auf der Basis eigener, vornehmlich informeller Regeln, Normen und Organisierungsmuster ermöglichen, strukturieren und stabilisieren“, und zum anderen auf „interne Differenzierungsprozesse, in denen sich mit der Zeit organisierende Kerne und meinungsführende Aktivisten mit umliegenden Peripherien aus unterstützenden Teilnehmern herauskristallisieren“ (S.19, für eine ähnliche Definition, siehe auch Dobusch and Quack 2011).

Es ist gerade diese „Institutionalisierung des Kollektiven“ (Dolata and Schrape 2014: 19), die Gemeinschaften auch zu strategisch handlungsfähigen Akteuren macht. Die gemeinsam geteilten Normen und informellen Regeln stellen die Grundlage für kollektives Handeln dar und ermöglichen Gemeinschaften damit eine gewisse Eigenständigkeit und Eigengesetzlichkeit gegenüber externen Akteuren. Mit der Berücksichtigung interner Differenzierung und Institutionalisierung von Regeln und Entscheidungsprozessen mischen sich in gewisser Weise organisatorische und gemeinschaftliche Formen der Governance (Wiesenthal 2005). Handlungsfähige Gemeinschaften können daher auch als *Gemeinschaftsorganisationen* bezeichnet werden. Ahrne and Brunsson (2011) fassen diese Prozess auch als „partielle Organisation“, um zu beschreiben, dass Gemeinschaften zwar organisatorische Elemente entwickeln, ohne dadurch allerdings alle Eigenschaften formaler Organisationen aufzuweisen: als wesentliche Unterschiede heben Dobusch und Quack (2011) in dieser Hinsicht zum einen die Regelung der Zugehörigkeit hervor, die in Gemeinschaften häufig auf Beiträgen zu dem gemeinschaftlichen Zielen und nicht auf formalen Zugangsberechtigungen und -bedingungen beruht. Eng damit zusammen hängt auch die Frage des Einflusses innerhalb der Gemeinschaft und des Modus der Entscheidungsfindung. Gemeinschaftliche Entscheidungen werden häufig konsensorientiert und nach (oft: impliziten) meritokratischen und nicht nach formal rechtlichen Regelungen gefällt. Gemeinschaften als strategisch handlungsfähige Akteure lassen sich also generell mit den drei Merkmalen einer *internen Institutionalisierung*, der Herausbildung einer eigenen *kollektiven Identität* und *interner Differenzierungs- und Organisationprozesse* bestimmen (Dolata and Schrape 2014:19; für ähnliche Definitionen siehe z.B. Dobusch and Quack 2011; Gläser 2007).

Jedoch weisen OSS-Communities auch Eigenschaften auf, die über die allgemeine Bestimmung gemeinschaftlicher Governance hinausgehen. Die besondere technische Infrastruktur ist für diesen Typ von Gemeinschaftsorganisation charakteristisch, wie Dolata and Schrape (2014) herausstellen. OSS-Communities organisieren sich im Rahmen technischer Infrastrukturen, die sowohl die Formen der technischen als auch der sozialen Interaktion entscheidend vorstrukturieren (Dolata and Schrape 2014). So ist der technische Prozess der Open Source Softwareentwicklung weitgehend transparent, er wird über kollaborative Entwicklungsplattformen im Web (z.B. GitHub) organisiert, auf denen sich Entwickler und Anwender aus aller Welt beteiligen können, um die neuen Funktionen in ihrer Systemumgebung zu testen und/oder Änderungen vorzuschlagen. Spezielle Lizenzen stellen sicher, dass der gemeinsam generierte Code nicht proprietär verwendet werden darf und offen bleibt (für einen Überblick über geläufige Lizenzen, siehe z.B. http://opensource.org/licenses). Über die Speicherung des Codes hinaus bieten viele Plattformen begleitende technisch basierte Informations- und Kommunikationskanäle (Mailinglisten, IRC, u.ä.), die das gemeinsame Arbeiten an geteilter Code-Basis strukturieren. Eine wichtige Funktion dabei ist häufig auch die Speicherung der vorherigen Kommunikation in Archiven, die einmal gefällte Entscheidungen und Auseinandersetzungen für später hinzugekommene Entwickler nachvollziehbar macht. Die Entwicklungsplattformen können damit als das zentrale Instrument für das Teilen expliziten oder explizierten Wissens in OSS-Communities angesehen werden. Wie jedoch im folgenden noch genauer herausgearbeitet wird, gehen OSS Communities keinesfalls in diesen technisch gestützten Infrastrukturen auf, sondern benötigen für ihr Funktionieren vielmehr der Ergänzung durch nicht-technische, soziale Elemente.

4. Fallstudie: Wissensproduktion, Steuerungsprobleme und -lösungen im OSS-Projekt „MSB“

4.1. Datenbasis, Design und Sample der Fallstudie

Die untersuchte MSB-Community entsteht in Rahmen der in den 90er Jahren erstarkenden Linux-Bewegung. Microsoft hat zu diesem Zeitpunkt eine Monopolstellung bei den PC-Betriebssystemen und beginnt diese auf den bis dato von anderen Anbietern beherrschten Markt von Client-Server Netzwerken auszudehnen. Die von Microsoft auf dieser Grundlage etablierte Netzwerk-Software basiert zwar auf einem ursprünglich offenen Protokoll, das aber von Microsoft zu einem geschlossenen Netzwerk-System weiterentwickelt wird, was alternative Netzwerksysteme zunehmend ausschließt. Die Gründer der MSB-Community beginnen Anfang der 1990er Jahre eine Software zu entwickeln, die die Interoperabilität zwischen Windows und Unix- bzw. Linux-basierten Netzwerkinfrastrukturen ermöglichen soll. Wegen der strategischen Bedeutung dieser Software als Schnittstelle zu Windows-Netzwerken wird die ‘MSB’-Community für Konkurrenten von Microsoft in diesem

Feld höchst interessant: sie entwickelt jenseits von singulären Unternehmensinteressen ein Produkt, das geeignet ist, der Dominanz von Microsoft in diesem Bereich entgegenzutreten, was keinem der großen Unternehmen vorher gelungen war. In der Folge nimmt die Zahl der in einem Konkurrenzunternehmen von Microsoft tätigen Entwickler in der Community stetig zu. Ein entscheidender Einschnitt erfolgt 2007, als der Europäische Gerichtshof u.a. auf Druck der an der Community beteiligten Akteure in einem Grundsatzurteil Microsoft zwingt, die Spezifikationen seines Netzwerkprotokolls zu dokumentieren und öffentlich zugänglich zu machen. Dieser Einschnitt markiert eine Wende in Microsofts bis dahin verfolgter Strategie gegenüber der MSB-Community. War Microsoft vorher bemüht, die Arbeit der Community zu erschweren und sah diese in erster Linie als Gefahr für den eigenen Unternehmenserfolg an, so zeigt sich Microsoft nun bereit, auch über die vom Gericht geforderten Maßnahmen hinaus mit der Community zu kooperieren. (für eine Auseinandersetzung mit Microsofts Strategiewechsel, siehe auch Lange 2009).

Die Verantwortung und auch das Copyright (als Open Source) für die Software liegt bei der Community, sie organisiert den Produktionsprozess nach den Regeln der Open Source Lizenz Gnu Public Licence (GPL) und im Rahmen einer Open Source Community.

Charakteristisch für die Mitgliederstruktur der MSB-Community ist einerseits ihre Konstanz, andererseits ihre enge Verknüpfung mit Unternehmen, deren Strategien in unterschiedlicher Weise mit dem von der Community verbreiteten Softwareprodukt verknüpft sind, z.B. weil sie die Software für ihre eigenen Produkte weiterentwickeln oder weil sie Dienstleistungen rund um die OSS anbieten.

