Fallstudie horizontales elektronisches Arbeitsnetz/ Open Source- Projekt

Brand, Andreas

Preprint / Preprint
Arbeitspapier / working paper

Fallstudie horizontales elektronisches Arbeitsnetz/Open Source-Projekt

von Andreas Brand

Kontakt:

Dipl. Soz. Andreas Brand
Project electronic labourmarkets (PELM) / Projekt elektronische Arbeitsmärkte
Johann W. v. Goethe Universität / Johann W. v. Goethe University
Institut für Polytechnik und Arbeitslehre / Institute for polytechnic and labourscience
Email: A.Brand@em.uni-frankfurt.de

1. Einleitung

1.1. Einordnung des untersuchten elektronischen Arbeitsnetzes als Open Source-Projekt

Der „Softwaremarkt“ und damit die Softwareerstellung unterteilt sich in zwei Bereiche: In einen eher kommerziellen mit Softwareunternehmen und in einen eher freiwilligen Sektor mit Open Source-Projekten. Der kommerzielle Sektor besteht aus Unternehmen, die Lizenzen für ihre proprietäre Software verkaufen. Der voluntaristische Bereich dagegen ist eine große Community von ähnlichen Softwareerstellungsprojekten, wovon die Linux-Community die bekannteste und wahrscheinlich größte ist und auf dessen Software das untersuchte Open Source-Projekt im Allgemeinen aufsetzt. Diese Communities erstellen gemeinsam Software bzw. Quellcode, d.h. ein immaterielles Gut, über das Internet, wobei die Beteiligten das Produkt ihrer Arbeit Nutzern kostenfrei zur Verfügung stellen. Ein Kennzeichen dieser sozialen Bewegung ist, daß sie den Quellcode ihrer Software offenlegen, die Unternehmen dagegen im Allgemeinen ihre Software im binären, d.h. maschinencodierten, Status zur Verfügung stellen. Die Software wird normalerweise im Softwaremarkt über Lizenzen geschützt, die die Verfügungsrechte über die Software spezifizieren. Bei proprietären Softwarelizenzen wird so die Nutzung des Maschinencodes festgelegt. Im Falle der Open Source-Communities schützt auch eine Lizenz den offenen Quellcode. Der offene Quellcode mit seiner speziellen Lizenz grenzt so die Open Source-Community, aber auch Firmen, die Software unter diese Lizenz stellen, von anderen proprietären Softwareerstellern ab.
Es gibt verschiedene Arten von solchen Communityprojekten, die sich um bestimmte Softwarebereiche gruppieren. Dabei kann es zu Doppelungen von Software mit gleichen Funktionen kommen. Diese Open Source-Projekte teilen sich in sehr viele kleine Ein-Personen-Projekte und einige wenige große Projekte auf, bei denen über 100 Personen mitarbeiten. Das hier untersuchte Projekt gehört zu einer solchen Groß-Community, die einen Desktop entwickelt, der im Allgemeinen auf dem Unix- bzw. Linux-Betriebssystem aufbaut. Das Großprojekt mit seinem Desktop in Verbindung mit dem Unix- bzw. Linux-Betriebssystem und anderer Software, wie Officeprogrammen, ist als direkter Konkurrent von Microsoft zu sehen.

1.2. Methodik

Die Fallstudie als explorative Untersuchungsform wurde für die Arbeitsnetze, in diesem Fall das Open Source-Projekt, verwendet, da es zu diesen Netzen relativ wenig qualitative Studien gibt[1]. Die Fallstudie stützt sich dabei auf verschiedene Methoden. Zur Einarbeitung in die Thematik wurde die Dokumentenanalyse der recht vielfältigen Homepage des Open Source-Projekts angewendet. Zusätzlich wurden 62 Interviews von der Projekt-Homepage, die als Selbstauskünfte zu werten sind, da ausgewählte Mitglieder von einem Communitymitglied per Email befragt wurden, aus der Open Source-Community inhaltsanalytisch ausgewertet (s. Anhang A). Später wurden sechs qualitative, leitfadengestützte Experteninterviews geführt. Die Interviewpartner setzten sich nach dem folgenden Schema zusammen. Es wurden ein Gründungsmitglied, eine Person aus der Peripherie bzw. ein neuer Teilnehmer, ein stark in-

[1] Die Literatur über das Open Source-Phänomen wächst ständig an. Erste Fallstudien sind die beiden Fallstudien von Mockus et. al. (2002) und Jörgensen (2001).

volvierter Entwickler aus dem inneren Kreis, ein Nicht-Entwickler (Übersetzer und Dokumentierer), ein französischer Entwickler auch aus dem inneren Kreis und eine Entwicklerin, die zwischen Peripherie und inneren Kreis ein zu ordnen ist. Bis auf den französischen Entwickler waren alle Interviewten Deutsche. Dabei ist zu vermerken, daß drei der Interviewten aus dem Open Source-Promoteam[2] kamen. Es gab später drei „Nachinterviews" bzw. Klärungsgespräche mit der Entwicklerin und zusammen mit weiteren Entwicklern. Weiterhin wurden „teilnehmende" Beobachtungen mit informellen Gesprächen auf zwei Linux-Konferenzen und einer eigens vom Projekt organisierten Entwicklerkonferenz durchgeführt. Dabei wurde Fragen vertieft nachgegangen, die sich aus den anderen Gesprächen und Interviews ergaben und Gedächtnisprotokolle erstellt. Auf einer dieser Konferenzen wurde an die anwesenden Open Source-Entwickler des untersuchten Projekts ein quantitativer Fragebogen ausgeteilt, dessen Fragen auf ersten qualitativen Interviews basierte. Es kamen 55 ausgefüllte Fragebögen zurück (s. Anhang C). Zusätzlich wurde die Kommunikation auf zwei ausgewählten Mailinglisten über den Zeitraum von zwei-drei Monaten beobachtet (s. Anhang B).

1.3. Gliederung der Fallstudie

In Teil II wird erst auf die Struktur des Open Source-Projekts eingegangen. In dieser Struktur werden die Beziehungen zu seiner Umwelt, d.h. wie das Open Source-Projekt in die umgebende Community eingelassen ist, thematisiert. Dabei wird auf die besondere Funktion der Lizenzen eingegangen. Danach werden die Personen und ihre Funktion bzw. ihr Arbeitsinhalt dargestellt, um einen Überblick über die Beteiligten und somit das Netz zu haben.
Im Teil III wird sich auf die Arbeitsmarktfunktionen, Allokation, Leistungserstellung, Kontrolle und Gratifikation, konzentriert. Diese Arbeitsmarktfunktionen werden dabei an die Bedingungen der Arbeitsnetze als Netzwerke angepaßt.
Es existiert ein doppeltes Allokationsproblem, d.h. die Allokation teilt sich in eine externe und interne Allokation der Arbeitsleistungen bezüglich des elektronischen Arbeitsnetzes auf. Die externe Allokation beschreibt dabei den Ein- bzw. Austritt in das Großprojekt. Dies ist zu trennen von der internen Allokation, die die projektinterne Arbeitsverteilung beschreibt. Dabei ist die projektinterne Arbeitsverteilung nicht von der Erstellung der Arbeitsleistung zu trennen. Zusätzlich wird in dem Open Source-Projekt keine Gratifikation vom Großprojekt an die Beteiligten gezahlt, da die Beteiligung freiwillig ist. Es überwiegen nicht-monetäre Motivationsgründe. Somit wird die Allokation als Ein- und Austritt, die Leistungserstellung als Arbeitsablauf/-verteilung, die Kontrolle als Qualitätskontrolle gesehen und die Gratifikation mit der Motivation gleichgesetzt.
Im Kapitel über den Ein-/Austritt wird der Weg von der Aufnahme in die Community bis zum Eingang in den inneren Kreis skizziert. Dabei wird auf die Reputation genauer eingegangen, da diese hierbei eine besondere Rolle spielt. Beim Kapitel über die Leistungserstellung wird am Anfang der Ablauf der Produktion, später die Art und Weise der Arbeitsaufteilung dargestellt. Beim Kontrollkapitel wird darauf eingegangen, wie eine Kontrolle aussieht, wenn alle Personen freiwillige Arbeit leisten. Im Kapitel über die Gratifikation/Motivation wird der Frage nachgegangen, welche motivationalen Elemente Einfluß auf die Beteiligung und die Arbeit an dem Projekt haben. Am Schluß wird eine kurze Zusammenfassung der Fallstudie

[2] Dieses Team organisiert den Auftritt auf Ausstellungen in Deutschland.

gegeben. Im Anhang erscheinen erstens die quantitativen Auswertungen von Email-Interviews, die ein Community-Mitglied gemacht hat und die auf der Homepage des Projekts liegen. Zweitens sind dort die Auswertungen einer quantitativen Befragung von 55 Community-Mitglied zu finden.

2. Struktur der Akteure - Die interne Struktur des Open-Source-Projekts

In diesem Teil wird auf die innere und äußere Struktur des Arbeitsnetzes eingegangen. In dem Unterkapitel der inneren Struktur wird das elektronische Arbeitsnetz und die genutzte Informationstechnik beschrieben, um das Arbeitsnetz zu charakterisieren. In dem Unterkapitel der äußeren Struktur wird die Einbettung des Open Source-Projekts in äußere Strukturen, d.h. die Beziehungen zu anderen Open Source-Projekten oder Organisationen bzw. Unternehmen, aufgezeigt.

2.1. Entstehung des Open Source-Projekts

Gegründet wurde das Open Source-Projekt an einer deutschen Universität durch drei Studenten, die schon Erfahrung in der Koordination eines Open source-Projekts hatten. Der Name der neuen Community wurde absichtlich an den Namen eines bekannten, proprietären, aber relativ erfolglosen Programms angelehnt, um neuen Mitgliedern direkt eine visionäre Absicht zu signalisieren. Die ersten Mitstreiter an diesem Projekt wurden aus diesem Projekt aber auch über eine Ankündigung in einer Mailingliste gewonnen. Den Mitstreitern wurden erste, funktionierende kleine Programme als Anreiz zum Mitmachen angeboten. Diese Programme wurden von Anfang an von einer Person als Projektkoordinator koordiniert.

In der Gründungszeit wurden verschiedene Personen über verschiedene Maßnahmen für das Projekt angeworben. Es wurde z.B. eine Ankündigungsmail in einer internationalen Linux-bezogenen Mailingliste gesendet. Außerdem wurden die z.T. internationalen Mitentwickler aus einem älteren Open Source-Projekt der Gründer für das neue Projekt begeistert. Also war das Projekt von Anfang an international ausgerichtet. Später, nach ein paar Monaten, hat ein Entwickler das Projekt durch einen Artikel in einer wichtigen Computerzeitschrift in Deutschland bekannt gemacht[3].
Die Anzahl der Projektbeteiligten wuchs sehr schnell, wobei sich die Bekanntheit des Projekts dazu parallel erhöhte. Das Projekt wurde recht schnell in den Linux-Distributionen, wie SuSE, Mandrake, etc., aufgenommen. Durch die immer größere Verbreitung dieser Distributionen wurden und werden immer mehr aus- und inländische Entwickler angezogen. Auch die Gründung des Übersetzungsbereichs führte zu einem Zuwachs an ausländischen Beteiligten.
Mittlerweile ist das Großprojekt zu einer festen Größe in der Open Source-Community geworden, da es vom Umfang des Quellcodes und der Masse an Entwicklern zu einer der größten Open Source-Communities gehört.

[3] Dieser Artikel führte dazu, daß sehr viele neue Mitentwickler aus Deutschland dazu gewonnen werden konnten.

2.2. Umschreibung des Open Source-Projekts

2.2.1. Gemeinsames Produkt

In diesem Teil werden kurz die Eckdaten des untersuchten Open Source-Projekts dargestellt, um das Untersuchungsobjekt für eine weitere Analyse faßbar zu machen.
Die Hauptaufgabe und damit der Sinn des Bestehens der Community ist das Programmieren von Quellcode, speziell eines funktionierenden Desktops im Allgemeinen für das Unix- bzw. Linux-Betriebssystem[4] mit dazugehörigen Anwendungen. Es werden neben der Software auch Dokumentationen der Programme sowie deren Übersetzung erstellt. Der entwickelte Desktop mit seinen Programmen wird von einigen Millionen Menschen genutzt, wobei die reale Anzahl schwer einzuschätzen ist. Der Grund für das schwere Einschätzen liegt einerseits in der Möglichkeit, die Software kostenfrei aus dem Internet herunterzuladen. Andererseits gibt es diverse Distributionen von Distributoren bzw. Distributionsfirmen für Linux-Systeme mit Desktops, wozu auch ein Open Source-Distributor namens Debian mit kostenfreier Internet-Distribution gehört. Diese Distributionen werden meistens von einer Person erworben und dann an viele Personen weitergegeben, da die Lizenz bzw. die gesetzlichen Regelungen dies nicht begrenzt. In diesen Distributionen steht die Einfachheit der Nutzung im Vordergrund, weswegen der Desktop eine wichtige Nutzerschnittstelle darstellt. In diesem Großprojekt wird deswegen verstärkt auf die Gebrauchsfertigkeit für unerfahrene Endnutzer geachtet.

Die Software bzw. das gemeinsame Produkt existiert in zwei Zuständen. Erstens ist es in einem instabilen Zustand der kontinuierlichen Softwareentwicklung, zweitens in einem stabilen Zustand als offiziell veröffentlichte Version vorzufinden. Diese offiziell veröffentlichte Version wird auch Release genannt. Der Release soll dabei eine funktionstüchtige und möglichst fehlerfreie Version des bisher Programmierten sein, daß an die Endnutzer weitergegeben werden kann. Die Releases erscheinen mittlerweile in rascher Folge.

2.2.2. Tätigkeiten

Die Beteiligten der Open Source-Community sind überwiegend einzelne Menschen, keine Organisationen, die in ihrer Freizeit ohne Auftrag eines Unternehmens, also freiwillig, die Open Source-Software erstellen. Es gibt die Norm, daß die Personen ihren realen Namen in der Community angeben; Anonymität durch einen falschen Namen tritt selten auf.
Seit des Beginns des Großprojekts sind die Komplexität und die Anforderungen an die Programme, wie Graphik und Sound, angestiegen, so daß neue Aufgaben hinzukamen. Diese Aufgaben umfaßten das Dokumentieren und Übersetzen, aber auch das Design der Graphiken oder die Erstellung des Sounds. Im Zentrum steht aber die Softwareentwicklung, auf die alle weiteren Aufgaben und Tätigkeiten aufbauen und damit eine nachgeordnete Stellung haben.
Es gibt also verschiedene Arbeitsaufgaben bzw. Tätigkeitsbereiche in der Open Source-Community, wie Softwareentwickler, Dokumentierer, Übersetzer, Grafik- und Soundersteller, Community-„Journalisten“ und wiederholte Fehlerberichterstatter.
Die Erstellung der Software wird von Entwicklern vorgenommen, die gleichzeitig auch Nutzer ihres Programms sind. Daneben gibt es sehr spezifische technische Funktionen, die dem

[4] Das Desktop kann auch bei anderen Betriebssystemen, wie Microsoft Windows, verwendet werden.

Erhalt der Community dienen, wie die Pflege der Mailinglisten, des Dateiablagesystems und weiterer Entwicklungstools. Diese Funktionen werden oft von Entwicklern übernommen, die für die Arbeit an dem Projekt von Distributoren angestellt wurden, da sie viel mehr Zeit durch ihre Arbeit in der geschäftlichen Arbeitszeit und Freizeit ins Projekt investieren als nur Freizeitarbeiter.
Die Unterscheidung in die jeweiligen Tätigkeitsbereiche, wie Softwareentwickler, Dokumentierer, Übersetzer, etc., ist schwierig, da es Überschneidungen gibt. Z.B. gibt es Entwickler, die übersetzen, oder Übersetzer, die Software schreiben.
Die Übersetzer übersetzen die Texte in einem Programm und die Programmdokumentationen, d.h. sie sind von der Arbeit der Entwickler abhängig. Das Ziel der Übersetzer ist eine einheitliche Übersetzung und unterscheidet sich vom Ziel der Entwickjler, die eine qualitativ hochwertige und einheitliche Software erstellen möchten. Trotz dieser Abhängigkeit sind die einzelnen Tätigkeiten des Programmierens und des Dokumentierens/Übersetzens relativ stark voneinander getrennt, was sich auch in einer mehr oder weniger starken Abgrenzung zwischen Entwicklern und Dokumentierern/Übersetzern niederschlägt. Diese Abgrenzung wird durch die Nutzung unterschiedlicher Mailinglisten bestimmt, die meistens nur von Entwicklern oder nur von Dokumentierern/Übersetzern genutzt werden. So haben Dokumentierer/Übersetzer wenig Einfluß auf die weitere Entwicklung der Software, genauso wie die Entwickler keinen Einfluß auf die Übersetzung haben. Außerdem zeigte sich ein geringerer sozialer Status der Dokumentierern/Übersetzern gegenüber den Entwicklern, da die Dokumentierer/Übersetzer an nachgeordneten Aufgaben bzw. Tätigkeiten arbeiten. Dies kann ein Grund für den Trend sein, daß Übersetzer zu Entwicklern werden, die Übersetzungsteams also Mitwirkende an die Entwickler verlieren.
Die Fehlerberichterstatter bestehen erstens aus Entwicklern und Dokumentierern/Übersetzern, die Fehlerberichte über Fehler in ihrem Projekt senden. Diese Projektbeteiligten schreiben relativ oft Fehlerberichte. Zweitens gibt es Fehlerberichte von externen Nutzern, die z.T. nur einmal einen Fehlerbericht abgeben, aber zur Masse der Fehlerberichte gehören. Diese externen Nutzer können wegen der einmaligen Fehlerberichterstellung und der fehlenden weiteren Kommunikation mit den Projektbeteiligten nicht direkt zum Projekt gezählt werden, weswegen sie hier nur kurz erwähnt werden.

2.2.3. Aufbau des Open Source-Großprojekts

Insgesamt sind an dem Großprojekt geschätzt ca. 1000-1200 Personen beteiligt, wovon die Programmierer ca. zwei Drittel ausmachen. Nach der Anzahl der Eintragungen im Dateiablagesystem kann von einer Größenordnung von ca. 800 Entwicklern, Übersetzern und Dokumentierern ausgegangen werden, wobei noch mal 200-400 nicht-registrierte Mitwirkende vor allem von Dokumentierern/Übersetzern dazu zu rechnen sind. Von dieser Zahl sind ca. 2/3 Entwickler und 1/3 Übersetzer und Dokumentierer, ganz wenige (1-2%) übernehmen journalistische Tätigkeiten oder die Bearbeitung von Graphik oder Sound/Musik. Dabei gibt es weit weniger Dokumentierer als Übersetzer[5].

[5] Die Arbeit der Dokumentierer und Übersetzer ist mittlerweile sehr weit gediehen. So schrieb eine Person „The docs really are now very up-to-date and in a lot of countries, several hundreds of individuals work very hard on translation“ (Aus Selbstinterviews Anhang A).

Die Leistungserstellung findet in (Sub-)Projekten statt, die im Durchschnitt aus 3-4 Personen bestehen. Es gibt aber auch Projekte mit bis zu 30-50 Personen[6]. Die Personen sind im Durchschnitt bei 2 Projekten im Projekt beteiligt, arbeiten aber auch an bis zu 10 verschiedenen mit (s. Anhang C).
Die konsensual festgelegte „Amtssprache" des Projekts ist amerikanisches Englisch. In dieser Sprache werden Texte in der Software, die bei der Nutzung erscheinen, und in den Mailinglisten geschrieben. Die Übersetzer übersetzen im Allgemeinen von dieser Standardsprache in ihre jeweiligen Landessprachen und gruppieren sich dafür zu Teams um eine Landessprache. Die meisten Übersetzer sind in den Sprachen Französisch und Deutsch, den aktivsten Teams, zu finden. Bei diesen gibt es je an die 100 Übersetzer, die aber nicht alle im Versionsmanagementsystem[7] eingetragen sind. Zugang zum Versionsmanagementsystem haben meistens nur die Projektleiter bzw. Subprojektleiter in einem Übersetzungsteam. Weitere sehr aktive Teams sind mit den restlichen gängigen europäischen Sprachen verbunden. Bei exotischen Sprachen sind nur wenige, manchmal nur eine Person, beteiligt.

Die Projektbeteiligten können in zwei Teilbereiche unterteilt werden. Der eine Teilbereich wird durch die durchstrukturierte Erwerbstätigkeit, der andere durch Flexibilität durch das Studium oder Erwerbslosigkeit bestimmt. In dem Teilbereich der Erwerbstätigen gibt es eine kleine Gruppe von ca. 10-20 Personen, die von verschiedenen Linux-Distributeuren für die Arbeit an dem Open Source-Produkt angestellt wurden[8]. Ein größerer Teil arbeitet wiederum in open source-nahen Unternehmen, die Dienstleistungen rund um Open Source bzw. Linux anbieten. Die Mehrheit dagegen ist in IT-Unternehmen ohne Open Source-Bezug angestellt. Ein paar wenige Personen haben eigene kleine Unternehmen gegründet, die einerseits direkt mit dem Open Source-Projekt verbunden[9] sein können oder allgemeine Linux-Dienstleistungen anbieten. Von dem Teilbereich der flexiblen Zeiteinteilung machen die große Mehrheit Studenten aus, die sich u.a. wegen ihrer freiverfügbaren Zeit an dem Großprojekt

---

[6] Projekte werden in Unterprojekte mit eigenen Unterprojektleiter unterteilt, so daß hier eine begriffliche Ungenauigkeit zu vermuten ist.

[7] Im Versionsmanagementsystem bzw. Dateiablagesystem befindet sich die zu entwickelnde Software bzw. die Dokumentationen und Übersetzungen. Weitere Erklärungen zum Versionsmanagementsystem siehe Kap. 3 zu der Informations- und Kommunikationstechnik.