Fast alle Mitglieder der Community sind professionelle Entwickler aus Unternehmen. Die Zahl der zu einem bestimmten Zeitpunkt aktiven Hauptentwickler schwankt in einer Größenordnung zwischen 25 bis 40. Die Gesamtzahl der Entwickler, die seit 2008 beigetragen haben liegt bei 360.

Uns interessieren im Rahmen unserer Fallstudie vor allem die Unternehmen, die die Software nicht nur für ihre eigenen Produkte einsetzen, sondern auch strategisch weiterentwickeln und dabei mit eigenen Mitarbeitern und Beiträgen konstant an der Entwicklung beteiligt sind. Mit einem Entwicklungsdienstleister und einem Linux-Distributor fokussieren wir in unserer Fallstudie auf zwei dieser Unternehmen.

Der Entwicklungsdienstleister bietet seinen Kunden ergänzende Dienstleistungen zur MSB-Software an. Seine Kunden sind Unternehmen, die „MSB“ entweder im operativen Einsatz im eigenen Unternehmen oder in ihren Produkten (in Hardware, Software oder IT-Services) verwenden. Die Leistungen reichen von der Problemlösung im operativen Einsatz, kleineren Entwicklungsaufträgen zur Anpassung an besondere Anforderungen bis hin zu großen und z.T. sehr offenen Entwicklungsaufträgen. Das Unternehmen ist mit mehreren Hauptentwicklern an der Community beteiligt.

Der Linux Distributor ist eines der größten Unternehmen im Linux-Ecosystem und entwickelt die MSB-Software als integralen Teil der eigenen Linux-Distribution mit. Für den Linux Distributor ist es wichtig, dass die eigene Linux Distribution mit gängigen Netzwerk-Infrastrukturen kompatibel bleibt – sowohl innerhalb des Linux Ecosystems wie auch zur Windowswelt, daher stellt MSB eine Kernkomponente des Produkts dar. Der untersuchte Linux Distributor stellt mittlerweile das größte unternehmenseigene Entwicklerteam in der Community.

Im Rahmen unserer Fallstudie haben wir zehn leitfadengestützte Interviews mit unterschiedlichen Akteuren der Community innerhalb und außerhalb der für diesen Aufsatz fokussierten Unternehmen geführt (Entwickler, Projektleiter und Geschäftsführung). Gegenstand der Interviews waren, je nach Position in unterschiedlicher Gewichtung, Fragen der Kooperation und der Wissensproduktion der Entwickler im Rahmen der OSS Community, sowie der Strategien der unterschiedlichen Akteure, auf den Entwicklungsprozess Einfluss zu nehmen. Die Vertreter der Unternehmen wurden zudem zu deren Geschäftsmodellen und strategischen Planungen in Bezug auf die OSS Community befragt. Zudem sind umfangreiche Recherchen einschlägiger Repositories und Websites der untersuchten Community durchgeführt worden. Insbesondere konnte auch an der dreitägigen Entwicklerkonferenz im Jahr 2014 beobachtend teilgenommen und die internen Diskussionen verfolgt werden[2].

4.2. Gemeinschaftliche Koordination überbetrieblicher Wissensproduktion

Im ersten Schritt soll untersucht werden, inwiefern die gemeinschaftliche Koordination des Entwicklungsprozesses innerhalb der OSS-Community die Wissensproduktion zwischen den beteiligten Akteuren beeinflusst. Gemäß der oben entwickelten Begriffsbestimmung orientieren wir uns bei der Untersuchung an den vier Dimensionen gemeinschaftlicher Governance in Form von OSS-Communities:

[2] Die Fallstudie entstand im Rahmen des Forschungsprojekts „Kollaborative Innovationen“ (COLLIN) das am Soziologischen Forschungsinstitut Göttingen (SOFI) und der Universität Oldenburg durchgeführt und vom Ministerium für Wissenschaft und Kultur Niedersachsen aus Mitteln der VW-Stiftung „Niedersächsisches Vorab“ gefördert wurde (Kennzeichen VWZN2833).

technische Infrastruktur, kollektive Identität und gemeinsame Ziele, Institutionalisierung kollektiver Vorstellungen bzw. Normen kollektiven Handelns und interne Differenzierungs- und Organisationsprozesse.

4.2.1. Offenes Produkt und transparenter Produktionsprozess

Der wohl auffälligste Aspekt des Wissensaustauschs innerhalb der Community ist die durch die verwendete OpenSource-Lizenz sichergestellte Offenheit des Produkts und des durch die technische Infrastruktur und digitale Natur des Produkts bedingte Transparenz des Produktionsprozesses.

Die Software ist nicht nur als Quellcode frei verfügbar, sondern auch der Prozess ihrer Entwicklung in der Community ist für jeden Interessierten auf der Kollaborationsplattform im Web transparent nachvollziehbar. Jeder der über entsprechende Kompetenzen verfügt, kann den aktuellen Stand und gegenwärtige Arbeiten an der Software einsehen. Dies betrifft nicht nur den aktuellen Bestand an geschriebenem Programmcode, sondern aufgrund eines bereit gestellten Archivs der Mailingliste auch die im Laufe der Entwicklung geführten Diskussionen zwischen den Entwicklern. Dadurch lässt sich der Entwicklungsprozess über Jahre zurückverfolgen.

Auf dieser Grundlage entfaltet sich im untersuchten Projekt[3] eine offene Diskussionskultur, die Austausch und Kooperation der involvierten Entwickler erleichtert, wie folgende Entwicklerin beschreibt:

> "That's the beauty of Open Source. If you can program you take the source and you start talking. It's not always easy. But you just start talking. If you have a problem, you take the code, you create a fix for the problem and you propose it to the community, usually done by the mailing list."
> (FS17-Entwickler-Community-1)

Dieser technisch gestützte, offene Entwicklungsprozess hat weitreichende Effekte für den Umgang mit dem in der Software verkörperten Wissen. So sind Entwickler in der Lage, beliebige Teile der Software zu verändern, indem sie den bestehenden Code nehmen und weiterentwickeln – unabhängig davon, von wem er geschrieben wurde. Durch die Rekonstruktion des Entstehungsprozesses besteht gerade für neu im Projekt beteiligte Entwickler die Möglichkeit, Entstehungsprozess und Entscheidungsgründe für bestimmte Design-Entscheidungen nachzuvollziehen, um die Logik des Codes besser erfassen zu können. Zudem haben andere Entwickler die Möglichkeit, bereits in einem frühen Stadium der Entwicklung Unterstützung anzubieten, wie das folgende Zitat verdeutlicht:

> „Das ist das Wesen [...] von Open Source [...]: ich muss mit dem, was ich tue, nicht ins Kämmerlein gehen und hinter mir zuschließen [...], sondern ich kann auch währenddessen schon offen sein. […] Und es passiert übrigens tatsächlich, dass auf der Mailingliste Leute auftauchen, die sagen, ist ja interessant, kann ich da irgendwas testen, kann ich irgendwas helfen."
> (FS17-IT19)

Bezogen auf die *Wissensproduktion* können wir daher festhalten, dass mit dem offenen Code und der offenen, digitalen Kollaborationsplattform die expliziten Bestandteile des Wissens über die Grenzen der Community und der involvierten Unternehmen hinweg frei verfügbar sind. Es wäre jedoch sehr vereinfachend, die Community auf diesen technischen Aspekt und die Wissensproduktion auf den freien Zugriff auf gemeinsame Code-Teile zu reduzieren (vgl. auch Dolata and Schrape 2014). Und auch wenn die Mailingliste (und das Archiv derselben) das zentrale Medium für den Austausch und die Koordination in der Community ist, nutzen die Entwickler darüber hinaus vielfältige Möglichkeiten der persönlichen bilateralen Kommunikation, sowohl digitale Medien, wie Email und Chat, aber auch face-to-face Treffen, insbesondere auf der jährlichen Entwicklerkonferenz der Community. Gerade um zu verstehen, wie der Wissensaustausch innerhalb der Community funktioniert, ist es wichtig, die sozialen Praxen der Community zu untersuchen, die es den beteiligten Entwicklern ermöglichen, auch implizite Wissensbestandteile zu teilen.