[8] Dieser Personenkreis wird von Unternehmen für die Arbeit an dem Großprojekt bezahlt und angestellt. Dieser Kreis hat dabei sein Hobby zum Beruf gemacht. Sie haben diese Art der Arbeit aus Idealismus, Spaß und Interesse an der Arbeit gewählt. Die meisten Firmen, besonders die Distributeure, haben einen Interesse an der Weiterentwicklung an der Software des Projekts. Sie haben einen guten Kontakt zur Community, weswegen sie die Personen direkt für Jobangebote ansprechen. Eine andere Vorgehensweise ist bekannte Entwickler auf ideale Kandidaten anzusprechen. Die Auswahl erfolgt dann nach den Kriterien Vorstellung der Firma, Verfügbarkeit und Qualifikation des Entwicklers. Die Arbeitszeit für das Großprojekt wird im Arbeitsvertrag festgelegt. So kann die Arbeitszeit für das Großprojekt die volle Arbeitszeit, aber auch Tages- bzw. Stundenkontingente sein. Ihr Einkommen liegt gegenüber anderen IT-Professionals eher im mittleren Bereich. Die Jobangebote gehen fast ausnahmslos an die Mitentwickler mit hoher Reputation.

[9] Z.B. nutzt ein Unternehmen die intime Kenntnis der Open Source-Software und übernimmt Arbeitsaufträge vom Staat für die Erweiterung der Projektsoftware zum Zwecke des Einsatzes in Verwaltungen.

beteiligen. Die Erwerbslosen dagegen sind auf Arbeitssuche und polieren mit der Arbeit an dem Open Source-Projekt ihr Wissen auf. Z.T. sind Zivildienstleistende oder Schüler darunter.
Die Mehrheit aller Projektbeteiligten arbeitet in ihrer Freizeit für das Open Source-Projekt. Die Arbeitszeit der Freiwilligen für das Projekt variiert deutlich, so daß sie zwischen ein paar Stunden pro Woche bis zu mehreren Stunden pro Tag reichen kann. Dabei kann der niedrigste Wert bei 0,5 Stunden und der höchste bei 90 Stunden pro Woche liegen (s. Anhang A, C). Die Arbeitszeit hängt von den sozio-ökonomischen Rahmenbedingungen der Projektbeteiligten ab. Dazu zählt z.B. die Höhe und Flexibilität der Freizeit, aber auch die Verpflichtungen in der Freizeit. So arbeiten Studenten und Schüler mit eher hoher und flexibler Freizeit mehr für das Projekt als Erwerbstätige mit projektfernen Aufgaben. Zusätzlich haben Projektbeteiligte mit Familie und Kindern weniger Projekt-Arbeitszeit als andere (Anhang A, B).
Kontinuierlich viel Arbeit wird von bestimmten Personen erbracht, die für das Großprojekt von einem Unternehmen bzw. Distributor angestellten wurden. Diese Open Source-Programmierer arbeiten entweder Vollzeit für das Open Source-Projekt oder sind für einen bestimmten Anteil der Arbeitszeit dafür freigestellt. Diese angestellten Open Source-Entwickler arbeiten z.T. auch neben ihrer Erwerbsarbeit in ihrer Freizeit an dem Open Source-Projekt. So ist die Durchschnittsarbeitszeit der für das Großprojekt Angestellten höher als die Durchschnittsarbeitszeit der Personen, die in ihrer Freizeit daran arbeiten (s. Anhang A).

Das Großprojekt besteht aus einer relativ homogenen Gruppe von z.T. noch ungebundenen, hochqualifizierten männlichen IT-Professionals. Die Mehrzahl der Projektbeteiligten ist zwischen Anfang bis Ende 20 Jahre alt[10]. Die Beteiligten sind vor allem Männer, die Frauen sind eine sehr kleine Minderheit. Die Großprojekt-Beteiligten kommen zum überwältigenden Teil aus dem technischen Bereich, d.h. aus dem Ingenieur- oder Informatikbereich. Fast alle Studenten und die große Masse an Erwerbstätigen haben Informatik- oder einen IT-nahen Studiengang[11] studiert oder sind gerade in einem solchen eingeschrieben. Es gibt nur ganz wenige Nicht-Akademiker und Nicht-Techniker. Ein großer Teil der Erwerbstätigen arbeitet als Softwareingenieure oder als Systemadministratoren. Es gibt aber auch einige Forscher bzw. Doktoranden, die an der Universität angestellt sind.
Der familiäre Status der Projektbeteiligten kann nach Ledige und Verheiratete unterteilt werden. Eine Mehrheit der Projektbeteiligten ist ledig und hat keine Kinder. Darunter fallen fast alle Studenten. Nur eine Minderheit der Verheirateten hat mehr als ein Kind.

Die räumliche Verteilung der Beteiligten an der Open Source-Community ist sehr unausgewogen. Der überwiegende Teil (ca. 50%) kommt aus dem westlichen Teil von Europa, wobei besonders viele aus Deutschland dabei sind. Diese Häufung der deutschen Beteiligten kommt durch die Gründung des Gesamtprojekts in Deutschland und der Werbung, die in Deutschland dafür gemacht wurde. Der Rest der Beteiligten an der Community kommt vor allem aus Nordamerika, Australien und Osteuropa, wobei es vereinzelte Beteiligte aus Asien und dem

[10] Es gibt zusätzlich einige Programmieranfänger im Alter von 14 bis 19 Jahren und einige, die im Alter von Ende 20 Jahre bis Mitte/Ende 30 Jahre sind. Sehr wenige sind über 40 Jahre alt.

[11] IT-nahe Studiengänge sind alle Ingenieurwissenschaften, Mathematik und Physik.

Orient gibt. Es gibt einzelne Beispiele von Projektbeteiligten, die aufgrund des internationalen Kontaktes des Projekts und wegen projektnahen Jobangeboten das Land wechselten (s. Anhang A). In den Ländern gibt es aber vereinzelte Cluster von Projektbeteiligten, die eine räumliche Nähe zueinander haben. Die Geschäftssprache des Projekts ist amerikanisches Englisch, um eine große Anzahl an Menschen zu erreichen und einen hohen Grad an Internationalität zu erreichen.

Zusammengefaßt wurde das Open Source-Projekt an einer deutschen Universität gegründet und hat deswegen in Deutschland und in Europa die meisten Entwickler. Kleine Programme wurden als Initiativbausteine eingebracht, woraus sich heute zueinander hierarchisch gegliederte Programmodule herausgebildet haben. Das Projekt als solches steht im Zentrum und ist von vielen kleinen Mikroprojekten umgeben. Das Groß-Projekt besteht vor allem aus 20-25jährigen, männlichen IT-Fachleuten, die z.T. noch studieren. Der Großteil der Beteiligten erstellt freiwillig in seiner Freizeit die Software, wobei die Angestellten von Distributoren für das Open Source-Projekt arbeiten und oft community-erhaltende Funktionen inne haben. Es gibt verschiedene Tätigkeitsbereiche, wobei Entwickler auf der einen Seite und Übersetzer/Dokumentierer auf der anderen Seite die größten Gruppen bilden.

3. Struktur der technischen Agenten und des technischen Umfelds - Die technischen Werkzeuge des Open Source-Projekts

Die alltägliche Produktion und Kommunikation läuft nur über die Informations- und Kommunikationstechnik ab. Es gibt relativ selten persönliche Treffen wegen der hohen geographischen Streuung. Die Treffen unterstützen somit die alltägliche informationstechnische Kommunikation. Es gibt aber auch regionale Cluster oder Mikrogruppen von Projektbeteiligten, die am gleichen Ort bzw. in der gleichen Region oder Stadt wohnen und sich deswegen öfters persönlich Treffen[12]. Dies gilt vor allem für Deutschland. Die Informations- und Kommunikationstechnik ist somit die Basis für das Entstehen und Bestehen des Open Source-Projekts.

Die wichtigsten Werkzeuge für das Open Source-Projekt sind die Homepage des Projekts, Emails über Mailinglisten, ein Chatsystem (Inter Relay Chat (IRC)), ein Versionsmanagementtool, ein Fehlermeldesystem und einige weitere entwicklungsunterstützenden Tools[13]. Die verschiedenen Werkzeuge werden durch einen beteiligten Verantwortlichen kontrolliert, d.h. es gibt bestimmte Posten in der Community, die die weitere Funktionsfähigkeit der Werkzeuge und somit der Community aufrecht erhalten. Solche Posten gibt es z.B. für Ho-

[12] Ein Beispiel hierfür ist das „Promoteam“, die für das Marketing, aber vor allem für die Organisation von Messeständen des Projekts zuständig ist. Diese Gruppe kommuniziert zur Vorbereitung von Messeauftritten nicht nur über Emails, sondern auch über Telefon und trifft sich manchmal für Beratungen.

[13] Es werden viele selbstgeschriebene Programme benutzt, die einige Funktionen von Groupware enthalten. Mailinglisten verteilen an sie gesandte Email räumlich und zeitlich asynchron von einer Person an die Personen, die sich z.T. anonym, in der Adreßliste aufnehmen ließen. Das Chatsystem erlaubt die räumlich asynchrone, zeitlich synchrone schriftliche Kommunikation. Dagegen werden aus Kostengründen keine Videokonferenzen genutzt. Nicht jeder hat die nötige Hard- und Software und Bandbreite um dieses Tool zu verwenden. Es werden außerdem keine Zeiterfassungssysteme in dem Großprojekt verwendet.

mepage, Fehlerberichtsystem, Versionsmanagementsystem und bestimmte Mailinglisten. Diese Personen verteilen u.U. auch Schreibberechtigungen für die jeweiligen Werkzeuge. Die Tools wurden z.T. von anderen Open Source-Projekten übernommen oder selbst erstellt.
Diese Werkzeuge können in den Bereich Information und Kommunikation und in den Bereich Produktion eingeteilt werden, wobei Homepage, Mailinglisten/Chatsystem zum ersten, das Versionsmanagement und das Fehlermeldesystem zum zweiten gehört.

3.1. Kommunikation

Die Kommunikation erfolgt normalerweise über die Mailinglisten oder Chat. Dabei sind die Mailinglisten das wichtigste Kommunikationsorgan im Projekt, auf dem die meisten Diskussionen und Entscheidungen über die Softwareproduktion gemacht werden. Die Mailinglisten sind nach Themen geordnet. Es gibt Mailinglisten für die allgemeine fachliche Diskussion im Gesamtprojekt, aber auch spezielle Mailinglisten für die spezifische Diskussion in den verschiedenen Arbeitspaketen bzw. für Themen mit spezifischen Aufgaben. Jedes größere (Sub-)Projekt im Gesamtprojekt hat eine eigene Mailingliste. Die Mailinglisten für die allgemeine fachliche Diskussion können in Listen für Nutzer, für die allgemeine Entwicklungsrichtung des Open Source-Projekts und in eine Mailingliste für alle restlichen Themen, die nicht abgedeckt wurden, unterteilt werden. Die speziellen Mailinglisten werden dann eingesetzt, wenn sich mehrere Personen bei einer Aufgabe, d.h. Entwicklung, Übersetzung, o.ä., abstimmen müssen. Diese Mailinglisten drehen sich z.B. um Subprojekte eines speziellen Softwarebereichs und um bestimmte (Sonder-)Aufgaben, wie Marketing, Übersetzerteams, Dokumentation, etc. So haben größere Softwaresubprojekte und größere Übersetzungen eigene Mailinglisten.
Die Listen sind auf konkrete Themen zugeschnitten, die nur dort besprochen werden dürfen[14]. Zuwiderhandelnde werden meistens ignoriert oder zurecht gewiesen. Eine Ausnahme stellt eine Mailingliste dar, die für die restlichen Themen ist, die von anderen Mailinglisten nicht abgedeckt werden. Hier wird auch über persönliche Vorlieben gesprochen. Mailinglisten sind zwar zeitlich und räumlich asynchron, doch erfolgen die meisten Diskussionen mit geringen zeitlichen Abständen von wenigen Minuten bzw. Stunden und sind meistens innerhalb weniger Stunden oder Tage beendet.
In den Mailinglisten kommunizieren, relativ gesehen, nur wenige, d.h. es gibt einen hohen Anteil an Angemeldeten, die niemals in Erscheinung tritt. So sind in den Mailinglistenverteilern meistens sehr viel mehr Projektbeteiligte eingetragen, die das geschehen verfolgen, als Personen, die aktiv dort kommunizieren. Dadurch ist ein großer Wirkungskreis/Wissensbasis bzw. eine hohe Erreichbarkeit von Gleichgesinnten möglich.
Für die meisten Mailinglisten gilt, daß der Lese- und Schreibzugriff für jeden möglich ist. In den Mailinglistenverteilern stehen dabei nur die Emailadressen der Angemeldeten. Es gibt aber für eine Mailingliste mit wichtigen Kernthemen nur einen begrenzten Zugang über eine Vermittlungsperson zum Schreibzugriff, um dem hohen Nachrichtenverkehr auf den Listen Herr zu werden und die Aufmerksamkeit auf wichtige Beiträge lenken zu können. Dieser „In-

[14] Es gibt eine Ausnahme, bei der alle anderen Themen besprochen werden können. Diese Mailingliste soll für den persönlichen Kontakt zwischen den Beteiligten sorgen. Hier werden emotionsbeladene und politische Themen kontrovers diskutiert.

formationsoverload“ überforderte die Aufmerksamkeit der Empfänger, die sich nun nach dem Absendernamen der Mailingliste richten. Für strategische Entscheidungen gibt es eine „geheime“ Mailingliste mit stark begrenztem Zugriff, d.h. die technische Grenze mit begrenztem Schreib- und Lesezugriff geht mit einer sozialen einher.
Das Chatsystem wird für eine synchrone Kommunikation verwendet, da dort die Anwesenden zeitgleich im System und identifizierbar sind. Außerdem ist das Medium flüchtig, da geschriebenes normalerweise im digitalen Nirwana verschwindet. Der Kreis der Empfänger kann durch technische Möglichkeiten klein gehalten werden.

3.2. Information

Zur Information von neuen Teilnehmern oder Sponsoren, aber auch zum Speichern von Wissen für die Community fungieren die Homepage, aber auch die Archive der Mailinglisten. Auf der Homepage werden wichtige Terminplanungen[15], Arbeitslisten, Anweisungen/ Einleitungen (diverse FAQs[16]), Regelungen[17], etc. zur Programmierung, Dokumentation oder Übersetzung veröffentlicht. Zusätzlich werden auf der Homepage die Zuständigen für bestimmte Arbeitseinheiten, Softwareteile, etc. genannt und einige Beteiligte des Projekts stellen sich durch ausgefüllte Interviews vor. Diese standardisierten Email-Interviews mit offenen Antwortkategorien sollen die Anonymität zwischen den Mitarbeitern aufbrechen. Weiterhin werden Programme und ihre Programmierer vorgestellt, um für diese Mitentwickler zu gewinnen. Auf der Homepage gibt es für wichtige Neuigkeiten über die Entwicklung des Projekts eine moderierte News-Seite, auf der selbsterstellte aber auch fremde Informationen erscheinen, die kommentiert werden können. Es gibt noch einen Service von einem externen Unterstützungsprojekt, der die wichtigsten Mails einer bestimmten Zeitperiode zusammenfaßt und veröffentlicht bzw. auf einer Mailingliste sendet. Außerdem besteht die Möglichkeit sich über die Mailinglisten-Archive über frühere Diskussionen und Entscheidungen zu informieren.

3.3. Produktion

Die Produktion wird vor allem durch ein Dateiablage- bzw. Versionsmanagementsystem im Internet[18] getragen, das den Quellcode verwaltet. Dieses Versionsmanagementsystem erlaubt ein räumlich und zeitlich synchrones und asynchrones Arbeiten einer sehr großen Anzahl an Projektbeteiligten an demselben Quellcode, wobei auf dem eigenen Rechner vor Ort programmiert und dann in die Dateiablage zurückgespielt wird[19]. In diesem System ist der ge-

[15] Die Terminplanung des nächsten Releases, d.h. der offiziellen, stabilen Version, wird dort veröffentlicht. Dies wird Koordination der verschiedenen Gruppen genutzt. (Dazu weiteres im Kapitel 3)

[16] FAQ heißt eigentlich frequently asked questions und ist eine Art Bedienungsanleitung für häufig gestellt Fragen.

[17] z.B. styleguides zur Programmierung, zum Aussehen der Programme, zu Graphiken (Icons), etc.

[18] Das Dateiablagesystem liegt auf speziellen Servern, die von Firmen gesponsert werden.

[19] Probleme entstehen, wenn zwei den gleichen Quellcode herunterladen, diesen verändern und dann wieder fast zur selben Zeit zurückschreiben. Der letzte, der zurückschreibt, hat dann das Problem, daß sein Code nicht mehr zum überschriebenen Code paßt. Dadurch gerät der Quellcode durcheinander und der letztere muß das Durch-

samte Quellcode archiviert, so daß alte Zustände wiederherstellbar sind. Daneben sind aber auch Dokumentationen über Programme, Übersetzungen, Bilder/Graphiken und Webseiten enthalten. Es gibt drei Großbereiche innerhalb des Versionsmanagementsystems. In einem der Bereiche steht der Quellcode, der offiziell in Distributionen veröffentlicht wird. In einem weiteren wird Code in Form von funktionstüchtigen Programmen erstellt, der über das Internet frei zugänglich ist, und in einem anderen sind alle unfertigen, experimentellen Programme versammelt. In diesem Bereich wird experimenteller Code von einigen Entwicklern neben ihren normalen Unterprojektbeteiligung erstellt. Außerdem gibt es einen Bereich für die Homepage. Die Sprachen und Dokumentationen sind jeweils in einzelnen eigenen Bereichen im Dateiablagesystem abgelegt. Diese Bereiche stehen in einer Hierarchie zueinander, d.h. der zentrale Bereich, aus dem der Release genommen wird, ist am wichtigsten und alle anderen Dateiablagebereiche sind diesem nach ihrer Wichtigkeit für die alltägliche Softwareerstellung nachgelagert. Quellcode, der nicht in diesem Ablagesystem erstellt wird, gehört nicht zum Großprojekt, auch wenn er auf Softwareteilen aus dem Großprojekt basiert.
Innerhalb des Dateiablagesystems ist der Quellcode modular aufgebaut, d.h. der Code wurde in kleine Pakete aufgeteilt, so daß jedes Programm einen eigenen Platz im System hat. Die Software besteht aus einer Basisbibliothek, die von einer Firma zur Verfügung gestellt wird, Haupt- und Nebenbibliotheken und ausführenden Programmen, die auf den Bibliotheken aufbauen. Diese Bibliotheken und Programme stehen in einem hierarchischen Verhältnis zueinander, wobei die Basisbibliothek im Zentrum und die jeweiligen Programme in der Peripherie stehen. Das Projekt hat einen ständigen Wachstum an allgemeinen Quellcode und neuen Programmen zu verzeichnen. Die wichtigen und zentralen Programme, wie die Kernbibliotheken, übernahmen oft benötigte Funktionalitäten von Anwendungen und sind dadurch zu komplexen Gebilden geworden. Im Allgemeinen wird der Quellcode wiederverwendet, in dem dieser auf Kernbibliotheken ausgelagert wird, damit andere Programme darauf zugreifen können.
Dieses System hat nach außen hin einen freien Lesezugriff, aber einen beschränkten Schreibzugriff. Dieser Schreibzugriff für die gesamte Dateiablage außer der Homepage[20] über das Versionsmanagementsystem wird an die jeweilige Person vergeben. Die Identifizierung erfolgt dabei über die Emailadresse und Paßwörter, weswegen nur die Emailadressen der Personen, aber keine weitere persönliche Informationen bekannt sind. Man könnte also von den kontextlosen Daten im Mailinglistenverteiler und im Dateiablagesystem auf eine verbreitete Anonymität im Open Source-Projekt schließen. Doch gibt es eine implizite Regel, daß nur reale und authentische Personen an der Kommunikation teilnehmen sollen, d.h. jeder seinen realen Namen kontinuierlich verwendet. Es gibt außerdem Anstrengungen die persönliche Seite zu stärken, in dem persönliche Daten freiwillig bekanntgegeben werden, wie z.B. bei den Interviews von verschiedenen Projektbeteiligten auf der Homepage (s. Anhang A).

Wichtig ist noch der Compiler der zur Verfügung gestellt wird. Ein Compiler übersetzt den Quellcode in maschinenlesbaren Code, um das Programm laufen zu lassen. Damit können die

---

einander wieder beseitigen. Um solche Probleme, zufällige Löschungen oder Fehler ungeschehen zu machen, gibt es im System die Möglichkeit den alten Quellcode zurückzuholen.

[20] Die Homepage wird von einer kleinen Gruppe von Projektbeteiligten betreut.

Entwickler die Stabilität des Codes überprüfen. Dies machen sie oft, wenn sie Software umgeschrieben haben.

Es gibt noch weitere produktionsunterstützende Werkzeuge, wie ein Fehlerberichtsystem, ein Compiler sowie andere kleine Tools oder ein automatisches Übersetzungssystem. Das Fehlerberichtsystem unterstützt die Programmierer bei der Fehlersuche, indem es den Berichtenden über eine Eingabemaske, d.h. über ein standardisiertes Fehlerbeschreibungsverfahren, führt. Dabei kann die Schwere des Fehlers oder Wünsche eingegeben werden. Das Fehlerberichtsystem erstellt nach Eingabe eine Fehleremail, die normalerweise an einen Koordinator eines Softwareprojekts, an eine spezielle Fehlermailingliste und an eine Fehlerliste auf der Homepage geleitet wird. Die Schwere der Fehler werden von den Entwicklern überprüft und gegebenenfalls umgestuft, da es zwischen Nutzern und Entwicklern Interessenunterschiede in der Bewertung gibt.
Es gibt noch einige weitere kleine Tools, die die Programmierung erleichtern, wie z.B. ein spezieller Programmiersprachеneditor oder ein Tool, das den Quellcode auf Konsistenz und Fehleranfälligkeit überprüft. Die Modularisierung, aber auch die verschiedenen Werkzeuge erleichtern das Programmieren.
Bei der Dokumentation[21] und der Übersetzung werden Werkzeuge benutzt, die die zu übersetzenden Texte zusammenstellen, automatisch übersetzen, Veränderungen darstellen und die Übersetzung bzw. Dokumentation nach einem Standard Layouten. Dies erleichtert die Übersetzungstätigkeit, da durch die Automatisierung nur noch vereinzelte Wörter oder Abschnitte statt ganzer Texte übersetzt werden müssen. Dabei hat das Übersetzen der Programme, die veröffentlicht werden, eine höhere Priorität als die Dokumentation der Programme.
Es gibt zusätzlich community-erhaltende Funktionen, die sich um die Wartung der oben genannten Werkzeuge drehen. Diese Funktionen überwachen und pflegen die Werkzeuge, wie das Versionsmanagementsystem, die Mailinglisten, die Verteilung von Community-Emailadressen, Teile der Homepage, das das Fehlerberichtssystem, der Compiler, etc.

Innerhalb der Community konstituiert sich das Zentrum und die Peripherie bei der Software, den Projekten und den Mailinglisten. So gibt es zentrale Software, wie bestimmte Bibliotheken, und periphere Programme, wie Anwendungen (s. nächstes Kap. über IuK-Technologien). Mit diesen Softwareteilen sind jeweils bestimmte Subprojekte verbunden, weswegen manche Subprojekte wichtiger als andere sind. So stehen einige wichtige und große Programme, die in einer gemeinschaftlichen Arbeit entwickelt werden, im Zentrum. In der Peripherie dagegen ist eine Unmenge an kleineren Hilfsprogrammen, die z.T. von einzelnen Personen geschrieben werden. Die Mailinglisten unterteilen sich auch nach Zentrum und Peripherie. So gibt es zentrale Mailinglisten, auf die nur wenige Personen Schreibberechtigung haben, und auf denen wichtige Themen für das Projekt diskutiert werden. Wiederum gibt es eher unwichtige allgemeine Mailinglisten auf denen viele schreiben können und die Themen eher unwichtig sind. Außerdem gibt es spezielle Mailinglisten für spezifische Aufgaben.
Auch außerhalb der Open Source-Community ergibt sich eine Strukturierung in Zentrum und Peripherie. Es gibt neben den Hauptprogrammen des Großprojekts, die im projekteigenen

[21] Die Dokumentationen sind mit einer ausführlichen Hilfefunktion verbunden.

Versionsmanagementsystem erstellt werden, ca. 2000 kleine Programme, die außerhalb des Systems entwickelt werden, aber auf der Technologie der Open Source-Community basieren. Dabei wird größtenteils auf einige Bibliotheken des Open Source-Großprojekts zurückgegriffen. Viele dieser externen Anwendungen liegen auf zwei Servern, die von Unternehmen zur Verfügung gestellt wurden. Diese Programme werden zum größten Teil von Einzelpersonen ohne Team und ohne direkten Bezug zur Open Source-Community geschrieben.

4. Institutionelles Umfeld - Die äußere Struktur des Open Source-Projekts

4.1. Beziehung zu anderen Open-Source-Projekten

Die Community des untersuchten Projekts gehört zur sehr großen Open Source-Gemeinde, worunter auch die sehr große, wenn nicht sogar größte Linux-Community zählt. Die Beziehungen zu anderen großen Open Source-Projekten sind im Allgemeinen durch kooperatives Verhalten geprägt, da man sich einer großen, allgemeinen Open Source-Gemeinschaft zugehörig fühlt[22]. Jede Community, auch wenn diese ähnliche Produkte herstellen, arbeitet vor sich hin. Aus diesem Grund sind Konflikte zwischen Open Source-Projekten selten. Kontakte zwischen den Communities ergeben sich meist bei technischen Schnittstellen oder Themenüberschneidungen. So hat das untersuchte Open Source-Großprojekt einige Beziehungen zu anderen Open Source-Projekten, wie z.B. Debian, einem externen Open Source-Distributor, einem anderen Desktop für die Unix-Technologie oder der Unix/Linux-Community. Zusätzlich verwenden die einzelnen Communities bestehende Werkzeuge anderer Communities kostenlos und ohne Einschränkungen, womit sich die verschiedenen einzelnen Communities sich gegenseitig unterstützen (s. Kap. 2.4.).

Demgegenüber gibt es noch weitere Open Source-Projekte, die auf der Technologie des untersuchten Großprojekts aufsetzen. Diese Programme gehören zum Umfeld des Großprojekts, werden aber nicht in dessen Community entwickelt. Der freie Zugang zum Quellcode läßt einzelne Entwickler den Programmcode des Großprojekts für ihr Programm benutzen. Diese Programme, aber auch andere kleine Open Source-Projekte, die nicht auf diesem Großprojekt aufbauen, werden manchmal in das Großprojekt integriert, d.h. in die gemeinsame Dateiablage aufgenommen. Dies geschieht, wenn diese Programme eine sinnvolle Erweiterung des Desktops darstellen.

Es gab aber auch Konflikte zwischen der untersuchten Community und der restlichen Open Source-Community. Die Beziehungen haben sich mittlerweile normalisiert. Es gab am Anfang des Bestehens einen Konflikt um die Auswahl der Grundbibliothek bei dem Open Source-Projekt. Bei der Gründung wurde auf eine proprietäre Grafikbibliothek einer Firma zurückgegriffen, die für die nicht-kommerzielle Nutzung freigegeben war. Der Rest der Software sollte unter der General Public Licence (GPL) bzw. Lesser General Public Licence

[22] Doch gibt es hin und wieder kleine Absetzbewegungen zwischen den Gruppierungen, die sich in nicht ganz ernst gemeinten, kleinen Sticheleien auf Kongressen manifestieren.

(LGPL)[23] gestellt werden. Einige Personen aus der Open Source-Community, besonders die Free Software Foundation, monierten, daß die Lizenzen zwischen proprietärer Basisbibliothek und der GPL/LGPL miteinander kollidieren. Dieser Zustand war für viele der Entwickler der Open Source-Community ein Dorn im Auge, da das untersuchte Projekt abhängig von der Herstellerfirma der proprietären Basisbibliothek war. Dies führte zur Trennung in zwei verschiedene Großprojekte, aber mit dem gleichen Ziel ein Desktop für Unix-Derivate[24]. zu erstellen. Erst später und nach und nach ließ sich die Firma überzeugen, die proprietäre Basisbibliothek für die freiwillige Softwareentwicklung unter der GPL zu lizenzieren.
Später wurden weitere Sicherungsmaßnahmen für die Basisbibliothek, die es zugleich unter proprietärer und GPL-Lizenz gab[25], zwischen der Firma und dem eingetragenen Verein des Open Source-Projekts verabredet, falls die Firma in Konkurs gehen oder verkauft werden sollte. In dem Fall eines Konkurses oder eines Verkaufs soll der Wechsel der Lizenz der proprietären Basisbibliothek zu einer für kommerzielle und Open Source-Bedürfnisse entsprechenden Lizenz[26] erfolgen.
Heute haben sich die beiden Open Source-Projekte für einen Unix-Desktop in der großen Open Source-Community etabliert und ihre Beziehung zueinander normalisiert. Die Entwickler der beiden Projekte nutzen das Konkurrenzprojekt hin und wieder, um sich vom jeweils anderen Programm über Entwicklungsideen zu informieren. Zusätzlich gibt es Tendenzen auch Software gemeinsam zu entwickeln.
Aus diesem Konflikt läßt sich ersehen, daß die Softwarelizenz, die die Verbreitung regelt, als wichtig erachtet wird. Die Lizenz, speziell die GPL, bildet somit die wichtigste Grundlage für die freiwillige Entwicklung in der Open Source-Community.

4.2. Beziehung zwischen Open Source-Projekt und Unternehmen

Die Beziehung zwischen Open Source-Projekt und Unternehmen ist auf der einen Seite durch Kooperation geprägt. Auf der anderen Seite gibt es Unternehmen, die als Trittbrettfahrer oder unbedacht mit den Normen der großen Open Source-Community kollidieren, der das Open Source-Projekt angehört.

---

[23] Die General public licence (GPL) legt fest, daß veränderter Quellcode wieder der Community zugeführt wird. Die GPL definiert also das Nutzungs-, Veränderungs- und Weiterverbreitungsrecht. Das Nutzungsrecht legt fest, daß diese Software kostenfrei zu haben ist. Das Veränderungs- und Weiterverbreitungsrecht bestimmt, daß veränderter Quellcode und davon abgeleitete Werke wieder der GPL unterstehen müssen. Damit soll eine proprietäre Aneignung des freiwillig geschriebenen Quellcodes mit nachfolgenden Verkauf ausgeschlossen werden. Außerdem steht in der GPL, daß im Programm ein copyright-Vermerk über den Autoren stehen soll. Die GPL ist in der Open Source-Community die am weitesten verbreitete Lizenz. Die GPL erlaubt die Nutzung von Softwareteilen, die unter der GPL stehen, in proprietärer Software nicht, die LGPL dagegen erlaubt dies.

[24] Unix-Derivate sind Abkömmlinge des Ur-Unix. Wichtigste Vertreter sind Linux, BSD-Unix, etc.

[25] Die Firma ist der Urheber der Software und kann diese Software unter eine proprietäre und eine GPL-Lizenz setzen, da sich beide Lizenzen ausschließen. Veränderte Software unter der GPL-Lizenz muß wieder unter diese Lizenz fallen und kann somit nicht proprietär und verkauft werden.

[26] Diese Lizenz ist die BSD-Lizenz, die den späteren Gebrauch von freiwillig erstellter Software außer einem Haftungsausschluß und der Bedingung, daß alle Entwickler namentlich erwähnt werden, freistellt. Damit kann die Software unter dieser Lizenz weiterentwickelt und verkauft werden.

4.2.1. Kooperative Beziehungen zwischen Open Source-Projekt und Unternehmen

Die Kooperation einiger open source-naher Unternehmen mit dem Open Source-Projekt basiert auf einem gegenseitigen Geben und Nehmen. Einige Unternehmen sind Distributoren, die eine hohe Abhängigkeit von dem Großprojekt haben, da sie ein modernes Betriebssystem, wie z.B. Linux, nicht ohne das Desktop dieser Community verkaufen können.
Die Abhängigkeit der Distributeure von dem Produkt fangen diese über angestellte Programmierer auf, die an dem Open Source-Produkt arbeiten. Einerseits arbeiten die angestellten Programmierer am Open Source-Produkt und somit zugleich am Unternehmensziel mit, da ein verbessertes Open Source-Produkt den Verkauf für das Unternehmen erhöht. Außerdem bringen die angestellten Programmierer technisches Wissen über das Open Source-Produkt mit, was das Unternehmen für seine Produktion benötigt. Andererseits stellen die angestellten Programmierer ein wichtiges Verbindungsglied zur Open Source-Community über ihre Kenntnisse der Projektbeteiligten dar.
Die Übereinstimmung von Unternehmens- und Großprojektziel ermöglicht es, die Weisungsbefugnis im Arbeitsvertrag für die Arbeit an dem Projekt auszusetzen. Die Weisungsbefugnis kann einerseits für die Vollzeitarbeit oder für einen bestimmten Teil der Arbeitszeit im Arbeitsvertrag explizit ausgeschlossen werden. Andererseits kann die Weisungsbefugnis zwar im Arbeitsvertrag enthalten sein, aber für die Arbeit nicht praktiziert werden. Die Abwesenheit von Weisung führt dazu, daß die angestellten Entwickler selbstbestimmt für das Projekt arbeiten. Dieses selbstbestimmte Arbeiten wird auch deswegen gewährt, da diese Arbeitsweise im Projekt vorherrscht und dieses vorantreibt. Die Abhängigkeit geht sogar so weit, daß die Firmen nach den Normen der Projektcommunity handeln müssen, da sie sonst Schwierigkeiten mit dieser bekommen (s. nächstes Kapitel). Einige wenige Firmen versuchten wegen dieser Abhängigkeit Einfluß auf das Open Source-Projekt zu bekommen. Bisher aber haben sich die Distributoren oder andere Firmen von Einflußnahmen zurückgehalten[27].

Es gibt noch andere open-source-nahe Unternehmen, die von dem Großprojekt abhängig sind. Diese Abhängigkeit kommt durch das Angebot von open-source-nahe Dienstleistungen, wie Anpassungen von Linux/Unix und Desktop an Unternehmens- bzw. Verwaltungsverhältnisse oder die Programmierung von speziellen Funktionen im Desktop, die für die Allgemeinheit von staatlichen Stellen bezahlt werden. Diese Firmen wurden von Personen aus der Community gegründet und haben z.T. andere Projektbeteiligte aus der Community angestellt. Diese Überschneidung zwischen Projektbeteiligten und Arbeit zeigt sich darin, daß die Programmierer einerseits an open-source-nahen Aufgaben arbeiten. Andererseits arbeiten sie oft nebenher, in ihrer Freizeit für das Open-Source-Großprojekt an z.T. kleinen Projekten mit.
Das Unternehmen mit der großprojektrelevanten Basisbibliothek hat dabei eine Sonderstellung. Einerseits hat es keine großen Abhängigkeiten zum Großprojekt, profitiert aber von ihr. So profitiert das Unternehmen von dem Wissen und der kurzen Einarbeitungszeit der Projektbeteiligten. Viele Angestellte kommen aus dem Großprojekt und arbeiten an dieser Basisbib-

[27] Ein Versuch einer Einflußnahme auf die Community wurde schon berichtet, da eine Firma Einfluß auf den eingetragenen Verein des Projekts (s. u.) nehmen wollte, da sie annahm, daß dort die wichtigen Entscheidungen erfolgen würden.

liothek, wobei einige noch an dem Großprojekt beteiligt sind. Außerdem profitiert das Unternehmen durch Fehlermeldungen und Wünsche aus dem Großprojekt, die sie für ihr kommerzielles Produkt verwenden können. Zusätzlich können sich zukünftige Käufer das Softwarepaket vorher ansehen, bevor sie dieses kaufen.

Die Firmen, die von dem Großprojekt abhängig sind, sind zugleich die Sponsoren des Open Source-Projekts, die dem Großprojekt öfters Ressourcen spenden, entweder Geld oder Sachspenden, ohne die das Projekt in dieser Größenordnung nicht mehr lebensfähig wäre. Gerade diese Firmen unterstützen das Großprojekt durch die Bereitstellung von wichtigen community-unterstützenden Systemen, wie z.B. Kommunikationsbandbreite oder Speicherplatz für Homepages oder Dateiablagesysteme. Sie unterstützen das Projekt auch bei den Präsentationen auf den Linux-Konferenzen bzw. -Messen durch Hard- und Software. Diese Unternehmen spenden aber auch Geld an den gemeinnützigen Verein des Open Source-Projekts, in dem fast nur Entwickler, z.T. die wichtigsten, Mitglieder sind.
Der Verein wurde einerseits gegründet, um die (Haupt-)Mitglieder für Rechts- und Finanzverpflichtungen abzusichern und um Spenden verbuchen zu können. Sein Auftrag ist die Softwareentwicklung des Großprojekts zu unterstützen, aber keine Entscheidungen über die Softwareentwicklung zu treffen. Die Geldspenden sollen dann für die Entwicklung ausgegeben werden, wie die Hardware für Entwickler von wichtiger Software, der Unterstützung von Konferenzauftritten oder für Treffen von Entwicklern. Zweitens soll dieser Verein die Grundlagenbibliothek des Projekts für die weitere Verwendung absichern[28] (s. Kap. 1.6.1.). Neuere Überlegungen gehen dahin, den Verein aufzuwerten und als Schiedsinstrument für großprojektinterne Konflikte zu institutionalisieren.
Die Sachspenden dagegen gehen meist direkt an die Projektbeteiligten. Dabei ist zu unterscheiden zwischen Sachspenden von den Firmen, die von Beteiligten erbeten werden, da z.B. ihre Hardware versagt und diese nicht mehr weiter arbeiten können, und freiwilligen Leistungen, wie der Verteilung von Firmen T-Shirts.
Für die Spende werden die Sponsoren je nach Anlaß und Häufigkeit auf der Homepage, den Flugblättern und an den Messeständen erwähnt. Die Sponsoren haben aber keine Einflußmöglichkeiten auf die Entwicklung; die Spende ist eher als eine Motivationsunterstützung zu sehen. Das Gleiche gilt auch für Ressourcen von Privatpersonen.

4.2.2. Konfligierende Beziehungen zwischen Open Source-Projekt und Unternehmen

Es gibt Unternehmen, die nach den Normen der Community handeln, wie oben gezeigt wurde. Es gibt aber auch Unternehmen, die nicht so handeln und versuchen das Produkt dieser Open Source-Community für ihre Zwecke auszunutzen. Hier zeigt sich, daß die Lizenz, in diesem Fall die GPL (s. Kap. 1.6.1.), der wichtigste Eckpfeiler der Communitynormen ist. Dabei geht es um die Frage der Verfügungsrechte[29] über das Produkt des Großprojekts und wie diese Rechte eingefordert werden können. Die GPL-Lizenz ist dabei die rechtliche Vor-

[28] Es gibt einen Vertrag zwischen dem Verein und dem Eigentümer der Basisbibliothek, daß im Falle eines Aufkaufs oder Insolvenz der Firma der Verein Rechte an der Bibliothek erhält. Dabei wird die Lizenz der Basisbibliothek auf eine Open Source-Lizenz umgestellt, die mit einer kommerziellen Verwertung kompatibel ist.

[29] Verfügungsrechte sind Rechte der Nutzung, Verleihung und Verkaufs.

schrift, die vorgibt, daß der veränderte Quellcode wieder unter die GPL-Lizenz zu stellen ist. Der Verkauf als proprietäre Software wird ausgeschlossen, möglich ist aber der Verkauf von Hardware mit dieser Software als Dreingabe bzw. Geschenk. Die Open source-Software ist ein Geschenk an die Allgemeinheit, in gewissem Sinne ein öffentliches Gut.
Der größte Teil der Software des Projekts ist unter der GPL oder der Lesser General Public Licence (LGPL) lizenziert, wobei die Anwendungsprogramme normalerweise unter der GPL, die Bibliotheken mit einzelnen Funktionen eher unter der LGPL stehen. Die LGPL ermöglicht es, daß proprietäre Software auf Funktionen in diesen Bibliotheken zugreifen darf., was bei der GPL nicht erlaubt ist[30]. Mit der LGPL soll es Unternehmen vereinfacht werden, für das Großprojekt Programme zu schreiben.
Wie aber mit der Lizenz zu verfahren ist, wird aber erst durch die Normen der allgemeinen Open Source-Community und der Projektcommunity festgelegt.
Dabei steht die Community des Open Source-Projekts in einem allgemeineren Dilemma, das nicht aufzulösen, d.h. systemimmanent, ist. Einerseits gibt es den offenen Quellcode, der frei zugänglich und ein zentraler Grundsatz der Community ist. Damit werden neue Mitprogrammierer aber auch Unternehmen angezogen, die das Produkt des Open Source-Projekts entweder verwenden oder dafür Programmieren. Auf der anderen Seite steht das mögliche Trittbrettfahren bzw. Ausnutzen durch Verkauf in proprietärer, compilierter Software, was z.T. schwer zu beweisen bzw. zu sanktionieren ist.
Gerade die allgemeine Open Source-Community und die Community des Großprojekts reagieren auf die verschiedenen Normenbrüche bezüglich der GPL und versuchen so die verschiedenen Arten des Trittbrettfahrens zu unterbinden[31]. Im folgenden werden verschiedene Normenbrüche dargelegt, die die verschiedenen Normen bezüglich der Lizenz dargestellt.

Ein Normenbruch liegt z.B. vor, wenn ein Open Source-Programm, das unter der GPL steht, proprietär verkauft wird oder es scheint als würde die Software proprietär verkauft. Darauf reagiert die Open Source-Bewegung, besonders die Hüterin der GPL, die Free Software Foundation, mit Sanktionen. So hat die Firma Softwaretechnik[32] auf deren Homepage einen Preisvergleich zwischen Microsoft Office-Programmen und dem Programmpaket des Großprojekts angestellt. Dabei erschien der Open Source-Community als würde die Firma das Open Sour-

[30] Die LGPL wurde für Bibliotheken entwickelt, da in diesen verschiedene oft benötigte Funktionen gesammelt werden. Der Unterschied zwischen GPL und LGPL ist, daß bei der LGPL der Quellcode von den Open source-Bibliotheksfunktionen in proprietären Programmen verwendet werden darf.

[31] Dabei besteht aber das Problem, daß Lizenzübertretungen manchmal nicht bewiesen werden können, da der Quellcode bei proprietärer bzw. kommerzieller Software nur in der compilierten Fassung vorliegt. So ist ein Beweis sehr schwierig. Trotzdem gibt es oft starke graphische Ähnlichkeiten zwischen Open Source-Software und proprietärer/kommerzieller Software. Bei einigen Sharewareprogrammen (Sharewareprogramme sind kleine proprietäre Programme, die gegen ein kleines Entgelt benutzt werden können.) unter Windows herrscht der begründete Verdacht, daß diese Open Source-Code aus dem Projekt enthalten und diesen verkaufen.

[32] Name der Firma geändert.

ce-Programmpaket verkaufen wollen. Diese Firma wurde mit Emails aus der Community überschüttet, so daß der Streitpunkt schnell von der Firma bereinigt wurde[33].

Die Open Source-Community, in diesem Fall Personen aus dem Großprojekt, reagiert auch auf das Fehlen von copyright-Vermerken, von den Urhebern, des Lizenztexts und dem Namen bzw. Emblems des Projekts. In diesem Fall hat eine Firma Hardware mit dem Desktopprodukt als Dreingabe verkauft, wobei diese Angaben fehlten. Das Großprojekt möchte die Reputation bzw. Anerkennung aus ihrer Umwelt für die Früchte ihrer Arbeit in Form von copyright-Texten. Außerdem impliziert das Fehlen von der Urheber, des Lizenztexts und dem Namen bzw. Emblems des Projekts, daß die Software der Firma gehört. Die kostenlose Nutzung der Programme unter dem Namen des Großprojekts ist aber erlaubt[34].