4.2.2. Kollektive Identität, Institutionen und interne Organisationsprozesse der Community

Grundlegend für die Praxis der untersuchten Community ist das gemeinsame Ziel seiner Mitglieder, die MSB-Software als funktionsfähige Alternative zu Microsofts Netzwerkprotokoll zu entwickeln und zu verbreiten. Die Beiträge der Entwickler in der Community sind wesentlicher Bestandteil ihres fachlichen und beruflichen Selbstverständnisses und ihrer Identität im Rahmen der Community wie in dem Unternehmen, in dem sie beschäftigt sind. Die Community etabliert Regeln und Normen für die Koordination der verteilten Arbeit, Positionen und Entscheidungsprozesse, d.h. sie bildet eine Organisationsform für die gemeinschaftliche Produktion.

---

[3] Dies muss allerdings nicht immer so sein, eine gute Diskussionskultur muss in OSS vielmehr als Variable betrachtet werden, die durchaus auch zum Scheitern von Projekten führen kann, wie wir auch später noch diskutieren werden.

Die Arbeitsverteilung in der Community ist selbstorganisiert und freiwillig. Daher gibt es auch keine Weisungsbefugnis einzelner Mitglieder anderen gegenüber. Eine wesentliche Triebfeder für Beiträge ist die Wahrnehmung von Bedarfen und Defiziten, sowie die Solidarität und Bereitschaft von einzelnen Entwicklern, die Probleme von anderen zu lösen:

> „Es war so eine Sache, der Schuh hat gedrückt und ich glaube, weil die Leute gesagt haben, hey du kannst doch WinBind, wir haben hier einen komischen WinBind-Bug und ich hab mir den angeguckt und der sah auch nicht so besonders toll aus. [...] Und dann haben wir halt innerhalb von einer Woche diesen DNS-Server da hingestellt. Weil es aussah, als ob man den braucht... Das war jetzt nicht, dass irgendjemand vorher gesagt hat, das brauchen wir dringend, sondern man hat halt gesehen, ok hier drückt der Schuh gerade und ich hab gerade Zeit, dann mach ich das einfach mal."
> (FS17-Entwickler-Community-2)

Neben der fehlenden Weisungsbefugnis existieren auch keine direkten Sanktionsmöglichkeiten, durch die Tätigkeiten von Mitgliedern erzwungen werden könnten. Statt dessen haben sich informelle Regeln herausgebildet, die soziale Erwartungen konstituieren. Die gemeinsamen Ziele bezogen auf die Software, sowie gemeinschaftliche Werte, wie Solidarität und Aufmerksamkeit für Anforderungen anderer, sind in der Community entscheidende Faktoren, die die Bereitschaft zur aktiven Mitarbeit befördern. Dabei spielen natürlich auch die Wahrnehmung der eigenen Kompetenz und die soziale Anerkennung und Zuweisung von Kompetenzen eine Rolle. In diesem Zusammenhang können soziale Beziehungen, Sympathie und Reziprozitätsnormen eine starke Bedeutung gewinnen:

> „Es passiert ja nichts wenn ich es nicht tue. Ich hab ja keinen Chef, der mir auf die Finger haut, wenn ich irgendwas nicht tue. Das trifft halt bei der ganzen Open Source-Entwicklung viel, viel stärker zu, dass man die Aufmerksamkeit der Leute sich ganz anders erarbeiten muss. Man kann nicht mit Geld und irgendwelchen Sanktionsmöglichkeiten etwas erzwingen. Sondern man muss eben die Aufmerksamkeit der Leute erreichen. Und es zwingt mich ja niemand, irgendwelche Patches[4], die mich nicht interessieren, anzugucken. [...] Wenn mich irgendjemand permanent nervt, dann hab ich auch keinen Bock, seine Patches zu reviewen. Und dann tue ich es auch nicht."
> (FS17-IT20-2)

Gemeinschaftliche Entwicklung bedeutet jedoch nicht, dass alle gleichermaßen über den Fortgang der Entwicklung bestimmen würden. So gibt es durchaus eine soziale Zuweisung von Verantwortlichkeiten, die auf einer zentralen Institution der Community basiert: ähnlich wie in wissenschaftlichen Produktionsgemeinschaften (Gläser 2006) gibt es auch in der „MSB"-Community eine persönliche *Autorenschaft* der Entwickler, die sich auf den Code bezieht, den sie (mit-) entwickelt haben:

> „Dabei ist natürlich ein wichtiger Punkt: also der DNS-Server ist meiner! Ich fühl mich natürlich auch verantwortlich wenn irgendjemand jetzt kommt und sagt: oh, hier ist ein Bug! Dann sag ich: oh, hoppla, den reparier ich mal lieber!"
> (FS17-Entwickler-Community-2)

„Autorenschaft" ist eine wichtige Institution innerhalb der Community, sie ist ein zentrales Prinzip der Koordination, das zwar informelle aber nichtsdestotrotz wirksame Zuweisungen von Verantwortlichkeiten beinhaltet, wie auch das folgende Zitat unterstreicht:

> „Also wenn jemand an der Software was ändert, wenn das was Neues ist, dann hat der gleichzeitig eine Aufgabe. Weil er oder sie ist dann nachher auch dafür zuständig. [...] Dann wird jemand, der da was reinschreibt, sich mit ihm abstimmen. Das heißt aber nicht, dass jemand das nicht darf. Sondern es ist dann nur Konvention, dass man sich dann abstimmt und dass man nicht in den Code von anderen Leuten reinschreibt. Es gibt immer riesen Geschrei [...] und Flamewars im Team, wenn jemand bei jemand anderem was reinschreibt und einfach Sachen ändert von anderen Leuten, ohne sie zu fragen."
> (FS17-IT20-1)

Autorenschaft konstituiert somit wechselseitige soziale Verpflichtungen: Entwickler sollen die Arbeit vorheriger Autoren respektieren und daran anschließen, indem sie dem Autor ein Mitspracherecht bei Änderungen einräumen. Der Nachvollzug der Autorenschaft wird durch die technische Kollaborationsplattform unterstützt, in der nicht nur der Code dokumentiert ist, sondern für jede Code-Zeile auch der Entwickler festgehalten wird, der sie geschrieben hat:

> „Also unser Versionskontrollsystem GIT kann an jeder Codezeile sagen: wer hat das zuletzt angefasst. Das ist einfach ein Basismechanismus der Versionskontrolle. Ich will immer nachgucken können, wer war es. Wen kann ich im Zweifel fragen, warum er das so gemacht hat. Wem kann ich auf die Finger hauen, wenn er Mist gebaut hat im Nachhinein. Das ist auch ganz entscheidend für irgendwelche Urheberrechtsgeschichten. "
> (FS17-IT20-2)

Der Erstautor hat letztlich sogar ein Vetorecht, wenn er dieses fachlich begründen kann. Zudem wird von Autoren erwartet, dass sie auf Fragen und Änderungsvorschläge anderer Entwickler antworten und diese