Nach der GPL ist die Veränderung des Quellcodes erlaubt, solange der Quellcode wieder zur Verfügung gestellt, d.h. wieder an die Community zurückgegeben wird. Aber ein weiteres wichtiges Thema bei Normenbrüchen war die Zeit, die gebraucht wird, um einen produzierten Quellcode wieder der Community zur Verfügung zu stellen. Die Community besteht darauf, daß der Quellcode sofort nach der Programmierung weitergegeben wird. Eine große Firma hatte sich dagegen einen Zeitraum von ein paar Monaten ausbedungen, um den technologischen Vorteil einer Veränderung des Produkts des Großprojekts nutzen zu können. Die Community[35] ist dagegen Sturm gelaufen, da damit ein Verwässern des Allgemeingutprinzips hin zum proprietären Gedanken eingeschlossen wäre. Außerdem hätte der veränderte Quellcode nach dem Zeitraum durch die ständige Weiterentwicklung nicht mehr zu dieser Version gepaßt. Aus diesem Grund und wegen der vielen Veränderungen wurde eine Gabelung des Projekts befürchtet. Die Gabelung von Software in zwei getrennte Projekte aus einem Originalcode wird nicht gerne gesehen, obwohl sie im Prinzip möglich und erlaubt sind.
Diese Abneigung gegen Gabelungen ist auf verschiedene Gründe zurück zu führen. Erstens existiert die Norm, daß es für eine Funktion nur ein bearbeitendes Projekt geben soll. Zweitens würde ein weiteres ähnliches Projekt die Reputation verwässern und drittens zu einer ungewollten Konkurrenz in einem auf ein Allgemeingut konzentrierte, solidarische Community führen. Viertens möchten die Beteiligten des Großprojekts nicht für den Quellcode eines Projektzweigs, für das sie nicht zuständig sind oder arbeiten, verantwortlich gemacht werden und Unterstützung leisten.
Die schlechte Publicity manifestierte sich in Emails, die an die Firma geschickt wurden, und in großen langen Diskussionen auf den Internet-Newsseiten slashdot bzw. zdnet. Es erfolgten aber keine Softwareattacken, wie z.B. Mailbombs.

---

[33] Dies zeigt, daß das Softwareprodukt durch die GPL als ein Gemeingut betrachtet wird, das allen zur Nutzung zur Verfügung gestellt wird. Der Verkauf eines Produkts dagegen impliziert den Besitz dieses Produkts. Es soll also niemand aus einem Allgemeingut seinen individuellen Profit ziehen.

[34] Eine Namensänderung ist nur erlaubt, wenn es eine Gabelung der Software unter der GPL-Lizenz in zwei Entwicklungsrichtungen gibt.

[35] Die Community des Großprojekts war in dieser Frage gespalten, da auf der einen Seite eine große Gruppierung diese Meinung vertrat. Auf der anderen Seite hatten einige wichtige Personen aus der Community guten Kontakt zur Firma und waren mit dessen Vorgehen einverstanden.

Die Community des Großprojekts reagiert folglich auf solche Normbrüche nur mittels ihrer „Marktmacht“. Sie ist sich seiner Größe und Bekanntheit bewußt, so daß manchmal diese Popularität gegen Normenbrecher benutzt werden. Es wird versucht, Firmen, die solche Schwierigkeiten machen, über Imageschäden zum Einlenken zu bewegen. Diese Imageschäden können dabei auch direkte Folgen auf die Verkaufszahlen haben, da viele Personen aus der Community wichtige Positionen in Unternehmen oder Rechenzentren und somit Budgetverantwortung haben. Bisher ging man mit dieser „Macht“ sehr sparsam und zögerlich um. Dabei möchte das Projekt die Firmen, die die Normen um die Lizenz ignorieren, erst einmal sachlich überzeugen und gewinnen. Dahinter steht die Grundeinstellung, daß Konfrontation der Sache, d.h. der Programmierung im Open Source, nicht dient und Unternehmen als Partner gewonnen werden sollen[36]. Sie nutzen diese Öffentlichkeit nur, wenn gravierende Verstöße gegen die Interessen, d.h. gegen die Normen um die Lizenz, des Großprojekts vorhanden sind. Solche Normenbrüche treten aber selten auf.
Die Nutzung der „Marktmacht“ erfolgt je nach Situation, d.h. die Anwendung des Protests bei Lizenzverstößen unterliegt einer Kosten-Nutzen-Abwägung, wobei die Verhältnismäßigkeit als wichtig erachtet wird. Die Größe des Unternehmens spielt als ein Kriterium der Situation eine Rolle. Auf der einen Seite könnten große Unternehmen zwar einen solchen Sturm der Entrüstung aussitzen, doch ist der Schaden für das Image der Firma nicht zu quantifizieren, kann also sehr groß sein. Damit sind große Firmen stark angreifbar. Auf der anderen Seite lenken kleine Firmen oft bei dem Androhen von schlechter Publicity schnell ein, da diese identifizierbar und von ihrem Leumund abhängig sind. Die Grenzen des Einflusses der Popularität liegt bei Sharewareprogrammen. Bei kleinen Sharewareprogrammen lohnt sich der Protest des Großprojekts nicht, da diese oft aus einem Eigentümer bestehen, der im Gegensatz zu Unternehmen keine Reputation zu verlieren hat. Eine Privatperson kann sich auch schneller aus der Affäre ziehen, da sie nicht vom Verkauf des Programms abhängig ist. Solche Mitläufer/Trittbrettfahrer müssen wohl oder übel toleriert werden.
Ein wichtiges Kriterium für die Situation ist das Größe des veränderten Programms. Bei kleinen Programmen setzt die Community des Großprojekts seine Popularität und Marktmacht nicht unbedingt ein. In diesem Fall hatte ein großes Unternehmen einen kleinen proprietären Windows-Druckertreiber herausgebracht, der eine genaue Kopie von der Software des Projekts mit allen Fehlern war. Bei Firmen dagegen, die große Teile von Programmen übernehmen und verändern, reagiert die allgemeine Open Source-Community sehr allergisch. Dabei ist es einerlei, ob diese Firmen groß oder klein sind.

Ein weiteres Ausnutzen der Community des Großprojekts durch die Unternehmen wird auf einem anderen Wege versucht. Die Open Source-Community soll als kostenfreie Entwickler, d.h. dem Anpassen des Projektproduktes für ein Firmenprodukt, ausgenutzt werden. Dies entspricht aber nicht der Kultur des Gebens und Nehmens in der Community, da die Firma nichts zurückgibt. Für solche Sonderwünsche wird eher von der Community erwartet, daß für dieses Problem Entwickler von der Firma bezahlt werden oder andere Gegenleistungen erfolgen.

[36] Man kann diese Haltung auch so deuten: Die kooperative Haltung im Großprojekt wird auch auf äußere Beziehungen mit Unternehmen übertragen.

Wie aus diesen Konflikten ersichtlich, ist die GPL-Lizenz ein wichtiger Rahmen und Grundpfeiler der Open Source-Community, da durch die Lizenz und die Lizenz umgebenden Normen die Nutzung des Quellcodes geregelt wird. Die GPL und die Normen schützen also das gemeinsame freiwillig programmierte Produkt vor dem Ausnutzen von Trittbrettfahrern. Der Bestand der GPL wurde aber bisher noch nie von einem Gericht geprüft, weder von dem Großprojekt noch von der Free Software Foundation, die diese entwickelt hat. Die meisten Lizenzprobleme mit Firmen werden durch das indirekte, unterschwellige Androhen von schlechter Publicity und der geballten Marktmacht der gesamten, internationalen Open Source-Community beigelegt.

4.3. Beziehung zwischen Entwicklern und Endnutzern

Die Beziehung zwischen Entwicklern und Endnutzern unterliegt einem Spannungsverhältnis. Das Spannungsverhältnis wird durch verschiedene Faktoren bestimmt. Der Hintergrund ist, daß die Entwickler einen Linux-Desktop produzieren, der von Endnutzern ohne große Programmier-Vorkenntnisse genutzt wird.
Die Entwickler sind dabei auf der einen Seite Ansprechpartner für Fehler in dem Produkt. Fehlerberichte und Funktionswünsche werden normalerweise begrüßt und es wird mit dem Endnutzer freundlich umgegangen, da die Entwickler von den externen Informationen abhängig sind. Sie wissen meistens nicht, wie eine bedienbare Funktion gestaltet wird. Dabei gilt das Transparenzprinzip und das Prinzip der kooperativen Kommunikation. Damit möchten die Entwickler neue Personen als Mitarbeiter an dem Großprojekt werben. Außerdem haben die Entwickler das Selbstverständnis, daß sie für die Endnutzer programmieren, d.h. daß die einfache Bedienbarkeit im Zentrum steht[37].
Auf der anderen Seite reagieren die Entwickler abwehrend, wenn Endnutzer Wünsche und Entwicklungsvorschläge vorschlagen oder einfordern. Dies zeigt, daß beide Gruppierungen unterschiedliche Interessen verfolgen[38].
Die Entwickler lassen sich aber nicht gerne ins Programmieren hineinreden, da in dem Open Source-Projekt die Programmierer selbstbestimmt und freiwillig Software entwickeln. Außerdem sind die Entwickler stolz auf ihre Arbeit. Die soziale Anerkennung im Großprojekt hängt mit dem Arbeitseinsatz und dem Wissen über die Software zusammen, wobei nur Personen, die sich am Projekt beteiligen, ernst genommen werden (s. Kap. 2.2.1.). Von diesen Personen werden eher Vorschläge akzeptiert, weswegen den externen Nutzern jegliche Kompetenzen in der Softwareentwicklung abgesprochen wird. Dieses Verhalten kann auch als eine Abgrenzung von den Nutzern verstanden werden.

---

[37] s. Ergebnisse der offenen Frage (Fragenr. 20) der quantitativen Befragung aus Anhang C

[38] Beispiele hierfür sind Beiträge von Nutzern auf bekannten Newsseiten der Open Source-Community, die nicht mit der Praxis der Projektbeteiligten übereinstimmen oder ein negatives Bild von dem Produkt der Community zeichnen. Diese Kritiken über die Ausführung des Desktops werden in dem Großprojekt z.T. emotional diskutiert. Als Reaktion darauf wurde von dem Großprojekt eine eigene Internetseite zu den Mythen über das Projekt und ihre Richtigstellung geschaltet. Dabei wird z.T. versucht, ein Open Source-Projekt gegen andere auszuspielen. Dies nahm aber im letzten Jahr ab, da eingeforderte Funktionen durch die rasante Entwicklung implementiert wurden.

Aus diesem Grund können die Antworten der Entwickler öfters recht harsch oder herablassend ausfallen und eine Aufforderung zur Mitarbeit des Kritisierenden folgen. Dabei hängt die Reaktion der Entwickler meistens auch von deren Charakter ab. Dies steht aber im Widerspruch zur Norm der kooperativen Kommunikation.
Ein solches Spannungsfeld gibt es auch bei Übersetzern/Dokumentierern. Doch stehen diese nicht im direkten Rampenlicht wie die Entwickler.

5. Funktionsweisen der Ausprägung des elektronischen Teilarbeitsmarkts - Die Funktionsweise eines Open Source-Projekts als elektronisches Arbeitsnetz

In diesem Teil wird auf die vier Arbeitsmarktfunktionen, Allokation, Leistungserstellung, Kontrolle und Gratifikation/Motivation, eingegangen. Die Funktionsweise der vier Arbeitsmarktfunktionen wird beschrieben, wie diese sich im Erhebungszeitraum darstellte. Die Allokation betrachtet die Verteilung von Gütern und Faktoren auf Personen oder Produktionsprozesse. Auf dem Arbeitsmarkt bestehen Organisationen, in denen die Arbeitskraftransformation abläuft, weswegen in eine organisationsexterne/äußere und eine organisationsinterne/innere Verteilung der Arbeitsleistungen aufgeteilt werden kann (s. Kap. 1.4.). Die äußere Verteilung wäre mit dem Ein- und Austritt in bzw. aus einer Organisation, hier dem Arbeitsnetz, vergleichbar. Die innere Arbeitsverteilung wird in einer Organisation über Weisung durch den Vorgesetzten vollzogen, was in einem Open Source-Projekt nicht vorhanden ist. In einem Open Source-Projekt ist die freiwillige Wahl eines Arbeitspakets nicht von der Leistungserstellung zu unterscheiden. Bei der Kontrolle wird besonders auf die Kontrolle der Qualität eingegangen, die in diesem Fall am wichtigsten ist. Eine Kontrolle, ob ein Beteiligter überhaupt arbeitet, ist wegen der Freiwilligkeit der Beteiligung nicht möglich. Die Gratifikationsfunktion wird als Motivationsfunktion umgedeutet, da in diesem Fall nicht-monetäre Gratifikationen eine große Rolle spielen.
Bei der Betrachtung der Funktionsweise von Arbeitsnetzen steht die große und zentrale Gruppe der Softwareentwickler im Vordergrund. Die Gruppe der Dokumentierer und Übersetzer wird als zweitgrößte Gruppe auch betrachtet. Die Gruppe der Journalisten und Graphiker/Soundspezialisten wird wegen ihrer geringen Größe vernachlässigt.

5.1. Eintritt und Austritt in das oder aus dem Projekt

In diesem Kapitel wird auf den Ein- und Austritt in das Großprojekt eingegangen. Im ersten Unterkapitel geht es um die Aufnahme in die Gruppe des Open Source-Projekts. Im zweiten um die Aufnahme in den inneren Kreis und im letzten um die Austritte aus der Gruppe des Open Source-Projekts.

5.1.1. Eintritt in das Großprojekt

Der Eintritt läßt sich in eine Anwerbungs-, eine Kontakt- und eine Eintrittsphase unterteilen. In der Anwerbungsphase werben Personen aus dem Open Source-Projekt Außenstehende an. So erfolgt die Anwerbung einerseits über direkte oder indirekte Anwerbungskanäle. Ein direkter Kanal besteht, wenn Projektbeteiligte aktiv über spezielle Medien Personen werben und neue Eintritte erfolgen. Ein wichtiger Kanal sind Artikel über das Projekt in bekannten Com-

puterzeitschriften, die entweder werbenden Charakter haben, wie in Anhang A beschrieben wurde, oder das Produkt mit seiner Technik darstellen. Weitere Werbeeffekte gehen von Interviews mit Entwicklern über die Technik oder die Darstellungen des Softwareprodukts in Online-Newsforen aus. Die Anwerbung über Freunde kommt auch vor. Dabei spielt das Sendungsbewußtsein des Open Source-Projekts als soziale Bewegung eine große Rolle. Z.B. wird auf Messen mit interessierten Außenstehenden sehr freundlich umgegangen und gefachsimpelt. Damit soll ein sehr positives Bild des Projekts präsentiert werden, um diese anzuwerben. Die aktive, ständige Anwerbung wird aus zwei Gründen benötigt. Erstens spielt die hohe Fluktuation von Mitarbeitern und zweitens die ständige Erweiterung der Software, die zum Projekt gehört, eine große Rolle.
Die meiste Werbung wird aber indirekt über die Linux-Distributionen mit dem Desktop-Programm gemacht. Ein sehr großer Teil der neuen Teilnehmer war vorher Nutzer des Desktops und entschied sich irgendwann für eine Beteiligung.

In der Kontaktphase müssen die Außenstehenden einen ersten Schritt aus eigenem Antrieb machen, um mit den Projektbeteiligten in Beziehung zu treten. Dies stellt eine erste Auswahlstufe dar. Der erste Kontakt findet entweder über Email oder über persönliche Treffen auf Linux-Konferenzen statt, wobei die Treffen nur einen geringfügigen Anteil darstellen.
Die Angeworbenen sind dabei nicht unbedingt Neulinge in der Open Source-Gemeinde. Einige haben schon in anderen Open Source-Projekten mitgearbeitet oder arbeiten mit. 41 von 55 Personen gaben an, schon einmal oder noch an einem anderen Open Source-Projekt mitgearbeitet zu haben (s. Anhang C). Doch laut Aussage eines Interviewten, arbeitet kaum einer parallel zu dem Großprojekt in anderen Open Source-Projekten, da die Arbeit an dem Großprojekt viel Zeit und Aufmerksamkeit benötigt.
Die neuen Teilnehmer, die sich aktiv beteiligen möchten und Kontakt suchen, werden normalerweise freundlich empfangen und auf mögliche Arbeit hingewiesen. Dabei werden einige Neueinsteiger durch die Fehlerberichte geworben, in dem den Neueinsteigern erklärt wird, daß dieser Fehler ohne ihn nicht beseitigt wird.
Die Transparenz der Kommunikation im Sinne der Zurechenbarkeit zu einem Kommunizierenden, der Entscheidungen, der Quellcodeveränderungen und der kooperative Umgangston spielen in dieser Phase eine große Rolle bei der weiteren Anwerbung. Unfreundliches Verhalten gegenüber Außenstehenden ohne viel Vorwissen ist die Ausnahme und wird nicht gerne gesehen. Das bedeutet, daß Authentizität und Kontinuität im Verhalten wichtig sind, um Vertrauen zur Community aufzubauen.
Die Angeworbenen mit Programmierkenntnissen werden aufgefordert erst eine Eintrittshürde zu überwinden, weswegen nur wenige[39], die sich beteiligen möchten, bleiben. Die restlichen Außenstehenden, die diese Kenntnisse nicht haben, werden an die Übersetzer/Dokumentierer weitergereicht.

Die Eintrittsphase stellt einen zweiten Schritt der Angeworbenen dar, den diese aus eigenem Antrieb machen müssen, um aufgenommen zu werden. Somit ist dies die zweite Auswahlstufe. Die neuen Teilnehmer, die sich für das Programmieren interessieren, müssen eine Ein-

[39] Eine Schätzung aus der Community sagt, daß nur jeder zehnte, der sich beteiligen möchte, länger mitarbeitet.

trittshürde überwinden, um ein vollwertiges Mitglied der Entwicklergemeinde zu werden. Die Eintrittshürde geht mit einer technischen Hürde einher, d.h. zur autonomen Mitarbeit ist eine Schreibberechtigung für das Dateiablagesystem nötig. Der Neueinsteiger ist dann in das Großprojekt aufgenommen worden, wenn er diese Schreibberechtigung zuerkannt bekommen hat.
Dieses Vorgehen versichert der Community einerseits, daß die eintretenden Personen Interesse und Motivation für die weitere Mitarbeit haben. Damit sollen Ressourcen, wie Aufmerksamkeit, nur in interessante Personen investiert werden. Der Neue muß erst zwei bis drei Patches[40] für ein Programm an den zuständigen Projektbeteiligten mit Schreibberechtigung im Dateiablagesystem schicken. Damit signalisiert der Neue das Interesse an einer längeren Mitarbeit, d.h. er muß sich das Vertrauen erst bei der Community erarbeiten. Auf der anderen Seite geht die Community dann davon aus, daß die Neueinsteiger sich an der Community kooperativ beteiligen möchten und deswegen nicht opportunistisch handeln werden. So geht man davon aus, daß die Patches keinen Code enthalten, der der Community Schaden, wie Viren o.ä., zufügen und sich in einer Reputationsminderung auswirken könnte. Die Patches müssen zusätzlich einem weiteren Kriterium genügen, d.h. der neue Patch soll keine zusätzliche Arbeit für den Projektbeteiligten verursachen, der diesen Patch in das Versionsmanagementsystem einpflegt. Zusätzlich garantiert die Hürde, daß die Angeworbenen Kenntnisse im Programmieren und ein Minimum an Qualität bei der Programmierung haben[41]. Der eingesessene Projektbeteiligte verschafft dem Neuen nach Einhaltung der Regeln eine Schreibberechtigung für Versionsmanagementsystem. Der Zugang wird schneller vergeben, wenn mehr Arbeitsaufwand in eine spezielle Software, wie in ein plug-in, d.h. ein spezielles Programmerweiterungsmodul mit neuen Funktionalitäten, für das Projekt investiert wird. Nach dem Überspringen dieser Hürde wird dem Neueinsteiger vertraut, daß dieser mit dem vergebenen Schreibzugriff auf die gesamte Software der Dateiablage, außer der Homepage, verantwortlich umgeht. Erst nachdem sie diese Stufe durchlaufen haben und sich weitestgehend orientiert haben, werden sie als Beteiligte wahrgenommen.

Bei den Übersetzern gibt es keine direkte Einstiegshürde. Die Schreibberechtigung für das Dateimanagementsystem wird nicht an die Neueinsteiger vergeben. Hier bildet die technische Schranke keine soziale Schranke. Die Aufnahmewilligen übernehmen vielmehr eine Aufgabe, die sie zugeteilt bekommen. Eine Übersetzungsgruppe einer Sprache aus Dokumentierern und Übersetzern besteht aus einem oder mehreren Koordinatoren, die die Aufgaben an die Beteiligten verteilen, und einigen weiteren Mitarbeitern. Diese Aufgabe haben die Aufnahmewilligen bis zu einem bestimmten Zeitpunkt zu erledigen, d.h. sie müssen sich bewähren, bevor sie in den Kreis aufgenommen werden. Nur die Koordinatoren einer Übersetzungsgruppe einer Sprache oder einer Teilgruppe haben eine Schreibberechtigung für das Dateimanagement, die

[40] Patches sind "Flicken" für ein Programm, die aus Quellcode, d.h. aus den Differenzen zwischen der Vorgängerversion und der neuen Version bestehen. Patches werden in der Kommunikation zwischen Entwicklern und z.T. auch zwischen Entwicklern und Anwendern benutzt.

[41] Die Hürde muß zwei entgegengesetzte Kriterien austarieren, um optimal zu wirken. Die Hürde darf nicht zu hoch sein, da viele motivierte Personen abgeschreckt werden. Sie darf aber nicht zu niedrig sein, um motivierte Personen von Trittbrettfahrern zu unterscheiden.

restlichen Beteiligten nicht. So werden die neuen Übersetzungen von den Mitarbeitern üblicherweise an diese Koordinatoren gesendet, die diese dann in das Dateisystem einpflegen. Die Dokumentierer und Übersetzer ohne Schreibberechtigung gehören aber zu dem Open source-Projekt dazu, da sie wichtige Dienstleistungen für das Produkt bereitstellen. Zudem gibt es auch eine schwache Wanderungsbewegung von den Dokumentierern und Übersetzern zu den Entwicklern. Außerdem sind einige Entwickler gleichzeitig Dokumentierer und Übersetzer.

Ein neuer Mitarbeiter ist vollkommen in die Community aufgenommen, wenn er eine projektspezifische Emailadresse bekommt. Dies gilt auch für Mitarbeiter ohne Zugang zur Dateiablage soweit diese wichtige Aufgaben für die Community übernehmen. Die projektspezifischen Emailadressen werden von einem Koordinator auf Anfrage der Neueinsteiger bei Verweis auf geleistete Arbeit, verteilt.

Eine ähnliche soziale Hürde gibt es bei jedem Unterprojekt des Großprojekts. Beim Eintritt ins Großprojekt fallen soziale und technische Hürden zusammen, bei den Softwareunterprojekten sind vor allem soziale Hürden bestimmend. Die Neueinsteiger dürfen nicht ungefragt in einem anderen Unterprojekt Quellcode verändern, sondern müssen erst patches einsenden, um sich bei dem Projektkoordinator und den weiteren Mitstreitern bekannt zu machen. Dies kann als eine Art Probebeschäftigung gesehen werden. Aus der Sicht des jeweiligen Softwareprojekts sind Neueinsteiger von außerhalb und innerhalb des Großprojekts prinzipiell gleich; sie müssen sich erst bekannt machen. Der einzige Unterschied besteht, daß den internen Entwicklern durch Reputation mehr Vertrauen entgegengebracht wird und somit diese Hürde u.U. niedriger ausfällt.

Interessant ist, daß diese Eintrittshürden in den Selbstaussagen der Community nicht vorkommen. In den Selbstaussagen wird eher ausgesagt, daß es beim Eintritt keine Vorauswahl gäbe, da das Projekt auf Freiwilligkeit basiere. Außerdem darf sich jeder beteiligen, der mitarbeiten möchte.
Eine weitere wichtige Vorraussetzung sei dabei, daß ein neuer Teilnehmer im Team arbeiten könne, sich kooperativ verhalten würde und sich dem Konsens- und Transparenzprinzip unterwerfen würde. Dies steht im Gegensatz zur Existenz von Vereinzelten im Großprojekt, die allein ein Subprojekt vorantreiben. So kommt es, daß es nicht nur im Umfeld von dem Großprojekt Ein-Personen-Projekte gibt, sondern auch in dem Großprojekt selbst. Dabei gehören diese Kleinstprojekte zur Peripherie in dem Großprojekt.

5.1.2. Eintritt in den Kern/inneren Kreis

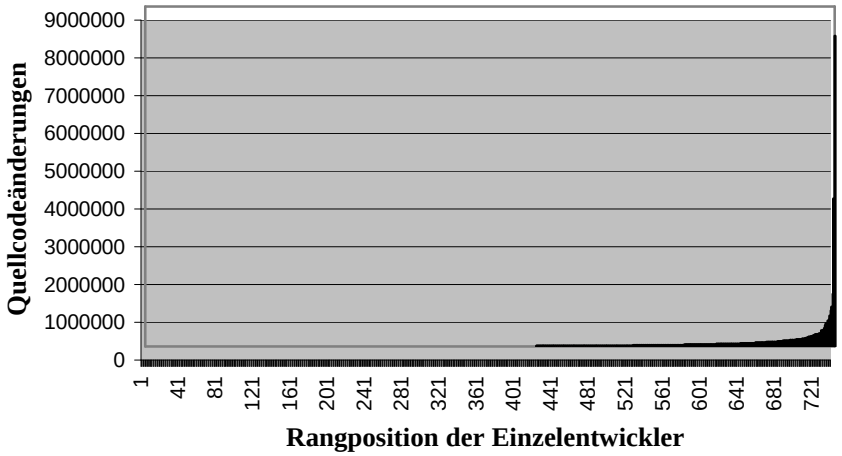
Zusammengefaßt ist der Einstieg in das Open Source-Projekt ein gestufter Eintritt. Erst muß sich bei den Entwicklern die Schreibberechtigung für die Dateiablage verdient werden. Nachdem man die erste relative einfache Hürde genommen hat, kommt der schwierige Aufstieg in den inneren Kreis des Projekts. Bei diesem Aufstieg muß durch ständige Mitarbeit Reputation erworben werden, um im inneren Kreis anerkannt zu sein. Dabei ist die Reputation das Selektionskriterium, nach dem die Projektbeteiligten ihren sozialen Status in dem Großprojekt festlegen. Der Unterschied zwischen den verschiedenen Personen mit ihrem sozialen Status ist

fließend, weswegen der innere Kreis von seiner Umgebung nicht strikt getrennt ist; seine Grenze ist fließend.

5.1.2.1. Reputation

Reputation oder besser soziale Anerkennung erhält man Schritt für Schritt durch Leistung, kooperative Kommunikation, Seniorität/Erfahrung und Sichtbarkeit/Involviertheit.
Unter Leistung versteht man einen hohen, kontinuierlichen Arbeitsaufwand und die Bearbeitung wichtiger bzw. dringlicher Arbeitsbeiträge. Die Arbeitsbeiträge der jeweiligen Person können dabei im Dateiablagesystem identifiziert werden. Somit stellen die Projektbeteiligten durch einen regelmäßigen Blick in das System Personen mit kontinuierlichen Arbeitsbeiträgen fest. Eine Annäherung an den Arbeitsaufwand stellen die veränderten Quellcodezeilen pro Entwickler im Laufe der Projektlebenszeit dar (s. Diagr. 1). In dem Diagramm fällt eine starke Hierarchisierung auf, da von 740 eingetragenen Entwicklern nur ca. 20 mehr als 1 Millionen Quellcodeveränderungen gemacht haben. Die Betonung des ständigen, hohen Arbeitseinsatzes zeigt sich auch gerade in den Selbst-Interviews aus der Community (s. Anhang A), bei denen eine sehr wichtige Frage die Frage nach dem geleisteten ist, dem „clame to fame“.

Diagr. 1: Verteilung der Mitarbeit bei dem Open Source-Projekt[42]



Eine erhöhte Reputation bekommt der, der wichtige bzw. dringliche Arbeitsbeiträge, z.B. für die Stabilität eines Programms, ständig schnell bearbeitet. Weiter gehören zu wichtigen Arbeitsbeiträgen, wenn neue schwierige und aufwendige Entwicklungsideen in der Software realisiert werden. Dabei spielt die Komplexität der Arbeitsaufgabe mit hinein. Außerdem müssen diese Arbeitsbeiträge einer hohen Qualität des selbstgeschriebenen Quellcodes genügen. Hohe Qualität bedeutet, daß der Code relativ einfach geschrieben und sofort verständlich sein soll. Zusätzlich darf der neue Code anderen keine Arbeit machen, indem dieser viele Fehler erzeugt. Die Arbeitsbeiträge müssen nicht unbedingt programmierte Quellcodezeilen sein, son-

[42] Ich danke Gregorio Robles für das Zur Verfügung stellen von Daten.

dern können auch anderweitige Arbeit für das KDE-Projekt sein, wie z.B. die Organisation der Repräsentation auf Konferenzen, die Organisation von großen Projekttreffen, etc.

Zusätzlich gehört zur sozialen Anerkennung ein kooperativer Kommunikationsstil. Dieser Kommunikationsstil zeichnet sich durch Freundlichkeit, einen Verzicht auf Provokationen und bei Streitigkeiten durch sachliche, gute und durchdachte Vorschläge bzw. Argumente aus. Ein sehr bekannter Entwickler kann sich sein Ansehen auch verspielen, in dem er kontrovers und unkooperativ kommuniziert bzw. argumentiert. Sympathie spielt hier eine, wenn minder wichtige, Rolle.

Die Seniorität basiert auf einer langen Beteiligungsdauer und dem daraus folgenden höheren Alter und einem daraus abgeleiteten Vorrecht. So haben die Projektgründer und alle, die seit der Gründerzeit dabei sind, eine hohe Reputation. Durch die lange Beteiligungsdauer haben die langjährigen Mitarbeiter Erfahrungen mit der Software, aber auch alle bisher entstandenen Probleme, gesammelt. Sie tragen somit vornehmlich das kollektive Gedächtnis der Community. Im Moment zeichnet sich ein Generationswechsel an, da sich die älteren Mitarbeiter langsam aus dem Großprojekt zurück ziehen und neuere Beteiligte in den inneren Kreis drängen.

Weiterhin spielt die Sichtbarkeit bzw. Aufmerksamkeit von Projektmitarbeitern und damit die Involviertheit eine große Rolle. Mit Sichtbarkeit bzw. Involviertheit wird das Vorhandensein und die Häufigkeit von Beiträgen eines Akteurs, mit Aufmerksamkeit bzw. Bekanntheit die Blickrichtung von anderen Akteuren bezeichnet. Beides geht Hand in Hand, da mit steigender Sichtbarkeit bzw. Involviertheit, die Bekanntheit und Aufmerksamkeit steigt.
Sichtbar ist man durch häufige Beteiligung an Diskussionen auf Mailinglisten, die von vielen gelesen wird, und häufige bzw. ständige Arbeitsbeiträge. Diese häufigen Arbeitsbeiträge werden in dem für alle einsehbaren Versionsmanagementsystem oder in einer speziellen Mailingliste dokumentiert. Dabei spielt das jeweilige Interesse bzw. Arbeitsgebiet eine Rolle. So sind die Entwickler, die an den Kernbibliotheken arbeiten, für die Anwendungsentwickler, die diese Bibliotheken benutzen, besonders wichtig und sichtbar[43]. Bei Mailinglisten ist es ähnlich, da die Email-Beiträge archiviert werden und Projektbeteiligte für andere unsichtbar sind, die nur in ihrer Subprojekt-Mailingliste Emails schreiben. So haben die Personen, die wichtige Arbeiten für die Community machen, wie die gemeinschaftserhaltenden Funktionen z.B. der Pflege des Versionsmanagementsystems oder der Mailinglisten, eine erhöhte Reputation. Sichtbarkeit bedeutet aber auch, daß sich die Personen schon mal persönlich begegnet sind. Dabei gilt, daß je höher die Reputation ist, desto mehr Personen aus dem Großprojekt hat das Individuum getroffen[44]. Außerdem kennen sich die Projektbeteiligten aus dem inneren Kreis auch durch größere, unregelmäßige Treffen des Großprojekts, z.B. vor größeren Releases, persönlich (s. Kap. 3.3.).

Die Projektbeteiligten können anhand des Kriteriums Reputation unterteilt werden. Das Kriterium Reputation ist eine kontinuierliche Größe, weswegen es keine scharfe Abgrenzung zwi-

[43] Hier spielt u.U. auch die Komplexität der zu bearbeitenden Kernbibliotheken mit hinein.

[44] Dies deutet sich in Anhang A an.

schen Zentrum und Peripherie gibt. Zusätzlich gibt es ein hierarchisches Gefälle von den Entwicklern zu den Dokumentierern und Übersetzern, die sich z.T. voneinander abgrenzen. Ähnliches gilt auch für die Fehlerberichterstatter; eine Minderheit aus dem Projekt erstellt die meisten, eine Mehrheit von außerhalb des Projekts die wenigsten Fehlerberichte, z.T. auch nur einen Bericht.

5.1.2.2. Projekterhaltende Posten im Open Source-Projekt und Reputation

Die Entwickler mit einer hohen Reputation gehören zum inneren Kreis. Meistens haben die Personen mit hoher Reputation aus dem inneren Kreis bestimmte gemeinschaftserhaltende Funktionen geschaffen, damit bestimmte Arbeitsaufgaben oder die Kommunikation einfacher ablaufen können. Die Kontrolle dieser Funktionen haben die Personen mit hoher Reputation dann aus pragmatischen Gründen und Getreu dem Motto, daß der Arbeitende seine Arbeit bestimmt, auch selbst besetzt. Die wichtigsten gemeinschaftserhaltenden Funktionen sind somit ungefähr seit der Gründerzeit mit der Einführung der Basiswerkzeuge vergeben worden. Viele der damaligen Personen, die die Funktionstüchtigkeit der Community aufrechterhielten, haben diese Posten heute (nach ca. fünf Jahren) immer noch inne. Also gilt das Kriterium der Seniorität auch bei der Reputation. Interessanterweise haben die Projektbeteiligten, die für diese Funktionen zuständig sind, Arbeit bei open-source- bzw. großprojekt-nahen Unternehmen gefunden.

Solche Posten werden per vorheriger Absprache an andere Projektbeteiligte mit relativ hoher Reputation weiter vergeben, d.h. der bisherige Funktionsträger gibt seine Funktion an eine Person aus dem inneren Kreis, die sich bereit erklärt[45]. Die hohe Reputation signalisiert durch die langjährige Beteiligung, daß eine Bindung an und Kenntnisse über die Community sowie die wichtigen Entwicklungstools besteht[46]. Zusätzlich bringt dieser Posten hohe Verpflichtungen und z.T. hohen Arbeitseinsatz mit sich, so daß es schwierig ist, Nachfolger zu finden.
Viele der Projektbeteiligten, die für die abhängigen Unternehmen an dem Großprojekt arbeiten, haben solche wichtigen Posten. Dies einerseits, da sie die meiste Zeit dafür aufbringen können. Sie können in der gesamten oder in Teilen ihrer Arbeitszeit, aber auch in ihrer Freizeit, an dem Projekt arbeiten und halten so ihre Reputation auf hohem Niveau. Dadurch haben diese einen Reputationsvorteil gegenüber Freizeitarbeitern. Andererseits wurden sie erst später wegen ihrer relativ hohen Reputation und guten Kenntnisse des Produkts angestellt, was diese Posten signalisierten. Viele dieser Gruppe, der für eine Firma an dem Projekt Arbeitenden, gehören aus den genannten Gründen zum inneren Kreis.

Mit dem Erwerb von Reputation steigt auf der einen Seite der Einfluß bzw. die Durchsetzungsmöglichkeiten in der Community, aber auch der Arbeitsaufwand an. Auf der anderen Seite geht ein Anstieg von Verantwortung und Verpflichtung der Community einher. Zusätz-

[45] Dieses Vorgehen hat starke Ähnlichkeit mit Verhandlungen und Absprachen in Netzwerken bzw. sozialen Gruppen, wie Parteien, Vereine, etc., wobei sich Entscheidungen im Plenum mit Überzeugungsarbeit und bilateralem Taktieren abwechseln.

[46] Jemand mit hoher Reputation erhält wegen seiner Reputationshöhe einen solchen Posten, wodurch sich diese Reputation wiederum stark erhöht.

lich steigt die Erwartungshaltung der anderen Mitbeteiligten an Jemanden mit hoher Reputation. So wird z.B. bei einem Konflikt erwartet, daß diese helfen diesen beizulegen.
Eine hohe Reputation erhält man nur mit der Zeit. So ist nicht verwunderlich, daß die Beteiligungsdauer, d.h. Seniorität, an dem Großprojekt wichtig ist. Dabei spielt die Beständigkeit mit hinein.
Die Beteiligten an dem Großprojekt sind dann im inneren Kreis angekommen, wenn sie in einer internen, nach außen geschlossenen Mailingliste aufgenommen werden. Diese Mailingliste ist nur für bestimmte Mitglieder, d.h. sie kann nur von bestimmten Personen gelesen und angeschrieben werden.

So läßt sich zusammenfassend sagen, daß der innere Kreis im Moment einer „gewachsenen Oligarchie“ bzw. informellen Hierarchie durch sozialen Status der älteren und anfänglichen Projekt-Beteiligten gleicht, die durch bestimmte Mailinglisten zusammengehalten werden. Ihre Einflußmöglichkeit wird durch ihre Reputation und die Übernahme von gemeinschaftserhaltenden Posten erhöht. Dabei spielen die angestellten Großprojekt-Programmierer eine wichtige Rolle in dieser Kerngruppe.

Ähnliches gilt für die Übersetzer/Dokumentierer, die auch Reputation durch Leistung, kooperative Kommunikation, Seniorität und Sichtbarkeit erhalten. Die Reputation wirkt einerseits in den jeweiligen Projekten, die sich um die Sprachen entwickelt haben. Andererseits wirkt sie zwischen diesen Projekten. In einem Projekt einer Sprache kann man den Arbeitseinsatz einer einzelnen Person aber nicht über ein Tool messen, da die einzelnen Personen ihre Arbeitsbeiträge an zentrale Koordinatoren senden und von denen ins Dateiablagesystem einpflegen lassen. Der Projektbeteiligten des jeweiligen Projekts, vor allem der Koordinator, der die Übersetzungen in die Dateiablage ablegt, lernen aber die Mitarbeiter kennen und können so Reputation zuschreiben. Ein Vergleich zwischen den Übersetzungsprojekten besteht aber über die Möglichkeit, die Veränderungen der einzelnen Sprachen im Dateiablagesystem zu verfolgen. Dort wird der Übersetzungsstatus dargestellt. So kann die Reputation zwischen den einzelnen Sprachen festgestellt werden. Dabei sind die deutsche, französische und englische Übersetzungsprojekte am aktivsten.
Bei den Übersetzern und Dokumentierern gab es auch einen inneren Kreis, der vergleichbar mit dem inneren Kreis der Entwicklergruppe war. Auf der obersten Ebene wurden die Übersetzer und Dokumentierer durch ein zentrales Gremium koordiniert, wobei die Personen aus dem inneren Kreis diese Position durch Reputation bekommen hatten. Von diesem zentralen Gremium ist nur noch eine Person übrig geblieben. Diese zentrale Person ist für bestimmte wichtige gemeinschaftserhaltende Funktionen zuständig, wie die Überprüfung des Fortschritts der Übersetzungen der verschiedenen Sprachen, die Wartung der Server/Systeme und der Werkzeuge. Dieser Hauptverantwortliche für die Übersetzer und Dokumentierer übernimmt die strategische Koordination und ist die Hauptkontaktperson für die Projektkoordinatoren der Übersetzungen.

5.1.3. Austritt aus dem Großprojekt

5.1.3.1. Freiwillige Austritte

Die Austritte aus dem Großprojekt können in freiwillige Austritte und in Zwangsaustritte unterteilt werden. Die freiwilligen Austritte sind am Anfang des Einstiegs ins Projekt am höchsten. Aus den eintretenden Personen werden die herausgefiltert, die eine hohe Beteiligungsbereitschaft versprechen. Nachdem der Neueinsteiger die zwei Stufen der Selbstmotivation bezwungen hat, muß er eine weitere Stufe der Selbstmotivation erklimmen. Diese Stufe bedeutet, daß die Person sich kontinuierlich aus eigenem Antrieb beteiligt und für das Projekt arbeitet. Viele Neueinsteiger können diese kontinuierliche Motivation nicht aufbringen, arbeiten nach dem Erhalt des Schreibzugriffs für das Versionsmanagementsystem nur kurze Zeit mit und hören dann auf. Von denjenigen aber, die diese Hürde der kontinuierlichen Selbstmotivation genommen haben, verlassen nur wenige das Großprojekt. Folglich ist die Austrittshürde wegen der Freiwilligkeit der Beteiligung, d.h. der Selbstmotivation, niedrig.
Viele Austritte der Entwickler aus der Peripherie sind nicht beobachtbar, da ein Austritt erst durch eine dauerhafte Nicht-Beteiligung an der Arbeit am Dateiablagesystem feststellbar wird. So werden die Zugänge zur Dateiablage regelmäßig überprüft und längere Zeit nicht benutzte Zugangsberechtigungen gelöscht. Zusätzlich wird die Abwesenheit der unbekannteren Personen nur von ihrem direkten Umfeld bemerkt. So fallen verwaiste Ein-Personen-Softwareprojekte kaum auf.
Die Zahl der Austritte sinkt mit der ansteigenden Beteiligungsdauer an dem Großprojekt. Dies ist an eine steigende Bindung an die Community durch das Gefühl der Verantwortung und Verpflichtung sowie durch die steigende Reputation gebunden. Zugleich steigt die Erwartungshaltung der anderen Projektmitarbeiter an die eigene Person. So ist die Bindung an die Community bei Projektbeteiligten aus dem inneren Kreis am höchsten, da diese sehr lange beteiligt sind, Posten mit community-erhaltender Funktion inne haben und soziale Anerkennung erworben haben. Diese Projektbeteiligten aus dem inneren Kreis geben den aufgewendeten Arbeitsaufwand für das Großprojekt und die darausfolgende Reputation ungern auf. Zusätzlich wurden einige aus dem inneren Kreis für die Arbeit an dem Großprojekt angestellt. Daraus erschließt sich, daß diese angestellten Entwickler mit hoher Reputation ein relativ konstantes Element in dem Projekt darstellen, solange sie für das Projekt angestellt sind.

Diese Beharrungskräfte, die die Projektbeteiligten im Großprojekt halten, werden aber durch gegenläufige Trends aufgehoben. So sind für die freiwilligen Austritte fast immer nur projektexterne Gründe ausschlaggebend[47]. Diese projektexternen Gründe beziehen sich besonders auf das Freizeitverhalten der Projektbeteiligten, da eine Mehrheit in ihrer Freizeit an dem Großprojekt arbeitet. Bei der großen Gruppe der Studenten bewirkt die Verringerung der Dauer der Freizeit durch die Aufnahme einer Erwerbsarbeit und der dann mangelnden Zeitflexibilität der Arbeitserstellung Austritte. Dazu gehören auch Abwerbungen von projektnahen Unternehmen, wie Distributoren oder der Firma der Baisbibliothek. Bei der Gruppe der Erwerbsarbeiter, die neben ihrer Vollzeit-Erwerbsarbeit in ihrer Freizeit für das Großprojekt

[47] Projekt-interne Gründe wurden mir für freiwillige Austritte nicht berichtet.

programmieren, ergibt sich eine veränderte Prioritätensetzung der Freizeit durch Partnerschaft, Familiengründung, Hausbau, o.ä.
Das Vorhandensein einer großen Gruppe von Studenten bzw. von ungebundenen, jungen Männern, die später durch eine veränderte Prioritätensetzung der Freizeit und Verringerung der Freizeitdauer das Großprojekt verlassen, gibt das Großprojekt als ein Lebensphasenphänomen zu erkennen.
Diese Feststellung wird durch den momentanen Generationswechsel im inneren Kreis gestützt. Die Gründer und einige andere langjährige Mitarbeiter ziehen sich langsam aus dem Projekt zurück, da sie sich mehr auf die Familiengründung und Erwerbsarbeit konzentrieren wollen und müssen und dadurch wenig Zeit für die Arbeit an dem Projekt haben. Nach und nach werden wichtige Positionen zu community-erhaltenden Funktionen an die neue Generation weiter gegeben.