[4] Ein Patch ist eine Erweiterung des Programmcodes, häufig, um ein spezifisches Problem zu lösen oder eine neue Funktionalität bereitzustellen.

unterstützen, indem sie ihre Erfahrungen bei der Bearbeitung dieses Programmabschnitts teilen. Autorenschaft ermöglicht darüber hinaus den persönlichen Kontakt und Austausch der Entwickler mit dem Erstbearbeiter bzw. den Erstbearbeitern des jeweiligen Programmteils. Der Austausch geht oft so weit, dass der Erstautor konkrete Hinweise zur Programmierung gibt, wodurch die neue Person wichtige Einblicke in nicht explizierte Vorgehensweisen und Techniken gewinnen kann:

> „Und das ist aber eben der schöne Nebeneffekt, dass wir dann immer sehr genau wissen, wen kann ich fragen, warum irgendwas so funktioniert, wie es denn funktioniert. [...] Nennt sich GIT-Blame, das Kommando. GIT-Blame auf eine Datei – an jeder Zeile steht drin, das ist jetzt von dem und dem zuletzt geändert worden. Und dann kannst du auch sehr genau die Historie nachverfolgen, wie sah es vor der Änderung aus, wer hat die Finger da drin gehabt. Also das ist schon sehr ausgefeilt. Dieses sogenannte verteilte Versions-Kontrollsystem ist sicherlich eine reine Technik. Eine rein technische Lösung für ein soziales Problem sozusagen. Nämlich, ich will mich viel freier mit Leuten austauschen können über irgendwelche Codestücke, ich möchte das nicht alles über irgendeinen zentralen Server machen müssen."
> (FS17-IT20-2)

Aus der Verantwortlichkeit für bestimmte Teile des Programms entwickeln sich dadurch innerhalb der Community *informelle Teamstrukturen* von Entwicklern, die zusammen an einem bestimmten Programmteil arbeiten. Durch die relativ dauerhaften Strukturen und Aufgabenfelder entsteht ein eigener Arbeitsbereich mit einer gemeinsamen Praxis, die durch die vertraute Beziehung und engen Kontakt auch den Austausch von implizitem Wissen nachhaltig fördert:

> „Irgendwann hat man halt raus, wer sich wofür zuständig fühlt, wer wovon Ahnung hat, und dann kann man halt eben entsprechend fragen. Und dann wechseln Ideen quasi hin und her."
> (FS17-IT20-2)

Gleichzeitig gibt es auch einige stärker formalisierte Formen des Erfahrungs- und Wissensaustauschs, wie z.B. den Reviewprozess von Patches (neuem Code), in dem im Vier-Augen-Prinzip entschieden wird, ob ein neues Stück Code in die Community-Software aufgenommen wird:

> „Was wir gemacht haben ist, wir haben einen sogenannten Review-Prozess eingeführt. Das ist im Moment auch noch nicht *wirklich* formal, aber jeder hält sich dran. Bevor irgendeine Änderung, ein Patch reinkommt in 'MSB', müssen zwei Leute vom 'MSB'-Team zustimmen."
> (FS17-IT20-2)

Im Reviewprozess scheint der erfahrungsbasierte Charakter des nötigen Wissens deutlich auf. Obwohl der Prozess an sich recht formalen Kriterien genügt, gibt es kaum formalisierte und explizit formulierte Qualitätskriterien, vielmehr handelt es sich um kollektiv geteilte Ansichten über „guten Code" und „gutes Design". Diese sind z.T. in professionellen Standards, wesentlich häufiger aber in spezifischen Erfahrungen aus dem „MSB" Erfahrungsschatz begründet:

> „Wenn wir intern in 'MSB' Hauptspeicher managen müssen. Die Programmiersprache C hat dafür ein Mittel, Hauptspeicher bereit zu stellen. Diese Routine nennt sich Malloc. Malloc ist nicht mehr schön, das ist erstens langsam und das ist zweitens fehlerträchtig. Wir haben Talloc entwickelt, was potentiell sehr viel schneller ist und nicht fehlerträchtig. Deswegen bitteschön, benutzt doch Talloc. Es gibt auch heute noch Code in 'MSB', der Malloc benutzt. Neuer Code würde das nicht tun. Das ist eine Entwicklung der letzten zehn Jahre ungefähr. Es entwickeln sich quasi im Code irgendwelche Konventionen, das tut man einfach nicht mehr. Aber das ist nicht immer offensichtlich, solche Sachen."
> (FS17-IT20-2)

Diese geteilten Ansichten, Konventionen und Regeln werden diskursiv eingefordert und spätestens im Zuge des Reviews durchgesetzt:

> „Wir haben in 'MSB' ein readme.coding. Da stehen unsere Coding-Guidelines drin. Und alles, was da nicht drinsteht, da gibt es dann Diskussionen drüber. Vieles steht nirgendwo wirklich konkret niedergeschrieben und das wird dann in einem Patch-Reviewprozess diskutiert. Und dann gibt es Diskussionen darüber und entweder derjenige fügt sich dann oder sieht es ein oder der ist dann beharrlich und stur genug, es doch durchzuziehen. Gibt halt niemanden, der wirklich sagt, so und nicht so."
> (FS17-IT20-2)

Die Diskussion über solche Konventionen für die 'MSB' Entwicklung haben für Entscheidungsprozesse in der Community deswegen eine besondere Relevanz, weil diese in der Community im Prinzip auf Konsensfindung und Abstimmung unter den Mitgliedern beruhen. Letztlich werden Entscheidungen mehrheitlich über Abstimmung auf der Mailliste getroffen. In solchen Abstimmungen haben die Autoren und die Gründungsmitglieder ein Vetorecht, d.h. deren Gegenargumente müssen entkräftet bzw. widerlegt werden. Die persönlichen Beiträge der Entwickler sind dabei die zentrale Einheit bei der Zuweisung sozialer Anerkennung und entsprechendem Einfluss in der Community:

> „Also das ist quasi so eine Art natürliche Autorität, die sich da entwickelt. Weil man einfach seit 20 Jahren das Zeug macht, hat man auch einfach relativ schnell aus dem Rückenmark irgendwelche Argumente, warum das eine gute Idee ist und das keine gute Idee ist. Weil man einfach in viele Fehler selber schon reingelaufen ist, deswegen kann man halt

> relativ schnell eben irgendwelche Vorschläge [...] wegdiskutieren. [...] Wenn die Leute sich aktiv an diesen Diskussionen beteiligen, ergibt sich sowas wie eine natürliche Autorität. Das ergibt sich einfach. Ohne dass man da jetzt irgendwie ein formales Organigramm dafür bräuchte."
> (FS17-IT20-2)

Wie sich in diesem Zitat deutlich zeigt, gibt es in der Community stark meritokratische Einflüsse: wer viel einbringt, kann darauf setzten, dass er/sie in der Community mehr Einfluss bekommt. Formale Abstimmungen finden zwar auch statt, aber sie sind überlagert von meritokratischen Positionen:

> „Dann gibt es aber den Effekt in diesen Communities, dass natürlich Leute gleicher sind, die schon ein bisschen länger dabei waren. D.h. wenn [Name entfernt – PF/HH] und [Name entfernt – PF/HH] sagen, wir wollen das so, dann passiert das meist auch so. Und das Problem ist, dann gab es jede Menge Diskussion und dann gab es zum Teil schon Abstimmungen, mit einem Ergebnis, das dann aber den Ältesten nicht passte und dann wurde das wieder neu aufgerollt und geändert."
> (FS17-IT20-1)

Doch trotz dieser (informellen) meritokratischen Struktur kann keiner der Hauptentwickler eine Entscheidung gegen die anderen erzwingen, sondern im Konfliktfall muss in einem mehr oder weniger heftigen Diskussionsprozess ein Kompromiss gefunden werden.