5.1.3.2. Zwangsaustritte

Zwangsaustritte bei den Entwicklern

Neben den freiwilligen Austritten gibt es auch Zwangsaustritte. Die Zwangsaustritte erfolgen vor allem aus projekt-internen Gründen, sind aber eine Seltenheit in dem Großprojekt. Diese Zwangsaustritte sind als Sanktionen gegen Normübertretungen zu sehen. Der wichtigste Sanktionsmechanismus besteht in der Sperrung des Schreibzugriffs auf das Versionsmanagementsystem. Ein abgeschwächter Sanktionsmechanismus ist der Ausschluß bzw. die Ablehnung von Arbeitspaketen aus dem Dateiablagesystem bzw. Release. Der Ausschluß von Personen aus einer oder mehrerer Mailinglisten wird dabei als Sanktionsmechanismus nicht verfolgt[48]. Der weitere Sanktionsmechanismus, die Sperrung der personalisierten Projekt-Emailadresse, wurde bisher auch nicht angewendet.
Die Sanktion der Sperrung des Schreibzugriffs wird bei verschiedenen Normenbrüchen angewandt. Es gibt z.B. die Norm, daß sich jeder eigenverantwortlich das nötige Wissen anhand bestehender Beschreibungen auf der Homepage oder Archiven von Mailinglisten aneignet. Diese Norm fällt besonders ins Gewicht, wenn Neueinsteiger mit dem Versionsmanagementsystem oder anderen wichtigen Werkzeugen wiederholt falsch umgehen und die Stabilität des Systems gefährden, weswegen den Neueinsteigern der Zugriff zu dem System entzogen wird. Sie werden aus dem System als erzieherische Maßnahme ausgesperrt, um größeren Schaden vom System abzuhalten und um anzuregen, den Umgang mit dem System nachzulesen. Nachdem die Neueinsteiger glaubhaft versichert haben, daß sie dies nachgeholt haben, wird ihnen der Zugriff wieder erteilt.
Eine weitere wichtige Norm des Großprojekts, aber auch der gesamten Open Source-Community (s.o.), betrifft den Quellcode, der im Dateiablagesystem verwaltet wird. Dabei gilt, daß Quellcode, der im Versionsmanagementsystem veröffentlicht wurde und der GPL untersteht (s.o.), ein öffentliches Gut ist und das Großprojekt als Vertreter der Öffentlichkeit diesen Quellcode verwaltet und weiterbearbeitet. Die Projektbeteiligten können dabei Quellcode auch ablehnen. Gerade aber der Quellcode, der in vorherigen Releases offiziell für die Öffentlichkeit als eigene Version freigegeben wurde, gehört der Öffentlichkeit und somit auch dem Großprojekt und kann nicht mehr zurückgenommen werden. Diese Norm gilt auch bei

[48] Die Sanktion wird wegen der geforderten Transparenz auf den Mailinglisten nicht verfolgt.

Mitarbeitern, die schon länger dabei sind und sich eine hohe Reputation erworben haben. Der Normbruch wurde ausgelöst durch einen Konflikt um die Einhaltung des Releaseplans. Beim Releaseplan werden alle Tätigkeiten im Hinblick auf die offizielle Veröffentlichung des Produkts, d.h. Release, aufeinander abgestimmt. In diesem Fall wurden kurz vor dem endgültigen Termin des Releases Softwareteile in das Dateiablagesystem eingestellt. Diese Softwareteile wurden wegen der Mißachtung des Release-Terminplans und rechtlicher Bedenken[49], was dem Entwickler mitgeteilt wurden, aus dem Dateiablagesystem entfernt und so von der Veröffentlichung in diesem Release ausgenommen. Dies hat diese Person geärgert, so daß dieser seinen gesamten Quellcode gelöscht hat, den dieser geschrieben und im Dateiablagesystem veröffentlicht hatte. Nach zweimaligem Löschen und Wiederherstellen[50] des Quellcodes wurde der regelbrechenden Person der Schreibzugriff entzogen. Ganz aufgegeben wurde in diesem Fall der Regelbrecher von den anderen Projektbeteiligten nicht, da er seine Projekt-Emailadresse behalten durfte. Das deutet darauf hin, daß diese dem Übeltäter früher oder später verzeihen und ihm den Dateiablage-Zugang wieder geben werden. Eine Versöhnung mit den Projektbeteiligten muß sich der Regelbrecher durch Arbeitseinsatz verdienen. Nach diesem schweren Regelvergehen hat der Regelbrecher den Zugriff nach längerer Zeit noch nicht zurückbekommen, obwohl er wieder Quellcode auch für das untersuchte Projekt erstellt.

Zwangsaustritte bei den Übersetzern

Auch bei den Übersetzern werden Regelvergehen durch Ausschlüsse geahndet. Diese Norm besagt, daß die Kommunikation neutral sein soll. Es soll keine politische Meinung über andere Nationen, die diese beleidigen geäußert werden, da damit die Arbeit beeinträchtigt wird. Diese Kommunikationsregel wurde auch im Bezug zu einer neuen Mailingliste ohne festgelegtes Thema deutlich, da einige der Interviewten ihr Mißfallen über die Diskussionen über nationale Identitäten und Stereotypen äußerten. Bei dieser Regelverletzung ging es um eine arabische Übersetzung, die von Palästinensern gemacht wurde. Einer dieser Übersetzer hat in der arabischen Version Israel mit occupied Palestine übersetzt. Dies wurde von den Koordinatoren der israelischen und arabischen Sprache ohne Absprache mit dem Übersetzer korrigiert[51]. Der Ausschluß erfolgt hier aber nicht über eine technische Hürde, sondern über soziale Kategorien bzw. Widerstand im Übersetzungsteam. Die Koordinatoren sind dabei die kontrollierenden, zentralen Personen, die vergleichbar Torwächtern den Eintritt erlauben oder verwehren. Dieser Konflikt zeigt auch, daß einseitige, nicht-neutrale politische Sichtweisen in dem Großprojekt nicht geduldet werden, da hier nur die Arbeit bzw. Übersetzung zählt.

5.2. Ablauf der Leistungserstellung, interne Verteilung der Arbeitsaufgaben und Kommunikation

Die Aufgabenverteilung kann von der Leistungserstellung nur bedingt getrennt werden, da es eine operative und eine strategische Ebene gibt, auf denen verschiedene Arten von Aufgaben

[49] Einige Projektbeteiligten hatten Bedenken, da das Programm bzw. Design sehr an das geschützte Design einer Firma erinnerte und sie deswegen schon mal Ärger mit dieser Firma hatten.

[50] Der alte Quellcode kann durch eine spezielle Funktion im Versionsmanagementsystem wieder hergestellt werden.

[51] Der Konflikt hat im Ganzen eine Woche und fünf Emails gedauert.

verteilt sind. Die operative Ebene bezeichnet die alltägliche, konkrete Softwareproduktion, bei der die konkrete Aufgabenverteilung in der Hand des Individuums liegt. Auf der strategischen Ebene sind die nicht-alltäglichen, außergewöhnlichen, abstrakteren Aufgaben angesiedelt, wobei die Entscheidungen von einer hierarchischen Instanz gefällt werden. In den beiden Unterkapiteln wird auf diese Ebenen eingegangen. Am Ende des Kapitels wird dann die Nutzung der Informations- und Kommunikationsmedien beleuchtet.

5.2.1. Operative Ebene

5.2.1.1. Arbeitsaufteilung und Softwareentwicklung auf der operativen Ebene

Auf der operativen Ebene erfolgt die Softwareentwicklung normalerweise Schritt für Schritt in einem trial-and-error-Prozeß (s. Anhang A). Jeder programmiert aufbauend auf dem Code aus dem Dateiablagesystem neue Software meistens zu Hause und spielt es in das Versionsmanagementsystem zurück. Es erfolgt also ein ständiger Zugriff auf das Versionsmanagementsystem, so daß alle Projektbeteiligten den gleichen Informationsstand haben. Dabei wird bei der Programmierung keinem festgelegtem Schema gefolgt, sondern die Kreativität der Lösung steht im Vordergrund. Ein Großteil der Quellcodeveränderungen sind eher kleine Arbeitsbeiträge, d.h. wenige Quellcodezeilen, wie patches. Viele dieser kleinen Veränderungen werden durch die Bugreports angestoßen, die bei großen und wichtigen Programmen ständig eingehen. Die Arbeitsbeiträge werden normalerweise andauernd abgeliefert, das Programm ständig ausgebaut, Fehler korrigiert und weitere Funktionsweisen angefügt. Nachdem der Quellcode fertig gestellt wurde, wird der Code anderen zur Überprüfung zugesendet oder ins Dateiablagesystem gestellt und dort von den anderen begutachtet. Auf dieser Ebene werden Entscheidungen gefällt.

Die Arbeitsaufteilung auf der operativen Ebene wird erstens durch ein selbstorganisiertes und selbstbestimmtes Arbeiten und zweitens durch einen Leistungsethos bestimmt.
Das selbstorganisierte und selbstbestimmte Arbeiten wird von den Projektbeteiligten erwartet. Aus diesem Grund gibt es im Großprojekt die Maxime: „The one who codes decides!“ Das bedeutet, daß selbstbestimmt an den jeweiligen Aufgaben gearbeitet wird und jeder die Höhe seines Arbeitsbeitrages und seiner Arbeitszeit selber nach seiner Motivation festlegt. Die Leistungserstellung erfolgt freiwillig, weswegen die Leistungserstellung vorwiegend in der Freizeit abläuft. Freiwilligkeit bedeutet hier, daß die meisten Mitarbeiter ohne Arbeitsvertrag und ohne Vorgesetzte arbeiten und diese Arbeit als Hobby sehen. Dies gilt auch für die Vielen, die von einem Distributoren für die Arbeit an dem Großprojekt engagiert wurden[52]. Außerdem kann jeder sich seine Aufgabe selbst aussuchen.

[52] Diese Spezialgruppe der Entwickler hat normalerweise einen normalen Arbeitsvertrag. Im Allgemeinen kontrolliert die Firma die Arbeit an dem Großprojekt nicht und regiert nicht hinein, d.h. die Firma gibt dem Arbeitenden freie Hand für die Arbeit, obwohl dies nicht im Arbeitsvertrag steht. Es kann aber auch im Arbeitsvertrag festgeschrieben sein, daß für die Arbeit an dem Großprojekt die Weisungsbefugnis aufgehoben ist. Das Interesse der Firma liegt eher darin, daß diese Vollzeitarbeiter die Software für die nächste Distribution voranbringen und daß das Know-how über die Software sowie das Wissen über die Community im Unternehmen bleibt.

So gibt es viele Hilfestellungen der etablierten Projektbeteiligten für Neueinsteiger auf der Homepage, wie einige Texte für die ersten Schritte und ersten Orientierung[53]. Die Neueinsteiger werden dazu von den etablierten Projektbeteiligten angehalten, sich erst das nötige Wissen und die Orientierung über die Community anzueignen, bevor sie anfangen zu Programmieren, die Werkzeuge benutzen und Fragen auf den Mailinglisten stellen. Dazu sollen sie auch die Mailinglisten-Archive nach ihren Fragen durchsuchen, da manchmal Fragen schon mal dort beantwortet wurden.

Die Softwareerstellung kann nicht ohne Absprachen funktionieren, da sonst parallel am gleichen Problem gearbeitet wird, was bei kleinen Problemen öfters auftritt. Es kann vorkommen, daß zu einem Fehler zwei patches geschrieben wurden, die fast zeitgleich eingehen. Einer dieser beiden patches wird dadurch ausgewählt, daß von den Projektbeteiligten des jeweiligen Projekts über die Qualität auf der Mailingliste im Konsens entschieden wird.
Auf der operativen Ebene werden manchmal auch weitere Entscheidungen diskutiert und getroffen, die die zukünftige Entwicklung des Subprojekts betreffen. Dabei beschließen alle Projektbeteiligten des Unterprojekts über neue Funktionalitäten der Software oder die von Grund auf Neuprogrammierung des Programms, da z.B. ein Technologiewechsel erfolgen soll und so die Schnelligkeit eines Programms sich verbessern läßt. Es gibt bei der Arbeitsverteilung auch Absprachen über die Übernahme von Aufgaben zwischen mehreren Personen. Dabei muß auf die Einhaltung von Absprachen vertraut werden, da keiner den anderen zur Übernahme von Arbeiten zwingen kann. Nur Reputation steigert das Vertrauen, daß der andere sein Wort hält.
Der Einfluß auf Entscheidungen unterscheidet sich nach der Stellung des Argumentierenden in der Community. Die Entscheidungen über die Erstellung des Codes werden vor allem nach dem Konsens- und Transparenzprinzip gefällt. Vor der Entscheidung werden die verschiedenen Argumente und Alternativen abgewogen. Die Reputation wirkt dabei auf die Entscheidungsfindung ein. Dabei wiegt das Argument einer Person mit Reputation schwerer als das eines anderen. Doch können diese mit guten Argumenten überstimmt werden.

Dieses selbstorganisierte und selbstbestimmte Arbeiten gilt auch für die Übersetzer bzw. Dokumentierer. So mißfällt den deutschen Entwicklern die deutsche Übersetzung ihrer Programme, regieren aber nicht in die Arbeit der Übersetzer hinein. Außerdem haben die Entwickler kein Mitspracherecht bei in der Koordination der Übersetzungsgruppe.

Bei der Arbeit für das Open Source-Projekt spielt ein Leistungsethos eine große Rolle, da sich über Leistung Reputation in der Community verdienen läßt. Dieser Leistungsgedanke drückt sich in den erstellten Arbeitsleistungen aus, die in dem Versionsmanagementsystem auf einen speziellen Projektbeteiligten zurück geführt werden können. Außerdem wurde bei den Interviews öfters betont, daß jeder das copyright auf seinen Quellcode behält. Dies ist ein Fingerzeig auf die Kontrollierbarkeit der Leistungserstellung, da der Quellcode, sobald er im Versi-

[53] Es gibt für alle Tätigkeitsbereiche, wie Entwickeln, Übersetzen, Dokumentieren und Graphikerstellung, eigene Vorschriften, wie z.B. styleguides, Hilfen (FAQs = frequently asked questions), Einweisungen in Werkzeuge, etc.

onsmanagementsystem steht, der Öffentlichkeit in Form der Open Source-Community gehört (s. Kap. 2.3.2.).
Doch gehören auch zur Reputation Verpflichtungen, so wird z.B. die ständige Korrektur von gemeldeten Fehlern, Kommentare zu bzw. Schlichtung von Diskussionen erwartet. Hohe Reputation heißt also auch hoher Arbeitsaufwand. So steht ein Open Source-Projektmitarbeiter in einem Spannungsfeld zwischen Freiwilligkeit und Erwartung der Community.
Die Restriktionen bei der Produktion und somit die Hauptressourcen im Projekt sind das Ausmaß der Freizeit bzw. die Arbeitszeit für das Großprojekt, das Wissen über die Software bzw. den Aufbau der Community und die Motivation.
Die Produkterstellung wird auch durch Vertrauen erleichtert. Es wird darauf vertraut, daß jeder die gleichen Ziele der gemeinsamen Produkterstellung hat und sich an die Normen der Community hält. Dies geht Hand in Hand mit einer kooperativen Kultur ohne Störungen und Trittbrettfahrer bei der Softwareentwicklung.

Die obigen Aussagen gelten auch für Übersetzer und Dokumentierer. So liegt die konkrete Übersetzungsausführung in den Händen der Arbeitenden selbst.

5.2.1.2. Rolle des Projektkoordinators bei der Arbeitsaufteilung und Leistungserstellung

Projektkoordinator bei Entwicklern

Der Posten des Projektkoordinators existiert seit dem Anfang des Großprojekts. Das Open Source-Projekt wuchs sehr schnell, so daß von Anfang an Absprachen mit mehreren Mitarbeitern nötig waren, weswegen sich in der Open source-Community kurz nach der Gründung koordinierende Strukturen herausbildeten.
Am Anfang des Projekts, vor der Einführung des Dateimanagementsystems, hatten einige Gründungsmitglieder eigene Teilprogramme bzw. Teilprojekte. Diese Gründungsmitglieder waren die Koordinatoren ihres Programms und somit verantwortlich dafür. Diese Projekte erstellten unkoordiniert, zu verschiedenen Zeitpunkten einen Release von ihrer programmierten Software und stellten diesen den anderen auf einem Internetserver zur Verfügung. Die anderen Beteiligten testeten den Release und sandten dem Koordinator z.B. patches oder neuprogrammierte Softwareteile zu. Mit diesem Modell wurde eine kritische Masse an funktionierendem Quellcode gebildet, den andere nutzen konnten, um daran weiter zu arbeiten. Das neue Versionsmanagementsystem unterstützte die Verbreitung dieser kritischen Masse unter den Interessierten, womit die Weiterentwicklung des Codes eine Art Selbstläufer wurde.
Einige dieser Regeln, die sich durch die technische Struktur formiert haben, sind heute immer noch in der Community vorhanden. So haben alle Softwareunterprojekte oder gemeinschaftserhaltende Funktionen einen Koordinator bzw. Maintainer, der in einer Liste auf der Homepage eingetragen wird. Bei großen, komplexen Softwareprojekten werden diese in kleinere Bereiche aufgeteilt, wobei jeder Bereich einen Koordinator hat, die sich untereinander absprechen.
In den Subprojekten, die alle von Projektleitern geleitet werden, erfolgt die konkrete Leistungserstellung. Bei besonders großen Programmen wurden verschiedene kleinere Subprojekte für bestimmte Softwareteile erstellt. Gabelungen von (Sub-)Projekten sind in dem KDE-Projekt kaum aufgetreten. Der Projektkoordinator ist der Gründer eines Programms, der den

ersten Code geschrieben hat. Die Personen mit hoher Reputation bzw. aus dem inneren Kreis sind oftmals Projektleiter eines wichtigen Projekts, da sie den ersten Quellcode dazu geschrieben haben. Nach mehreren Jahren des Bestehens des Open Source-Projekts haben aber die Projektleiter gewechselt und andere Personen mit hoher Reputation haben die Stelle übernommen.
Es gibt aber auch einige Projekte, an denen nur der Gründer bzw. Koordinator arbeitet. Bei größeren Projekten arbeiten neben dem Koordinator oft weitere Entwickler mit.
Der Koordinator muß für einen Nachfolger sorgen, wenn er keine Zeit mehr für die Bearbeitung des Projekts hat. Dies ist nicht immer möglich, weswegen das Unterprojekt verwaist, wenn sich kein Nachfolger freiwillig meldet. Ein länger verwaistes Unterprojekt darf nach Anfrage und Einverständnis des ursprünglichen Koordinators oder wenn sich dieser nicht meldet übernommen werden. Dabei kann es zu Streitigkeiten zwischen dem neuen und dem alten Koordinator kommen, wenn jemand ein Subprojekt übernimmt und der alte Koordinator Anrechte anmeldet. Diese Konflikte gingen meistens zugunsten des neuen Projektleiters aus. Oftmals wurde auch geeigneten neueren Projektmitarbeitern eine Subprojektleiterposition von Personen mit hoher Reputation vermittelt.

Der Projektkoordinator hat auf der operativen Ebene verschiedene Funktionen. Erstens ist der Projektkoordinator der Ansprechpartner und Verantwortliche für die restlichen Entwickler, der den Ablauf der Programmierung überwacht. Er ist ein intimer Kenner seines Programms und kann so sein Wissen weitergeben.
Außerdem organisiert er Diskussionen bzw. Entscheidungen über Softwarebeiträge oder zukünftige Entwicklungen. Der Projektkoordinator hat dabei aber keine Alleinentscheidungsmacht, da sonst bei unpopulären Entscheidungen die Mitarbeiter das Unterprojekt verlassen. So werden Entscheidungen vorher mit allen Beteiligten besprochen, Argumente ausgetauscht und versucht ein Konsens herzustellen. Auch bei einer überwältigenden Mehrheit für einen Vorschlag wird dieser umgesetzt. Der Projektleiter hat aber Einfluß auf die Entscheidung, da er sein Subprojekt mit der Software kennt und Reputation für das Subprojekt erhalten hat. Zusätzlich sind auch einige Projektleiter Mitglied im inneren Kreis, weswegen sie noch mehr Reputation mitbringen. Damit ist der Projektleiter als Gleicher unter Gleichen zu betrachten, der jeweils einen eigenen Führungsstil hat, der von seinem Charakter abhängig ist, aber im Großen und Ganzen auf Kooperation ausgelegt ist. So kann der Projektleiter eher etwas autoritär auftreten, aber auch sehr kooperativ sein.
Auch andere Projektbeteiligte aus dem innerem Kreis mit hoher Reputation von außerhalb des Subprojekts können Einfluß nehmen. Doch sind diese Projektbeteiligten eher als Gleiche unter Gleichen zu sehen, da der Einfluß sich dabei als ein gut durchdachtes Argument darstellt, das aus der Kenntnis der Softwareentwicklung kommt. Aus diesem Grund kann der Einfluß auch ins Leere laufen, wenn sich eine Mehrheit dagegen entscheidet.
Der Projektkoordinator ist zweitens der Ansprechpartner für neue Mitstreiter an einem Projekt, die die soziale Eintrittshürde überwinden wollen. Er weist Ihnen fürs Erste Arbeitsaufgaben zu.
Drittens verteilt er die Fehlerberichte an interessierte Mitstreiter. Er stellt auch hin und wieder to-do-Listen auf, aus denen sich die Mitarbeitenden ihre Arbeitsaufgaben aussuchen können.

Viertens ist er für die Fortführung der Programmierung und der Fehlerfreiheit des Programms gegenüber dem Releasekoordinator verantwortlich. In vielen Fällen, vor allem bei kleineren Projekten, ist der Projektkoordinator der einzige, der viel Arbeit in die Software steckt. Damit gehen auch Erwartungen der restlichen Programmierer einher, diese Aufgaben wahrzunehmen. So wird z.B. von den Projektkoordinatoren erwartet, daß diese die Fehler ihres Programms schnell korrigieren oder bestimmte gewünschte Funktionen programmieren.

Projektkoordinator bei Übersetzern und Dokumentationsschreibern

Die Funktion des Projektkoordinators gibt es auch bei den Übersetzern und Dokumentationsschreibern. Die Übersetzer unterteilen sich nach Sprachen in Übersetzerprojekte. Die Übersetzungsteams sind aber relativ autonom von anderen Übersetzungsprojekten und der Entwicklergruppe, d.h. interne Strukturen werden vor Ort ausgehandelt, soweit dies die eigene Gruppe betrifft.

Bei den Übersetzern hat jede Sprache ein bis zwei verantwortliche Hauptkoordinatoren[54], die ihren Zuarbeitern die Arbeit zuteilen, d.h. diesen zu übersetzende Textdateien schicken oder Unterkoordinatoren für bestimmte Bereiche einsetzen. Die jeweiligen Mitarbeiter melden sich also freiwillig bei den Projektkoordinatoren der jeweiligen Sprache, bekommen dann Texte zum Übersetzen zugesandt, die sie später übersetzt zurücksenden. Hier besteht ein zentraler Arbeitsverteilungsprozeß. Bei größeren Übersetzungsteams, wie dem deutschen Team, gibt es für größere Übersetzungseinheiten wieder eigene Koordinatoren. Diese Hauptkoordinatoren haben normalerweise als einzige im Team einen Dateiablagesystemschreibzugang und bekommen deswegen die Arbeit der anderen Übersetzer zugesandt. Die Übersetzungsprojekte entscheiden selbst, wer einen Zugang zum Dateiablagesystem bekommt. Diese eingeschränkte Nutzung des Dateiablagesystems liegt an dem geringen technischen Kenntnisstand der Übersetzer, besonders beim Dateiablagesystem, und der geringen Bereitschaft sich einzuarbeiten oder einen großen Arbeitsaufwand zu betreiben.

Die Dokumentationsschreiber sind ähnlich strukturiert. Es gibt für jedes Programm einen Koordinator, der einen Zugang zum Dateiablagesystem hat. Bei größeren Programmen bzw. Dokumentationen werden diese an Unter-Projektkoordinatoren aufgeteilt. Meistens bearbeiten die Koordinatoren ihre Unterprojekte selbst, manchmal wird ihnen von Mitarbeitern geholfen. Über den Dokumentationsschreibern steht ein Administrator, der für alle Dokumentationen und für die Übersetzungsteams zuständig ist. Die Dokumentierer senden ihre Texte entweder an diesen oder an ihre direkten Projektkoordinatoren. Diese Koordinatoren kontrollieren die Dokumentation und stellen diese dann ins Dateiablagesystem.

5.2.2. Strategische Ebene

Die strategische Ebene wird durch Personen mit hoher Reputation aus dem inneren Kreis bestimmt. Die Personen aus dem inneren Kreis können auf drei Bereiche Einfluß nehmen.

Erstens bestimmen sie die operativen Entscheidungen über die zukünftigen Entwicklungen des Projekts. Zweitens nehmen sie strategischen Einfluß auf die Entscheidungen an sich. Drit-

[54] Der Erste, der in seine Sprache übersetzt, hat meist einen Dateiablagesystem-Zugang, um die Übersetzungen dort speichern zu können. Dieser sammelt dann nach und nach einen Kreis von Freiwilligen um sich herum.

tens gibt es einen institutionalisierten Einfluß auf die operativen Entscheidungen bezüglich der Softwareveröffentlichung.

5.2.2.1. Einfluß von Personen mit hoher Reputation auf die operative Ebene

Einfluß auf operative Entscheidungen und Konflikte bei der Softwareentwicklung

Die Personen aus dem inneren Kreis beeinflussen die operative Ebene bei außergewöhnlichen Anlässen. Ein solcher Anlaß ist z.B. ein eskalierter Konflikt, bei dem sich zwei gleich starke Parteien, eine pro und eine contra, gegenüberstehen. Ein Konsens konnte nach einer längeren Diskussion nicht gefunden werden, weswegen ein Unbeteiligter, meistens eine Person mit hoher Reputation, den Streit zu schlichten versucht. Von diesen Personen mit hoher Reputation wird die Schlichtung z.T. auch erwartet. Außerdem wiegen die Argumente von Projektbeteiligten mit hoher Reputation meistens schwerer als die der anderen.
Die Personen aus dem inneren Kreis fungieren also als Schiedsinstanz. Zwangsmittel können wegen der Freiwilligkeit der Beteiligung nicht eingesetzt werden, da die Hürde, aus einem Unterprojekt auszusteigen, gering ist. Dies wurde in mehreren Interviews betont. Der Aufwand ist gerade für Projektbeteiligte ohne Reputation, die kein Projektkoordinator sind, gering. Ein Ausstieg ist deswegen immer möglich. Das ist auch der Grund warum das Überzeugen in dieser kooperativen Kultur eine zentrale Rolle spielt.

Es gibt eine neuere Entwicklung in dem Open Source-Projekt. Der eingetragene Verein soll als Schiedsinstanz institutionalisiert werden. Dies liegt an dem diffusen inneren Kreis, der bisher Einfluß hatte. Das Transparenzprinzip gilt beim inneren Kreis nicht, weswegen versucht wird durch die Formalisierung des eingetragenen Vereins eine Transparenz der Entscheidungen einzuführen.

Der innere Kreis übernimmt auch eine Koordinationsfunktion. Es gibt eine Grenze zwischen Entwicklern und Übersetzern/Dokumentierern, die sich in der Nutzung unterschiedlicher Mailinglisten manifestiert. Diese Grenze wird versucht über eine spezielle Mailingliste zu überbrücken, auf der Programmierer mit hoher Reputation und die wichtigsten Übersetzer/Dokumentierer angemeldet sind. Diese Mailingliste fungiert dabei als Kommunikations- und Koordinationskanal für auftretende Konflikte zwischen den Gruppen der Entwickler und Übersetzer/Dokumentierer vor allem vor einem Release.

Einfluß auf Diskussionen über zukünftige Entwicklungen

Die Entscheidungen und Diskussionen auf der strategischen Ebene zeichnet aus, daß dort Entscheidungen über Themen gefällt werden, die das ganze Open Source-Projekt betreffen oder weitreichende Kreise im Projekt ziehen. So wird über die zukünftige Entwicklungen des Gesamtprojekts diskutiert. Dabei gehört dazu, welche Programme noch in das Großprojekt mit Desktop dazugehören sollen und was noch entwickelt werden soll. Ein Beispiel für eine solche Diskussion ist die Entscheidung über die Aufnahme von größeren Quellcode-Paketen aus dem experimentellen Bereichen des Dateiablagesystems in den Releasebereich. Die Quellcode-Pakete werden bis zu einer gewissen Stabilität programmiert, bis ein Konsens herrscht, daß diese Software in den Release aufgenommen werden kann. Ein weiteres Beispiel ist die manchmal vorkommende Entscheidung über die Aufnahme von größeren Programmen mit ih-

rem externen Programmierern[55], die technisch auf das Großprojekt aufbauen und extern erstellt wurden.

Der innere Kreis spielt eine wichtige Rolle bei der Meinungsgenerierung und Meinungsführerschaft über solchen Entscheidungen, die die zukünftige Entwicklung des Projekts berühren. Normalerweise wird die zukünftige Entwicklung des Projekts in einem größeren Kreis auf einer oder mehrer Mailinglisten diskutiert. Der innere Kreis beeinflußt diese Diskussion, indem die Personen des inneren Kreises wichtige strategische Entscheidungen im kleinen Kreis diskutieren, über bestimmte Dinge vorher abstimmen und so die anderen mit besseren Argumenten überzeugen können. Dies geschieht z.B. über eine „geheime" Emailliste, die zugleich dem Zusammenhalt dient. In dieser Liste werden vor allem nicht-öffentliche, rechtliche Fragen bezüglich der Verletzung der Lizenz besprochen. Folglich ist der innere Kreis, der ca. 20-30 Personen umfaßt, mit einer gewachsenen „Oligarchie" der älteren Beteiligten aus der Gründungszeit vergleichbar, die sich in Form eines Meinungs- und Argumentationskartells darstellt. Die Führungsgruppe ist aber nicht fest umrissen. Außerdem tritt dieser innere Kreis nicht als eine gemeinsame Gruppe auf, sondern jeder muß seine eigene Meinung auf den Mailinglisten vertreten. Die restlichen Mitarbeiter können sich deswegen auch gegen die Argumente des inneren Kreises stellen. Der innere Kreis stellt also keine „Macht- bzw. Herrschaftsstruktur" in Form eines Unternehmens oder einer Organisation mit Hierarchie dar, da dieser keine Entscheidungen durch Weisung oder durch Befehl und Gehorsam durchsetzen könnte.

5.2.2.2. Arbeitsverteilung beim Release

Stellenwert des Releases

Die normale Arbeit an der Software wird in periodischen Abständen durch die Arbeit am Release beeinflußt. Während der Vorbereitungszeit für den Release gelten spezielle Regeln zusätzlich zur normalen Programmierarbeit. Für diese Zeit wurde auf der strategischen Ebene eine koordinierende Funktion aus der Entscheidungskompetenz des inneren Kreises ausgegliedert. Diese Funktion ist der Releasekoordinator bzw. –maintainer, der für die Erstellung des Releases zuständig ist.

Der Releasemaintainer wird dabei aus dem inneren Kreis rekrutiert, d.h. muß eine hohe Reputation und den nötigen technischen Überblick haben, um diese Aufgabe zu übernehmen. Die Auswahl eines neuen Releasekoordinators verläuft wie bei der Suche nach einem neuen Projektkoordinator. Wenn der amtierende Releasemaintainer sich ablösen lassen möchte, muß er einen Nachfolger suchen. Dabei helfen ihm Personen aus dem inneren Kreis durch Tips, um adäquate Kandidaten zu finden. Die Zustimmung wichtiger Personen des Großprojekts zu der Person wird nicht unbedingt benötigt. Auch eine Abstimmung des inneren Kreises erfolgt nicht.

Der Releasemaintainer befragt mögliche Kandidaten mit hoher Reputation, nach ihrem Interesse bis er eine Zusage erhält. Die Suche gestaltet sich aber u.U. schwierig, da nicht jeder Projektbeteiligte die benötigte Arbeitszeit aufbringen kann und möchte. Einerseits bekommt

[55] Dies bedeutet für den Programmierer aus dem Umfeld ein Reputationsgewinn. Entgegen steht aber, daß einige dieser Entwickler Angst vor Verpflichtungen in der Community haben und lieber außerhalb bleiben.

der Releasemaintainer für diese Arbeit eine sehr hohe Reputation in der Community. Andererseits wird er von den Beteiligten des Großprojekts beobachtet und muß den Erwartungen gerecht werden. Außerdem ist die Koordination eines Release mit einem sehr hohen Arbeitsaufwand verbunden, vergleichbar einer Vollzeitbeschäftigung, da sehr viel Koordinationsarbeit zu machen ist. Ein Beispiel hierfür ist die Dauer der Übernahme dieses Postens. Diese Dauer hängt von der Zeitplanung und Streßfestigkeit der Person ab. Im Schnitt haben die Releasemaintainer maximal ein bis zwei Releases hintereinander diese Funktion übernommen. Aus diesem Grund wurden schon Releasekoordinatoren von Distributeuren für diese Aufgabe bezahlt.
Er wird aber durch eine Gruppe erfahrener Entwickler unterstützt, da er sich wegen der Masse an Anfragen nicht um alle Probleme sorgen kann. Diese Gruppe rekrutiert sich vor allem aus Personen des inneren Kreises. Auch der oberste Koordinator der Übersetzer und Dokumentierer unterstützt den Releasekoordinator, in dem er auf die Einhaltung des Releaseplans unter den Übersetzern und Dokumentierern achtet.

Aufgaben des Releasekoordinators beim Release

Für die Releasekoordination hat der Releasemaintainer mehrere Aufgaben inne. Die Aufgabe des Releasekoordinators ist erstens, einen Koordinationsplan bzw. Releaseplan, an den sich alle zu halten haben, für den Ablauf des Release zu entwerfen und dessen Einhaltung zu kontrollieren. Der Releaseplan wurde eingeführt, um die Tätigkeit der Übersetzer und der Entwickler aufeinander abzustimmen, da die Übersetzer von den geschriebenen Texten der Entwickler abhängig sind. Mit dem koordinierenden Releaseplan sind die Übersetzungen zeitgleich mit dem überarbeiteten Quellcode fertig.
Der Releaseplan wird am Anfang vom Releasekoordinator aufgestellt, der auf der Projekt-Homepage veröffentlicht wird. Der Releaseplan stellt eine gestufte Einschränkung der Veränderungsmöglichkeiten in dem Versionsmanagementsystems dar, d.h. es gibt verschiedene Phasen, in denen nacheinander die Anzahl der Quellcodeveränderungen bis zu einem kompletten Einfrieren des Dateiablagesystems verringert werden. Diese Markierung und der Plan sind keine technischen Hürden, sondern sie sind Symbole für Verhaltensregeln gebunden, an die sich alle Beteiligten halten sollen. Nach der Setzung einer Markierung kann also immer noch Quellcode abgelegt werden, darf aber nicht[56]. Der Releasekoordinator muß also die Einhaltung dieser Einschränkungen kontrollieren und Übertretungen ahnden, d.h. den Quellcode aus dem Dateiablagesystem entfernen. Der Releaseplan als Verhaltensnorm wird i.d.R. von allen Beteiligten befolgt. In den Interviews wurde von einem einzigen Fall berichtet, in dem

[56] Als erstes erfolgt das Verbot neue Funktionsweisen einzubauen (feature freeze), wobei sich diese Phase in ein Verbot für neue, unabgesprochene features und später für geplante features unterteilt. Ab dem kompletten feature freeze werden die Programme fehlerfrei gemacht; patches, d.h. Quellcodeveränderungen, müssen dann genehmigt werden. Danach kommt das Verbot Zeichenketten zu verändern oder hinzu zufügen oder neue Dokumentationen ein zu stellen (message freeze), da dies kurz vor dem Releasedatum angesetzt ist. Ab dem message freeze wird nur noch wenig am jeweiligen Programm verändert. Später wird die Korrektur von unwichtigen Fehlern (bugfreeze) verboten. Zum Schluß gibt es ein Verbot den Quellcode zu verändern (absoluter freeze), da dieser für jedes Programm auf seine Stabilität geprüft wird. Erst nach diesem Test wird der neue Release herausgegeben.

diese Verhaltensregeln massiv gebrochen wurde. Nachdem der Release veröffentlicht wurde, wird das Dateiablagesystem wieder für normale Veränderungen geöffnet.
Eine weitere Aufgabe des Releasekoordinators ist, die Programme aus dem Kern-Dateiablagesystems auf ihre Stabilität und Fehlerfreiheit zu testen. Die Entscheidungsbefugnisse des Releasemaintainers sind dabei auf die Aufnahme stabiler Programme beschränkt. Dabei gilt, daß der Releasekoordinator kein stabiles Programm ausschließen darf. Der Releasekoordinator entscheidet dann als einziger darüber, ob die Programme fehlerfrei und stabil sind, um in den Release aufgenommen zu werden. Er muß, bevor er ein Programm aus dem Release ausschließen darf, dies bei dem Koordinator und den Entwicklern ankündigen. Mit diesen muß der Releasemaintainer den zeitlichen Rahmen für die Verbesserung des Programms absprechen.
In dieser Funktion ist drittens der Releasekoordinator die zentrale Anlaufstelle für technische und soziale Probleme, die den Release gefährden könnten. So schlichtet er persönliche und fachliche Konflikte zwischen Programmierern, die die Fertigstellung eines Programms behindern und handelt mit den Kontrahenten das weitere Vorgehen aus, damit das jeweilige Programm bis zum Termin fertiggestellt ist.
Oft leiden Subprojekte an fehlenden Mitarbeitern, so daß wichtige Projekte nicht stabil genug sind. Den Subprojekten werden dann helfende Projektbeteiligte zugewiesen bzw. vermittelt. Manchmal hilft der Releasekoordinator auch selbst bei der Programmierung kurz aus.

Die Termine für diese verschiedenen Phasen des Releaseplans sind eher als Orientierungspunkte zu verstehen, die variieren können, da unvorhergesehene Probleme eine Verschiebung nötig machen. Es ist eher so, daß diese Phasen in Realität nicht so stark eingehalten werden. Wenn ein oder mehrere wichtige Programme noch überarbeitet werden müssen und nicht genug Zeit bis zum festgelegten Termin ist, kann der Releasetermin verschoben werden, wie beim letzten Release des Open Source-Projekts geschehen.
Außerdem ist der Releasemaintainer z.T. so überlastet, daß er erst auf Beschwerden der Übersetzer, Dringlichkeit oder schwereren Verletzungen der Regelungen reagiert. Bei kleineren Übertretungen der Regelungen werden je nach Charakters des Releasemaintainers und der verbleibenden Zeit bis zum Releasetermin diese u.U. toleriert. Bei schwereren Verletzungen der Regelungen wird ein Exempel statuiert, indem unerlaubte Änderungen vom Releasemaintainer rückgängig gemacht werden.

Die Entwickler arbeiten kontinuierlich an ihrem Programm, indem sie Fehler beseitigen neue Funktionen programmieren. Durch den Releaseplan werden ihre Tätigkeiten nicht gebremst, sondern auf bestimmte Aufgaben gelenkt. Bei den Übersetzern ist dies anders. Der Hauptaufwand ist immer kurz vor einem Release, da vorher die vielen Änderungen im Dateiablagesystem das regelmäßige Übersetzen zu einer Sysiphusarbeit macht[57]. So fangen die Übersetzer erst ca. zwei Monate vor einem Release richtig an zu arbeiten, um kurz vor dem Releasetermin fertig zu sein. Es kommt oft vor, daß Übersetzungsprojekte zum Release nicht alles komplett übersetzt haben. Aus diesem Grund gibt es, eine Bearbeitungshierarchie der Texte bei

[57] Das deutsche Übersetzungsteam übersetzt permanent die aktuellen Versionen im Dateiablagesystem, da sie genug Personen zur Mitarbeit haben.

den Übersetzern. Die Texte in den Programmen sollen als erstes übersetzt werden, dann folgen die FAQ, der Userguide und die allgemeinen Dokumentationen der Programme[58].
Damit der Releasekoordinator den Überblick behalten und Entscheidungen über die Aufnahme von Übersetzungen fällen kann, wird der Fortschritt einer Übersetzung/Dokumentation in einer Sprache in einer Arbeitsstatistiken automatisch erstellt. Die Statistik fungiert aber auch als Rückmeldung an die Übersetzungsteams. Dadurch wird der jeweilige Arbeitsaufwand, also auch die Reputation, des jeweiligen Ünbersetzungsteams angezeigt.

Paketierer
Nachdem die Programme getestet wurden, werden diese an die Paketierer weitergegeben. Die Paketierer packen den verstreuten Quellcode zu Paketen zusammen und compilieren den Quellcode in maschinenlesbaren Code für die verschiedenen Unix-/Linux-Derivate. Der stabile Release wird dann auf der Homepage für die verschiedenen Distributoren freigegeben. Paketierer sind über eine Mailingliste miteinander verbunden, um Schwierigkeiten beim Paketieren zu besprechen. Die Paketierer melden sich freiwillig oder werden von einer Firma bzw. einem Distributeur abgestellt. Die Auswahl ist beliebig, da die Paketierer als Hauptvoraussetzung schnelle Rechner für das compilieren benötigen.

5.2.3. Nutzung der Kommunikationsmedien

5.2.3.1. Mailinglisten/Email

Bei der Produktion ist die Kommunikation über Email und Mailinglisten das wichtigste Koordinationsinstrument. Die Softwareentwicklung muß an ständige Veränderungen in anderer Projekt-Software angepaßt werden, z.B. an wichtige Bibliotheken oder an Programme, mit denen das eigene Programm über Schnittstellen in Verbindungen steht. Dabei wird oft auch alter Quellcode aus der Software entfernt. Dies generiert einen ständigen Abstimmungsprozeß über Email zwischen den Subprojekten. So werden die Mailinglisten auch für die gegenseitige Hilfe verwendet, d.h. Fragen zur Programmierung und zur Fehlerbehebung werden auf den Mailinglisten gestellt und von anderen freiwillig beantwortet. Es können sehr viele Personen erreicht werden, die u.U. an der Diskussion zur Problemlösung teilnehmen. Dadurch wird Wissen kontinuierlich weitergegeben und wiederum durch den gemeinsamen Problemlöseprozeß generiert. Dies geschieht z.B. auf den Entwicklungs-Mailinglisten. Also funktionieren die Mailinglisten wegen des großen erreichbaren Beteiligtenkreises auch als Wissensreservoir.

Die Mailinglisten unterteilen sich in verschiedene Aufgabengebiete (s. Teil II, Kap. 1.5.3.), in denen die Aufgaben besprochen werden, die in einem Unterprojekt zu erledigen sind. Dort werden einerseits über das Fehlersystem, aber auch direkt, Fehler gemeldet. Andererseits sind diese Mailinglisten der Ort, an dem die Aufgaben verteilt werden und Diskussionen über strategische und operative Entscheidungen erfolgen. Persönliche Dinge werden auf den Mailinglisten meistens nicht diskutiert, sondern eher technische Problemstellungen.

[58] In der kurz nach dem großen Release folgenden Version x.x.1 werden dann die restlichen Übersetzungen und Fehlerkorrekturen eingefügt, die aus Zeitgründen abgelehnt wurden. Dies gilt auch für die Software.

Die meisten Konflikte werden in den Mailinglisten ausgetragen, da dies der wichtigste Kommunikationskanal im Projekt ist. Die Konflikte bestehen aus Emails, die aufeinander folgend ein Konfliktthema haben, sogenannte threads.
Es gibt verschiedene Konfliktarten, wie persönliche oder technisch-fachliche. Persönliche Konflikte sind selten, da das Medium Email keine sozialen Attribute transportiert und die Projekt-Mitarbeiter wenig über den anderen Wissen. Konflikte entstehen eher durch die Wortwahl, weswegen ein Projektbeteiligter dazu im Interview meinte: „Die Art und Weise der Wortwahl entscheidet über Sympathie und Antipathie, nach dem Sprichwort: So wie es in den Wald hineinschallt so schallt es wieder hinaus."