> „Also es gibt Leute, die wirklich viel machen [...] die eine Meinung haben, die mehr Gewicht hat. Der Meinung von neueren Teammitgliedern [...] wird vielleicht nicht ganz so viel Gewicht gegeben. Also der muss mehr argumentieren, als jemand von den alten Hasen. [...] Aber, ich hab nicht den Eindruck, dass es irgendjemanden gibt, der wirklich in Anspruch nehmen kann, wirklich alleine aufgrund seiner Position im Projekt eine Entscheidung zu treffen, die nicht einem bestimmten technischen Bereich zuzuordnen ist."
> (FS17-Entwickler-Community-2)

Dies kann ein langwieriger und schmerzhafter Prozess sein. Entscheidungsprozesse in der Community können so erschwert und verlangsamt werden, wenn es zu bestimmten Punkten abweichende Meinungen gibt:

> „Diskussionen werden manchmal doch sehr emotional. Also ich hab den Eindruck, eine Menge von den schwierigen Entscheidungen sind deswegen schwierig, nicht weil irgendwie die technische Umsetzung davon schwierig wäre, sondern weil es halt doch von Leuten gemacht wird, und das heißt man muss eigentlich immer Egos navigieren und seltener wirklich die technische Herausforderung alleine."
> (FS17-Entwickler-Community-2)

Wie in diesem Abschnitt argumentiert, bietet die Community eine für den Austausch von Wissen förderliche soziale Praxis, die durch die gemeinsame Identität und geteilte Zielsetzungen, interne Institutionalisierungsprozesse geteilter Normen und Vorstellungen und interne Organisations- und Differenzierungsprozesse geprägt ist. Zusammen mit der technischen Infrastruktur der Entwicklungsplattform ermöglicht diese Praxis, dass verschiedene Akteure mit ganz unterschiedlichen Erfahrungs- und Wissensbeständen gemeinsam ein Produkt (weiter-) entwickeln. Bieten neue digitale technische Infrastrukturen damit ein gutes Potenzial für die Etablierung neuer Innovations- und Produktionsmodelle, so zeigt sich, dass diese in vielfacher Weise durch den unmittelbaren persönlichen Austausch und insbesondere durch face-to-face Treffen auf Entwicklerkonferenzen sozial eingebunden sind. Dies verweist auch auf die Fragilität, die für die skizzierten Community-Prozesse charakteristisch ist. Kommunikation und Kooperation sind nicht erzwingbar und können nicht vorausgesetzt werden. Gerade im Hinblick auf die Kooperation, bzw. auch die Konflikte zwischen Community und Unternehmen wird dieser Punkt relevant.

4.3. Spannungsverhältnisse zwischen Unternehmen und Community

Die beteiligten Unternehmen haben ein strategisches Interesse an der Entwicklung von 'MSB', ihre Geschäftsmodelle im betreffenden Geschäftsfeld (alle haben auch andere Geschäftsfelder) sind von der Entwicklung der Software abhängig. Sie befinden sich in einem schwierigen Spannungsverhältnis gegenüber der Community: dem selbstorganisierten und –gesteuerten Entwicklungsprozess der Community stehen die eigenen ökonomische Ziele gegenüber, die in der Community keine Rolle spielen. Dieser Konflikt führt zu zwei Problemen für die beteiligten Unternehmen.

4.3.1. Steuerungsprobleme und Lösungsansätze der beteiligten Unternehmen

Das erste Steuerungsproblem betrifft die Unmöglichkeit der genauen zeitlichen Planung des Entwicklungsprozesses in der Community: Charakteristisch für Community-basierte Entwicklungsprozesse ist, wie oben bereits diskutiert, dass es keine Möglichkeit gibt, eine Planung oder Roadmap zwingend festzulegen und umzusetzen. Dies hat zwei Gründe: zum einen verfügt die Community nicht über kalkulierbare Personalressourcen, sondern ist immer von der individuellen Bereitschaft von Entwicklern und deren oft begrenzten Zeitkapazitäten abhängig. Die meisten Entwickler der Community sind bei einem Unternehmen beschäftigt, ihre für „MSB" verfügbare Zeit ist daher auch vom häufig wechselnden und nicht einschätzbaren

Arbeitseinsatz bei den jeweiligen Unternehmen abhängig. Zum anderen sind die Entwickler in der Community grundsätzlich frei in ihrer Entscheidung, wann und woran sie mit welcher Intensität arbeiten. Ein Entwickler skizziert dieses Problem folgendermaßen:

> Und es gibt keine ganz klare Roadmap. [...] Ich sag mal so, wir haben eine Liste von Features, die wir gerne entwickeln möchten. Zum Teil ergibt sich sinnvollerweise eine zeitliche Abfolge, aber ohne zu sagen, dann und dann muss es fertig sein – das können wir nicht. Welche Reihenfolge genommen wird, das ist im Endeffekt den Leuten überlassen. Welche Wichtigkeit da ist, wird dann häufig von dem Arbeitgeber derjenigen bestimmt, die daran arbeiten.“ (FS17-IT20-3).

Es sind die zentralen Eigenschaften der Community, die oben als förderlich für den Wissensaustausch hervorgehoben wurden, die zu diesem Problem beitragen. Für die beteiligten Unternehmen erwächst daraus das Problem, nicht genau planen zu können, wann die Software über gewünschte funktionsfähige Features verfügen wird. Für den Dienstleister bedeutet dies z.B. eine grundsätzliche Ungewissheit, wann den Kunden bestimmte Leistungen angeboten werden können, für den Distributor stellt sich die Frage, wann die Distribution in bestimmten Umgebungen nutzbar sein wird.

Das zweite Steuerungsproblem bezieht sich auf die strategische Ausrichtung der gemeinsamen Arbeit, da auch die Diskussion über zukünftige Entwicklungsrichtungen innerhalb der Community weitgehend ergebnisoffen geführt wird. So stellt sich immer die Frage der unmittelbaren Anschlussfähigkeit der Lösung für den je eigenen Bedarf. Die beiden von uns untersuchten Unternehmen unterscheiden sich in dieser Hinsicht in ihrer Perspektive.

Das strategische Interesse des Dienstleisters an der inhaltlichen Entwicklung von ‘MSB’ ist kundengetrieben (auch wenn einzelne Entwickler andere individuelle Interessen haben mögen). Daher ist die Steuerungsproblematik aus Sicht des Dienstleisters kein zentrales Problem, solange er die Arbeiten für die Kunden selbst durchführt. Es ist dann ein Problem, wenn die Community nicht mehr wie erwartet funktioniert, sodass die Erwartungen und Bedarfe der Anwender/Kunden nicht erfüllt werden können. Insofern befindet sich der Dienstleister in einem Dilemma zwischen Community und Kunden und muss beide Ansprüche vermitteln:

> „Und wenn man mit Communities arbeitet, heißt es immer, du bist irgendwo im Nachteil. [...] Immer musst du abwägen, ist der Kunde benachteiligt, ist deine Firma benachteiligt, ist die Community benachteiligt, ist der einzelne Entwickler benachteiligt.“
> (FS17-IT20-1)

Entscheidungen, die in der Community getroffen werden, sind aus der Perspektive des Dienstleisters oft kompliziert und langwierig:

> „Die Community-Effekte dahinter sind natürlich schwierig. In Firmen kriegst du das schön gedeckelt, da gibt es Entscheider und die machen die Wege kurz. In Communities sind die Wege lang, die Software ist dann nachher aber durch alle Flamewars durch und die ist veröffentlicht und im Internet, jeder kann seinen Senf dazugeben, die ist in der Regel besser. In der Regel. Aber diese Abstimmungseffekte machen dann eigentlich die Problematik aus.“ (FS17-IT20-1)

Auch der Linux-Distributor hat seine eigene Roadmap für seine Produkte und die dafür benötigen Personalressourcen. Seine Produkte haben strategisch gewollte und wichtige Überschneidungen mit ‘MSB’. Andererseits gibt es andere Features von ‘MSB’ in denen er weder Kompetenz noch Interessen hat, die dennoch Bestandteil seines Gesamtproduktes sind. Daher ist auch der Linux-Distributor von den Problemen und Schwierigkeiten im Planungsprozess betroffen.

Es gilt insofern für beide Unternehmen, dass Leistungen, die kooperativ innerhalb der Community entwickelt werden, den je eigenen Aufwand für Entwicklungen reduzieren. Daher ist der Versuch, die Community strategisch zu beeinflussen, beiden Unternehmensstrategien inhärent. Der wichtigste Mechanismus, Einfluss auf die Entwicklung der Software und die Entscheidungen in der Community zu nehmen, besteht darin, führende Köpfe der Community als Mitarbeiter im Unternehmen zu beschäftigen. Beide Unternehmen des Samples betreiben eine strategische Personalpolitik, die darauf ausgerichtet ist, eigene Mitarbeiter als Hauptentwickler in der Community aufzubauen, oder gezielt Entwickler aus der Community einzustellen. Deren persönliche Beiträge zur Entwicklung der Software in der Community sind die zentrale Form der Beteiligung der Unternehmen. Diese entstehen zu wesentlichen Teilen im Rahmen ihrer Arbeitszeit und im Auftrag des Unternehmens.

Dabei besteht ein strategisches Interesse, die Arbeitsergebnisse aus dem Unternehmen so aufzubereiten, dass sie als Beiträge in der Community sinnvoll sind und „Upstream“[5] akzeptiert werden.

[5] “Upstream“ bezeichnet den Zustand, wenn ein Codeabschnitt in die geteilte Codebasis aufgenommen und damit allgemein verfügbar gemacht worden ist.

„Die Patches die wir machen, als Entwickler [im Unternehmen – PF/HH], die gehen sowieso alle irgendwie mal Upstream (in die 'MSB' Software). Und zwar unter persönlichem Copyright. Wir machen Patches, die wir an Kunden ausliefern und die wir Upstream einbringen." (FS17-IT20-2)

Die Unternehmen haben ein strategisches Interesse daran, dass die Arbeitsergebnisse der Entwickler im Unternehmen so weit wie möglich „Upstream gebracht" werden, wie der folgende Gesprächspartner des Dienstleisters ausführt:

„Also es macht überhaupt keinen Sinn, für einen Kunden zu arbeiten und die Sachen an [...] 'MSB'.org vorbei zu entwickeln. Es gibt manchmal den Fall, dass ein Kunde den expliziten Wunsch hat, dass wir eine Erweiterung machen, die nur für ihn proprietär ist. Ist aber in den seltensten Fällen sinnvoll, weil Upstream sich weiterentwickelt und jedes Mal, wenn der Kunde auf eine neue Upstream-Version will, muss seine private Änderung wieder angepasst werden. Langfristig erzeugt das nur Kosten und unnötige Arbeit. Viel besser ist es, Sachen die generell nützlich sind, direkt Upstream zu bringen. Dann haben zwar auch tendenziell alle anderen was davon, andererseits profitiert auch der Kunde davon, dass die Community das Ding weiter pflegt. Es wird automatisch mitgepflegt, es besteht eine Qualitätssicherung, über die man sich nicht mehr kümmern muss, weil es alle testen, sozusagen.
(FS17-IT20-3)

Daher sind die Unternehmen in begrenztem Rahmen bereit, den Entwicklern Arbeitszeit für die Mitarbeit in der Community einzuräumen.

4.3.2. Entwickler im Spannungsfeld zwischen Community und Unternehmen

Die Mitarbeit von Angestellten der Unternehmen als Hauptentwickler in der Community führt dazu, dass diese immer im Spannungsfeld unterschiedlicher Orientierungslogiken agieren. Sie sind zum einen den Weisungen der Unternehmen unterworfen, zum anderen sind sie als Teil der Community auch an die dort geltenden Normen und Organisationsformen gebunden. Und schließlich haben sie als Entwickler auch eigene inhaltliche und berufliche Interessen.

„Es ist nicht so, dass wir uns hinsetzen und einfach entwickeln wozu wir Lust und Laune haben. Das können wir vielleicht zu fünf bis zehn Prozent, je nach Auftragslage. Es gab auch schon Zeiten, da war mehr Zeit für sowas, Community-Pflege und einfach generelles 'MSB'-Voranbringen. Ist in letzter Zeit nicht so gewesen, da haben wir wirklich fast ausschließlich im Kundenauftrag gearbeitet, aber natürlich immer im Sinne des 'MSB'-Teams. (FS17-IT20-3)

Diese doppelten (oder sogar dreifachen) Orientierungsrahmen sind den Beteiligten präsent und müssen im alltäglichen Handeln ausbalanciert werden. Dies gilt für das Unternehmen genauso wie für die Entwickler selbst. Der Geschäftsführer des Dienstleisters beschreibt diese Konstellation folgendermaßen:

„Der einzelne Entwickler hat eine bestimmte Motivation, das MSB Team hat ein Ziel und das Unternehmen, für das der Entwickler arbeitet hat auch eine Strategie. Drei unterschiedliche Orientierungen. Und es wäre schön, wenn das parallel ist – ist es natürlich nie... [...] Aber manchmal geht es völlig auseinander und dann hat der einzelne Teilnehmer in dieser Community ein Problem."
(FS17-IT20-1)

Den Entwicklern ist dieses Spannungsfeld durchaus bewusst, wie folgende Aussage zeigt:

„Sagen wir mal so, ich habe immer quasi meinen 'MSB'-Team Hut (d.h. den Community-Hut, PF/HH) und meinen Unternehmens-Hut auf. Und auf den Mailinglisten poste ich immer @[Firmenname entfernt – PF/HH].de. Das heißt, das ist völlig klar, dass ich von [Firmenname entfernt – PF/HH] komme und wenn ich dann irgendwelche schlauen Kommentare abgebe. [...] Ich hoffe schon so ein bisschen darauf, dass die Leute, wenn sie nicht weiterkommen, dann irgendwann mal bei uns anrufen. Das ist die Hoffnung. [...] Es passiert ganz selten, vielleicht ein Mal im Jahr, dass ich jemanden, der in der Community-Mailingliste eine Frage stellt, tatsächlich direkt anmaile und sage, hier, ich habe jetzt keine Zeit mehr, mich kostenfrei drum zu kümmern, soll ich meinen Vertrieb auf dich loslassen? Das passiert auch, ist aber selten, und natürlich mit unterschiedlichem Erfolg. Klar, das ist sicherlich ein Zielkonflikt. [...] Aber wenn jemand quasi gute Patches liefert, die das Projekt weiterbringen, dann hat [Firmenname entfernt – PF/HH] letztendlich auch was davon. Dass 'MSB' halt weiter benutzt wird und wenn 'MSB' tot ist, ist die MSB-Abteilung von [Firmenname entfernt – PF/HH] auch tot. Also das ist immer schwierig zu fassen."
(FS17-IT20-2)

Dabei sind auch der unterschiedliche soziale Kontext und die daraus folgenden Interessenlagen sehr präsent, wie folgende Aussage eines anderen Entwicklers zeigt:

„Dass ich in meiner Rolle einerseits als [...] Angestellter, andererseits als MSB-Team-Mitglied – dass da hin und wieder mal irgendwelche Interessen nicht ganz kongruent sind, das gibt es definitiv. Und da muss ich dann im Einzelfall gucken muss, wie verhalte ich mich. Ich meine, es ist halt auch immer [Firmenname entfernt – PF/HH] ureigenstes Interesse, irgendwie mit MSB Geld zu verdienen. Aus der Software an sich, GPL, kann man kein Geld verdienen, das heißt wir versuchen als [Firmenname entfernt – PF/HH] immer wieder, Geschäftsmodelle rund um MSB zu finden. Und da muss man halt im Einzelfall gucken, würde das die Community toll finden und wenn sie es nicht toll findet, wie groß ist der Schmerz der Community damit? Können wir es uns noch erlauben als [Firmenname entfernt – PF/HH]? Oder

würde das halt irgendwie sehr böse aufstoßen? Muss man im Einzelfall immer gucken. Aber klar, gibt es, natürlich."" (FS17-IT20-1)

Der Gesprächspartner formuliert hier sehr deutlich, dass der Umgang mit OSS Communities ein ständiges Ausbalancieren und Reagieren auf die anderen Akteure voraussetzt. Und dass dieser Umgang weder primär wirtschaftlicher Logik folgen kann, noch primär der Logik der Community, sondern immer die unterschiedlichen – meist nicht wirtschaftlich-rationalen Orientierungen - der übrigen Akteure erahnen und in Rechnung stellen muss.

Es zeigt sich ein deutliches Spannungsverhältnis zwischen der hierarchischen Arbeitsorganisation im Unternehmen und den Spielräumen, die die Entwickler haben müssen, um ihre Rolle in der Community spielen zu können. Viele der Hauptentwickler der Community haben auch im Unternehmen eine wichtige Position, sodass sie in beiden Rollen Entscheidungen selbst verantworten können. Dies hat im Rahmen der Unternehmenshierarchie natürlich Grenzen: im Zweifel ist klar, dass für die Arbeit, die im Unternehmen und im Rahmen der Arbeitszeit geleistet wird, die Interessen des Unternehmens Vorrang haben, d.h. Aufträge im Unternehmen werden zuerst abgearbeitet. Wenn hier keine Zeit bleibt für Aufgaben in der Community, bleibt den Entwicklern nur die Verlagerung dieser Tätigkeiten in die Freizeit. Denn das grundlegende Prinzip der Arbeitsorganisation in den Unternehmen ist, dass Kundenaufträge immer vorgehen, d.h. sie werden zunächst abgearbeitet, erst dann können die Entwickler ihre eigenen Entwicklungen und Projekte auch im Rahmen der betrieblichen Arbeitszeit vorantreiben. Zusätzlich fließt allerdings oft noch ein mehr oder weniger großer Anteil von unbezahlter Arbeit der Entwickler ein. D.h. das konkrete Spannungsverhältnis (er)trägt letztlich der Entwickler oder die Anforderungen aus der Community werden auf einen späteren Zeitpunkt verschoben, was zu den bereits erwähnten Unwägbarkeiten im Entwicklungsprozess führt.

5. Fazit und Ausblick

Der vorliegende Aufsatz hat Open Source Communities mit Unternehmensbeteiligung als ein spezifisches Innovationsmodell untersucht, das durch den digitalen Charakter von Produkt und Arbeitsprozessen ermöglicht wird. Im Fokus standen dabei die Spannungsverhältnisse zwischen Unternehmen und Akteuren der Community, die aus den unterschiedlichen Logiken und Handlungsorientierungen hierarchischer und gemeinschaftlicher Governance erwachsen.

Die Fallstudie der OSS Community hat ergeben, dass die gemeinschaftlichen Formen der Wissensproduktion innerhalb der Community den Austausch nicht nur expliziten, sondern auch impliziten Wissens über Unternehmensgrenzen hinweg, positiv fördern können. Wie gezeigt werden konnte, ermöglichte die untersuchte OSS-Community eine für die verschiedenen Akteure anschlussfähige soziale Praxis, die neben der technischen Infrastruktur, vor allem auch durch die für Gemeinschaften typischen sozialen Eigenschaften einer gemeinsamen kollektiven Identität und Zielsetzung, der „Institutionalisierung des Kollektiven" (Dolata and Schrape 2014) und der internen Differenzierung und Organisierung, beeinflusst und ermöglicht wurde. Bestätigt unsere Fallstudie damit zum einen die Ergebnisse u.a. von Faraj et al. (2016), die ebenfalls auf die für das Teilen impliziten Wissens förderlichen Eigenschaften von Online Communities verweisen, so konnten anhand der starken Unternehmensbeteiligung im untersuchten Fall zum anderen auch neue Erkenntnisse über die Spannungsverhältnisse zwischen Communities und beteiligten Unternehmen gewonnen werden, ein Bereich, der bei bisherigen Untersuchungen überbetrieblicher und gemeinschaftlicher Innovationsprozesse häufig unterbelichtet blieb (vgl. Faraj et al. 2016; Lakhani et al. 2013). Wie gezeigt werden konnte, bildeten dieselben Merkmale, die für die gemeinschaftliche Wissensproduktion förderlich waren, gleichzeitig auch die Quelle z.T. erheblicher Spannungen zwischen den Akteuren in der Community und den beteiligten Unternehmen: Gemeinschaften beruhen auf selbstorganisierter und freiwilliger Kooperation und stehen dadurch immer latent in Konflikt mit der hierarchischen und an den eigenen Zielen orientierten Governance der Unternehmen. So erwiesen sich die z.T. langwierigen und in vielerlei Hinsicht auf persönlichen Beziehungen beruhenden Abstimmungs- und Diskussionsprozesse der Community für die Unternehmen auch als hinderlich, indem sie zu z.T. erheblichen Steuerungsproblemen, sowohl in zeitlicher, als auch strategischer Hinsicht führten. Gleichzeitig spielen in den konfliktreichen Diskussionsprozessen in der Community meist unausgesprochen auch die unterschiedlichen Unternehmenszugehörigkeiten eine Rolle. Zudem konnte gezeigt werden, dass die Fragilität der gemeinschaftlichen Prozesse immer ein Risiko für Unternehmen darstellen, führen persönliche Konflikte, mangelnde oder feindliche Kommunikation doch schnell zu einer Stagnation und erheblichen Verzögerungen. Allerdings zeigen unsere Befunde, dass die Unternehmen der Community nicht ähnlich hilflos gegenüberstehen, wie Dobusch und Quack (2011) in ihrer Untersuchung von auf Gemeinschaften beruhenden non-profit Organisationen herausstellen. Community und Unternehmen sind in unserem Fall stärker komplementär: die Open Source Community mit ihren spezifischen Koordinationsmechanismen fördert den Austausch und die Integration von verteiltem Wissen im Kontext gemeinschaftlich organisierter Innovationsprozesse. Im Gegenzug bietet die strategische Beteiligung der Unternehmen der Community den Zugang zu Personalressourcen, über die

die Community selbst nicht verfügt. Allerdings erzeugt der Zugriff auf Personalressourcen der Unternehmen gleichzeitig auch spezifische Abhängigkeiten der Community von den Unternehmen und damit eine Möglichkeit der Unternehmen, zumindest über die Weisungsmacht über die von ihnen beschäftigten Entwickler, auf die Entwicklung Einfluss zu nehmen. Charakteristisch für die hier untersuchte Konstellation ist daher, dass die beschriebenen Spannungen zwischen Community und Unternehmen von den involvierten Entwicklern der Unternehmen ausbalanciert werden mussten: sie befinden sich an der Schnittstelle der beiden Handlungsorientierungen, ihre Doppelrolle ist der Schlüssel für das Verständnis des spezifischen – und im untersuchten Fall sehr erfolgreichen - Zusammenspiels aus gemeinschaftlicher und hierarchischer Governance.

Zusammenfassend lässt sich also konstatieren, dass im untersuchten Fall einer OSS Community mit Unternehmensbeteiligung die internen Governancemechanismen der Community auf der einen und des Unternehmens auf der anderen Seite unabhängig voneinander bestehen bleiben: weder werden die Communities zu einem Anhängsel der kommerziellen Interessen der Unternehmen, noch sind die Unternehmen völlig machtlos den Launen und Unwägbarkeiten der Community ausgeliefert.

Mit der Untersuchung eines Falles beruhen unsere Ergebnisse selbstverständlich auf einem begrenzten empirischen Material. Dies ist vor allem vor dem Hintergrund der Varianz von OSS-Communities kritisch zu reflektieren. Wie Schrape (2015) herausarbeitet, gibt es eine breite Varianz an Communities, die sich besonders im Hinblick auf die in ihnen wirksamen Koordinationsmuster (hierarchisch/horizontal) und ihre Abhängigkeit von etablierten Unternehmen voneinander unterscheiden. Es ist zu erwarten, dass der Konflikt zwischen der eher horizontalen (gemeinschaftlichen) und der hierarchischen Governance (innerhalb der Unternehmen) umso stärker ist, je stärker Communities eine eigenständige gemeinschaftliche Institutionalisierung durchlaufen haben. Auch wenn OSS-Communities mittlerweile in der proprietären Softwareindustrie zur normalen Praxis gehören und nicht mehr als Vorbote revolutionärer Umwälzungen des Modus der Softwareproduktion angesehen werden können, wie Schrape (2015) zu Recht hervorhebt, ist die Einbindung von Community-basierter Wissensproduktion in kommerzielle Wertschöpfungsprozesse damit keineswegs trivialisiert worden, sondern beinhaltet nach wie vor z.T. erhebliche Herausforderungen für die beteiligten Unternehmen und vor allem die in der Community engagierten Beschäftigten.

6. Literatur

Ahrne, Göran und Brunsson, Nils (2011), 'Organization outside organizations: the significance of partial organization', Organization, 18 (1), 83-104.

Dahlander, Linus und Magnusson, Mats G. (2005), 'Relationships between open source software companies and communities: Observations from Nordic firms', *Research Policy,* 34 (4), 481-93.

David, Paul A. und Shapiro, Joseph S. (2008), 'Community-based production of open-source software: What do we know about the developers who participate?', *Information Economics and Policy,* 20 (4), 364-98.

Dobusch, Leonhard und Quack, Sigrid (2011), 'Interorganisationale Netzwerke und digitale Gemeinschaften: Von Beiträgen zu Beteiligung?', *Managementforschung,* 21, 171-213.

Dolata, Ulrich und Schrape, Jan-Felix (2014), 'Kollektives Handeln im Internet. Eine akteurtheoretische Fundierung', *Berliner Journal für Soziologie,* 24 (1), 5-30.

Faraj, S., von Krogh, G., Monteiro, E. and Lakhani, K.R. (2016), 'Special Section Introduction—Online Community as Space for Knowledge Flows', Information Systems Research, 27, 4, 668-684.

Giuri, Paola, Rullani, Francesco, und Torrisi, Salvatore (2008), 'Explaining leadership in virtual teams: The case of open source software', *Information Economics and Policy,* 20 (4), 305-15.

Gläser, Jochen (2006), *Wissenschaftliche Produktionsgemeinschaften die soziale Ordnung der Forschung*, Frankfurt am Main [u.a.]: Campus.

--- (2007), 'Gemeinschaft', in Arthur Benz, et al. (eds.), *Handbuch Governance - theoretische Grundlagen und empirische Anwendungsfelder,* Wiesbaden: VS Verlag, 82-92.

Heidenreich, Martin (1995), *Informatisierung und Kultur - Die Einführung und Nutzung von Informationssystemen in italienischen, französischen und westdeutschen Unternehmen*, Wiesbaden: VS Verlag.

--- (2003), 'Die Debatte um die Wissensgesellschaft', in Stefan Böschen and Ingo Schulz-Schaeffer (eds.), *Wissenschaft in der Wissensgesellschaft*, Wiesbaden: Westdt. Verlag, 25-55.

Krogh, Goeorg von (2011), 'Knowledge Sharing in Organizations: The Role of Communities', *Handbook of Organizational Learning & Knowledge Management*, West Sussex: Wiley, 403-33.

Lakhani, Karim R, Lifshitz-Assaf, Hila, and Tushman, Michael (2013), 'Open innovation and organizational boundaries: task decomposition, knowledge distribution and the locus of innovation', in Anna Grandori (ed.), *Handbook of economic organization: Integrating economic and organizational theory* (Cheltenham, UK; Northampton, MA, USA: Edward Elgar), 355-82.

Lange, Mark (2009) *Interoperability and Microsoft: Then and Now* [online text], SSRN eLibrary <http://ssrn.com/abstract=1523940>

Mateos-Garcia, Juan und Steinmueller, W. Edward (2008), 'The institutions of open source software: Examining the Debian community', *Information Economics and Policy,* 20 (4), 333-44.

O'Mahony, Siobhán (2007), 'The governance of open source initiatives: what does it mean to be community managed?', *Journal of Management & Governance,* 11 (2), 139-50.

O'Mahony, Siobhán und Lakhani, Karim R. (2011), 'Organizations in the Shadow of Communities', *Research in the Sociology of Organizations,* 33, 3-36.

O'Mahony, Siobhan und Lakhani, Karim R. (2011), 'Organizations in the Shadow of Communities', *Harvard Business School Working Paper 11-131*. <http://ssrn.com/abstract=1873989>.

Polanyi, Michael (1985), *Implizites Wissen*, Frankfurt am Main: Suhrkamp.

Schrape, Jan-Felix (2015) *Open Source Softwareprojekte zwischen Passion und Kalkül* [online text], Universität Stuttgart <http://www.uni-stuttgart.de/soz/oi/publikationen/soi_2015_2_Schrape_Open_Source_Softwareprojekte_zwischen_Passion_und_Kalkuel.pdf>

Streeck, Wolfgang und Schmitter, Philippe C. (1985), 'Community, Market, state - and associations? The prospective contribution of interest governance to social order', in Wolfgang Streeck and Philippe C. Schmitter (eds.), *Private Interest Government - Beyond Market and State*, London: Sage, 1-29.

Tsoukas, Haridimos (1996), 'The firm as a distributed knowledge system: a costructionist approach', *Strategic Management Journal,* 17, 11-25.

Von Hippel, Eric (2005) *Democratizing Innovation* [online text], The MIT Press <http://mit.edu/evhippel/www/books/DI/DemocInn.pdf>

West, Joel und Gallagher, Scott (2006), 'Challenges of open innovation: the paradox of firm investment in open-source software', *R&D Management,* 36 (3), 319-31.

West, Joel und O'Mahony, Siobhán (2008), 'The Role of Participation Architecture in Growing Sponsored Open Source Communities', *Industry and Innovation,* 15 (2), 145-68.

West, Joel und Lakhani, Karim R. (2008), 'Getting Clear About Communities in Open Innovation', *Industry and Innovation,* 15 (2), 223-31.

Wiesenthal, Helmut (2005), 'Markt, Organisation und Gemeinschaft als "zweitbeste" Verfahren sozialer Koordination', in Wieland Jäger and Uwe Schimank (eds.), *Organisationsgesellschaft*, Wiesbaden: VS-Verlag, 223-64.