Dabei kann es zu rüden und beleidigenden Bemerkungen kommen, die aber nicht toleriert werden. Es bestimmen „Alltagskonflikte" sehr geringer Intensität die Kommunikation, die meistens zwei-drei Emails dauern. So können sogenannte flamewars, d.h. rüde Beschimpfungen, auf Mailinglisten auftreten. Doch sind solche flamewars[59] auf den Mailinglisten des Großprojekts relativ selten und wenn treten sie am ehesten auf der non-topic-Mailingliste[60] auf. Als ein möglicher Grund für die Abwesenheit solcher flamewars wurde genannt, daß zur Störung ein hoher Arbeitseinsatz erforderlich ist, was viele Störer davon abhält. Auf der non-topic-Mailingliste treten Konflikte über nationale Ressentiments und die politische Beurteilung anderer Länder über nationale Stereotype auf. Mehrere Entwickler sagten in den Interviews, daß sie diese Mailingliste aus diesen Gründen ablehnen[61]. Außerdem wurde von Interviewten gesagt, daß solche Aussagen die Reputation einer Person erniedrigen können, auch von Projektbeteiligten mit hoher Reputation.
Bei technisch-fachlichen Konflikten geht es einerseits um die Versprechungen und Leistungen, die ein Mitarbeiter erbracht hat, oder über die Arbeitsverteilung. Damit ist die mangelnde Übereinstimmung von Versprechen mit der Praxis und die mangelnde Einhaltung von Terminen verbunden. Bei der Arbeitsverteilung wird über die Zuständigkeiten und Entscheidungsgewalt bei der Programmierung diskutiert. Dabei muß auf den Wahrheitsgehalt der Emails bzw. der Fehlerberichte über Emails vertraut werden.

Andererseits gibt es öfters Diskussionen über die technische Ausführung von Programmen oder die Auswahl von eingesandtem Quellcode, doch bleiben diese meistens auf einem sachlichen Niveau und wachsen sich nur selten zu größeren Konflikten aus. Wenn einer einen Fehler macht, bekommt er eher entsprechende, z.T. sarkastische, Kommentare zu hören. Diese Kommentare laufen dabei über die Mailingliste, so daß ein Streit in der Öffentlichkeit der angemeldeten Mailinglisten-Emailempfänger geschieht. Aber Konflikte, die das Durchsetzen unangenehmer Entscheidungen oder hohe Kosten durch Überzeugungsarbeit hervorrufen, werden eher gemieden. Außerdem wurde während der Feldphase nichts bekannt, daß jemand wegen der Konflikte das Open Source-Projekt verlassen hätte.

---

[59] flame heißt soviel wie beleidigen, ausfallend sein

[60] non-topic Mailingliste = Mailingliste ohne spezielles Thema

[61] Es ist die Frage, ob dies eher ein Ventil oder eine Ausdrucksform unterschwellig vorhandener Vorurteile ist. Jedenfalls paßt dies nicht zum Selbstbild der internationalen, multikulturellen Open Source-Community.

5.2.3.2. Chatsystem

Ein weiteres Kommunikationsmedium, das IRC (s. Kap. 1.5.3.), wird selten verwendet. Das Chatsystem wird wegen der synchronen Kommunikation mit wenigen Teilnehmern eher für den persönlichen Kontakt („socializing“) benutzt, da es durch diese Charakteristika der gesprochenen Interaktion ähnelt. Es gibt auch Konflikte im IRC, da dort Reaktionen unmittelbarer und direkter erfolgen. Doch ist hier der Konflikt auf das IRC und die anwesenden Personen begrenzt, da im IRC nur wenige Projektbeteiligte zur gleichen Zeit anwesend sind und die Nachrichten nicht archiviert werden.
Manchmal wird dieses Medium für die Koordination von Arbeit und für die Diskussion bzw. Ideensammlung über das weitere Vorgehen in einem Subprojekt verwendet, indem im Chatsystem zeitgleiche Treffen veranstaltet werden. Nach diesem Treffen werden die Diskussion und Beschlüsse protokolliert und anderen Projektbeteiligten über die Mailingliste zur Verfügung gestellt. Diese Chat-Treffen ersetzen hier face-to-face-Konferenzen, da diese durch die hohen Reisekosten nicht möglich sind, wegen der großen räumlichen Trennung und der geringen Geldmittel der Beteiligten. Nachdem die groben Ziele abgesteckt wurden, werden to-do-Listen erstellt, von denen die Beteiligten freiwillig eine Aufgabe übernehmen. Eine Absprache über Kalender im Internet erfolgt nicht[62].

5.2.3.3. Persönliche Treffen

Treffen zwischen Projektbeteiligten sind selten, wenn sie nicht in räumlicher Nähe bzw. der gleichen Region wohnen. Diese räumlichen Bezüge läßt regionale Cluster mit Freundschaften entstehen. In diesen regionalen Clustern werden aber auch Absprachen über vorhandene und zukünftige Projekte diskutiert.
Seltener treffen sich die Projektbeteiligten zwischen den regionalen bzw. nationalen Clustern, wobei vorwiegend Personen aus dem inneren Kreis dort anzutreffen sind. Die Treffen, an denen die Projektbeteiligten zusammenkommen, sind Treffen kurz vor großen Releases oder bei bestimmten Linux-Konferenzen[63], die es in den verschiedenen Ländern gibt. Die persönlichen Treffen kurz vor großen Releases[64] werden wegen großer technischer Veränderungen über das Wochenende arrangiert und von den Linux-Distributoren mit der Übernahme der Anreise-, Verpflegungs- und Übernachtungskosten unterstützt[65]. Bei diesen Treffen ging es vor allem um den besseren Austausch und die bessere Koordination der ca. 30 bis 40 Projektbeteiligten bei der Fehlerbeseitigung und der endgültigen Fertigstellung des Releases. Dabei herrscht eine Arbeitsatmosphäre, u.U. wird rund um die Uhr programmiert. Aus diesem Grund werden der Releasekoordinator, die wichtigsten Open Source-Mitarbeiter mit hoher Reputation, Wis-

---

[62] Kalender benutzt keiner, da die Abstimmung normalerweise über allgemeine oder temporäre/ für das Event erstellte Mailinglisten läuft. Bisher gab es nur einen gescheiterten Versuch im Promoteam für die Vorbereitung von Ständen auf Konferenzen einen Kalender auf der Homepage zu benutzen.

[63] Bei vielen Konferenzen wird das Projekt mit seinem Produkt nur vorgestellt, so daß nur wenige Beteiligte anwesend sind.

[64] wie z.B. Versionen 1.0, 2.0 oder 3.0

[65] Die Distributoren unterstützen dies, da deren Verkauf von dem Open Source-Produkt abhängt.

sen und Problemlösekompetenz, aber auch Personen, deren Softwareprojekte Unterstützung benötigen, eingeladen.
Auf den Konferenzen steht neben der Repräsentation das Kennenlernen und der Austausch der Mitarbeiter im Vordergrund. Auf einer solchen Veranstaltung treffen sich oft die Projektbeteiligten persönlich das erste Mal und besprechen das weitere Vorgehen in ihrem und in anderen Softwareprojekten. Die neuen Ideen können dann in einem eigenen kleinen Softwareentwicklungsbereich mit Computern auf dem Stand programmiert werden[66]. Diese Besprechungen entstehen entweder zufällig oder werden manchmal von vorneherein geplant. Während des Konferenzgeschehens wird meistens gefachsimpelt und programmiert, aber auch abends wird über das Projekt gesprochen.
Sehr selten sind Urlaubsreisen oder anderen Gelegenheiten, bei denen sich die Projektbeteiligten treffen.

5.3. Kontrolle im Open Source-Projekt

Die Kontrolle in dem Open Source-Projekt ist keine Verhaltenskontrolle bei der eine räumliche Anwesenheit gegeben sein muß. Die Projektbeteiligten arbeiten zu Hause und räumlich getrennt und nur über Internet/Emails verbunden, weswegen eine direkte Verhaltenskontrolle nicht möglich ist. Vor kurzem wurde dem Open Source-Projekt ein Open Source-Werkzeug aus einem anderen Kleinprojekt überlassen, das Inkonsistenzen im Quellcode herausfinden kann. Dieses Werkzeug wird aber nur manchmal verwendet.
Es gibt zwei Kontrollarten. Die eine Kontrollart bezieht sich auf die Produktion, d.h. auf den Quellcode, die andere auf die Normen des Großprojekts. In diesem Unterkapitel wird die Kontrolle des Quellcodes, im nächsten die Kontrolle der Normen untersucht.

5.3.1. Kontrolle des Quellcodes als Qualitätskontrolle

5.3.1.1. Art der Kontrolle

Die Kontrolle des Quellcodes ist als eine Endkontrolle bzw. Kontrolle der Arbeitsleistungen zu bezeichnen, wobei sie auf die Qualität des Quellcodes fokussiert ist. Folglich wird einerseits unter qualitativ hochwertigen Code fehlerfreier Code verstanden, wobei das Verhalten der Person keinen Einfluß auf die Bewertung der Qualität des Codes hat. Diese Fehlerfreiheit gilt auch für die Übersetzer und Dokumentierer. Dazu gehört auch, daß sich die schlechte Qualität einer Fehlerkorrektur oder einer Codeveränderung dadurch feststellen läßt, daß mehr Arbeit hineingesteckt wird, um die durch die Korrektur verursachten Probleme zu lösen[67]. Andererseits gilt für hochwertigen Quellcode das Kriterium, daß er einfach und sofort verständlich sein soll.

---

[66] Auch räumlich nimmt dieser Programmier- und Austauschbereich ein Drittel der Gesamtstandfläche ein, was seine Wichtigkeit zeigt.

[67] Dies ist z.B. der Fall, wenn eine Fehlerkorrektur nur für einen Fall bewerkstelligt wurde und nicht für alle möglichen Fälle. Der Projektkoordinator, schreibt dann den Verursacher an und macht ihn höflich auf die gemachten Fehler aufmerksam, damit dieser den Fehler korrigiert.

Eine Grundlage für den hochwertigen Quellcode ist, daß minderwertiger Code und Sabotageakte, wie mutwilliges löschen oder verändern von Quellcode, durch das Zurückspielen älterer Programmversionen ausgeschaltet werden können. Durch diese Abwesenheit von starken Bedrohungen auf das Programmierte wird das Vertrauen zu anderen gestützt. Zusätzlich kann jeder an der Mitarbeit ausgeschlossen werden, da normalerweise jeder nur eine Dateiablage-Schreibberechtigung hat und identifizierbar ist. So sind Veränderungen und Fehler auf eine Person zurückführbar; es gibt keinen anonymen Dateiablage-Schreibzugang. Es kann also der Saboteur festgestellt und ausgeschlossen werden. Diese Sanktion ist das letzte Mittel der Community, um bösartiges Verhalten bei Zugriff auf den Quellcode zu bestrafen.

5.3.1.2. Qualitätskontrolle

Die Qualitätskontrolle erfolgt auf drei Ebenen. Die erste Ebene bezieht sich auf die interne, konkrete Softwareentwicklung, die zweite auf eine interne Kontrolle durch den Projektkoordinator und die dritte durch eine externe Fehlerkontrolle mittels Fehlerberichtssystem.
Die Basis der Kontrolle stellt die Selbstkontrolle der Entwickler dar, die freiwillig und selbständig programmieren und hoch motiviert sind. Einerseits gibt es eine Norm, nach der Entwickler hochwertigen Code schreiben sollen. Dies zeigt sich in den style guides, die Regeln der Programmierung festlegen, um einen gleichmäßig hohen Qualitätsstandard zu erhalten. Außerdem gehen die Entwickler davon aus, daß auch andere hochwertigen Quellcode erstellen, der keine schädigenden Auswirkungen wie Viren, etc. hat. Es gibt auch eine informelle Regelung, daß der nächste Release qualitativ besser als der vorhergehende sein soll. Andererseits können sie über das kontinuierliche Programmieren von hochwertigen Code Reputationszugewinne bekommen. Umgekehrt gilt auch, daß mit steigender Reputation das Vertrauen in die Fähigkeiten eines Projektbeteiligten steigt und sich damit das Vertrauen in die Hochwertigkeit des Quellcodes erhöht. Also werden Entwickler mit hoher Reputation weniger kontrolliert; es wird aber hochwertiger Quellcode erwartet.

Auf der zweiten Ebene gibt es eine stichprobenartige Kontrolle der Projektkoordinatoren. So hat der Projektmaintainer eines Subprojekts ein Auge auf die neuen Quellcodeveränderungen seines Subprojekts. Es werden hin und wieder auffällige Kommentare der Veränderungen überprüft, wobei die Kommentare die Veränderung beschreiben. Einige Entwickler mit hoher Reputation aus dem Subprojekt kontrollieren manchmal den Quellcode. Zusätzlich sind die Entwickler und Übersetzer/Dokumentierer zugleich Nutzer ihres Produkts, wobei Sie normalerweise die neuesten entwickelten Programme nutzen und so eigene und andere Fehler aufspüren. Diese Kontrolle der Projektkoordinatoren und ihrer Mitarbeiter unterbindet den mutwilligen Einbau von Fehlern oder Viren.
Bei den Übersetzern gibt es zusätzlich Korrekturleser, die das Übersetzte kontrollieren. Bei Übersetzern erleichtern ein elektronisches Wörterbuch und ein Werkzeug zur Überprüfung der Textveränderungen[68] die Kontrolle der übersetzten Texte.

[68] Mit dem Werkzeug werden die Textveränderungen, die durch die Entwickler verursacht werden, im Vergleich zu einem vorherigen, übersetzten Text aufgezeigt. Bei unübersetzten Texten werden diese von dem Werkzeug automatisch übersetzt. Außerdem werden diese Texte automatisch formatiert.

Die Entwickler und Übersetzer/Dokumentierer melden die Fehler entweder in dem sie direkt auf der entsprechenden Mailingliste eine Email senden oder das Fehlerberichtssystem benutzen.

Die dritte Ebene stellt eine Kontrolle über externe Endnutzer des Produkts dar. Ein Großteil der Fehlerberichte sind von externen Endnutzern, die nur einmalig einen Fehlerbericht abgeben. Wiederholte Fehlerberichte geben dafür einige Entwickler und Übersetzer/Dokumentierer ab. Die Programmierer lassen normalerweise den neuesten geschriebenen Quellcode laufen, um Fehler ihres eigenen und von anderen umgeschriebenen Quellcodes heraus zu bekommen, den sie dann über das Berichtssystem melden.
Das Fehlerberichtssystem ist dabei das wichtigste Instrument für die Qualitätskontrolle, über das ein sehr großer Teil der Fehlerberichte läuft. Bevor das Bugreportsystem eingeführt wurde, wurden die Fehler von der jeweiligen Anwendung aus an eine Mailingliste gesendet. Hier gab es das Problem, daß viele Nutzer ihren Ärger über den Fehler artikulierten und keinen nachvollziehbaren Fehlerreport ablieferten. Das Fehlermeldesystem hat die Qualität der Fehlerreports durch die Standardisierung und Disziplinierung der Meldungen verbessert. Außerdem sei nach Aussage der Interviewten der Aufwand einen Report zu schreiben hoch, so daß nur Motivierte einen Fehlerreport schreiben. Das Fehlerberichtssystem verbessert außerdem die Kommunikation zwischen Entwicklern und Fehlerberichterstattenden, was zu besseren Fehlererkennung und -beseitigungen[69] führt. Der Fehlerbericht wird trotz der Standardisierung durch den Projektkoordinator überprüft und nötigenfalls umgestuft. Falls ein Fehler nicht reproduzierbar ist, wird dieser ignoriert.
Insgesamt vereinfacht dieses Vorgehen die Fehlersuche und die Korrektur, was sich in der erhöhten Qualität des Programms niederschlägt.

5.3.2. Innerer Kreis und Kontrolle der Normen

Die Personen aus dem inneren Kreis überwachen die Einhaltung der Normen im Großprojekt. Die Projektbeteiligten aus dem inneren Kreis sind auch Projektkoordinatoren, die einerseits Normen auf der operativen Ebene, andererseits auf der strategischen Ebene überwachen. Hier wird auf die Normen eingegangen, die von den Projektbeteiligten mit hoher Reputation kontrolliert werden. Sie haben dabei die Einheit des Großprojekts im Blick. Die Vereinheitlichung wird z.T. durch Sanktionen durchgesetzt. Die Überwachung der Normen geht aber i.d.R. mit einer Diskussion über den Normenbruch einher. Dafür haben einige Personen aus dem inneren Kreis Zugriff auf community-erhaltende Funktionen und können Entscheidungen durchsetzen, in dem sie die Normenübertreter sanktionieren können, d.h. ausschließen (s. Kap. 2.3.2.). Damit haben diese wichtige Machtwerkzeuge in der Hand, die sie aber nur ein-

[69] Es wird versucht, die Fehlereinsender über das Fehlerberichtssystem in die Entwicklung einzubinden. So wird die Identität des Fehlereinsenders festgehalten, so daß der Fehlermelder per Email über die Berichtigung des Fehlers informiert werden kann. Außerdem kann der Berichtende über die Homepage nachprüfen, was mit seinem Fehler passiert ist. Der Fehlereinsender wird bei Schwierigkeiten und mangelnder Nachvollziehbarkeit mit dem Bugreport von einem Entwickler nach Details gefragt und es wird gemeinsam versucht das Problem zu lösen.

setzen, wenn sich in einer Diskussion eine Entscheidung abzeichnet. Damit ist der innere Kreis auch eine machtausübende Instanz.

Es gibt in dem Großprojekt Normen bei Produktion, Kommunikation und Nutzung, die von dem inneren Kreis überwacht werden. Diese drei Bereiche werden durch projekt-interne und -externe Einflüsse bestimmt. Die Produktion des gemeinsamen Gutes wird vor allem durch projekt-interne Einflüsse beeinflußt, da die Produktion vor allem in dem Großprojekt stattfindet. Die Kommunikation wird durch projekt-interne und –externe Personen beeinflußt, die über die Mailinglisten ohne eingeschränkten Schreibzugriff Emails versenden. Die Nutzung des gemeinsamen Gutes wird vor allem durch projekt-externe Einflüsse bestimmt.

5.3.2.1. Kontrolle der Produktion

Bei der Produktion spielt einerseits die Norm um die Lizenz eine Rolle, daß der im Versionsmanagement veröffentlichte Quellcode der Öffentlichkeit, vertreten durch die Community des Großprojekts, gehört (s. Kap. 2.3.2.). In diesem Fall überwachen die Personen aus dem inneren Kreis, daß Quellcode nicht beliebig gelöscht wird und damit eine einheitliche Softwareentwicklung möglich ist. Der innere Kreis hat Zugriff auf die projekterhaltenden Posten, die auch Sanktionsmöglichkeiten enthalten. Besetzt wurden die projekterhaltenden Posten von Personen des inneren Kreises als diese Posten geschaffen wurden, wie bei den Projektgründern als Projektleiter.
Andererseits wirkt die Norm der Einheit des Projekts direkt, die im Gegensatz zur Gabelung der Softwareentwicklung steht. Es soll eine Software zu ihrem technischen Optimum ausgebaut werden. So hat ein Entwickler mit Reputation eine Diskussion über die Duplikation von Applikationen angestoßen, d.h. der Mehrfachentwicklung oder Diversifizierung von gleichen bzw. sehr ähnlichen Programmen mit gleichen bzw. sehr ähnlichen Funktionen. Die Programmierer der betreffenden Applikationsduplikationen wurden von den Personen mit Reputation angemahnt, sich auf ein gemeinsames Programm zu einigen, das in den Release aufgenommen werden soll[70]. Diese Applikationsduplikationen konnten im Laufe der Projektentwicklung entstehen, da durch die Freiwilligkeit der Programmierung, keiner dem anderen vorschreiben mochte, was dieser Programmieren soll[71]. Weiter darf ein funktionsfähiges und stabiles Programm nicht aus dem Release ausgeschlossen werden. Dies wird verstärkt durch die Programmierer, die ohne voneinander zu wissen, an ähnlichen Programmen arbeiten.

---

[70] Eine Lösung ist der modulare Aufbau eines gemeinsamen Programms, so daß diese Software durch plug-ins den jeweiligen Wünschen angepaßt werden kann. Der Konflikt wurde aber gelöst, indem die Programme aus dem Release herausgenommen und zum offenen Download auf der Homepage frei gegeben wurden. Endgültig wird das Problem wahrscheinlich durch eine Entscheidung der Nutzer gelöst, was sich z.B. an den Downloadzahlen messen läßt. Ein Funktionsträger mit Versionsmanagementsystemzugriff hat dann die Programme im Dateiablagesystem in einen speziellen halboffiziellen Bereich verschoben.

[71] Die Programme wurden parallel in einem speziellen Bereich des Dateiablagesystems entwickelt oder aus der Umgebung des Open Source-Projekts aufgenommen. Eine Entscheidung wurde auch aus dem Grund unterlassen, da diese Programme unterschiedliche Vorzüge und Nachteile hatten und es kein überzeugendes Spitzenprodukt für eine eindeutige Auswahl gab.

Es stehen sich zwei konfligierende Normen gegenüber. Die eine Norm betont das eigenständige, freiwillige und selbstbestimmte Arbeiten, die andere die Einheitlichkeit und Verbesserung der Software für die Nutzer. Die Norm der Einheitlichkeit und Verbesserung der Software bezieht sich dabei auf das Ideal, daß das (technisch) beste Programm geschrieben werden soll. Diese Norm verlangt die stete Verbesserung der Programme zu ihrem technischen Optimum, was sich nur durch die gemeinsame Arbeit an einem Programm realisieren läßt, weswegen Gabelungen[72] bzw. Duplikationen von Software abgelehnt werden.
Dem steht die Ausdifferenzierung und somit das Verschwimmen der Grenzen der Community gegenüber[73]. Das bedeutet, daß sich Diversifizierung und Konsens/Einheit bzw. Eigeninteresse und Allgemeinwohl gegenüberstehen.

5.3.2.2. Kontrolle der Kommunikation

Bei der Kontrolle der Kommunikation geht es darum, daß einheitlich der Norm der kooperativen Kommunikation gefolgt wird. Dabei ist zwischen projekt-internen und –externen Personen zu unterscheiden.
Bei projekt-internen Personen die Beleidigungen und vulgäre Ausdrücke wiederholt verwenden, werden direkt in einer Entgegnungs-Email darauf angesprochen und aufgefordert eine Wiederholung zu unterlassen (s. Anhang B).

Die Störungen von projekt-externen Personen gehen von Provokateuren aus, sogenannten Trollen[74]. Die Trolle versuchen durch provokative Aussagen auf einer Projekt-Mailingliste, die z.T. persönlich und beleidigend sind, eine Reaktion von einem oder mehreren Projektbeteiligten zu bekommen. Damit verstoßen diese gegen die Norm der sachlichen Auseinandersetzung. Diese Störenfriede werden im Allgemeinen ignoriert, da durch die Blockade von deren Emailadressen weitere Konflikte heraufbeschworen werden würden. Die Trolle verschwinden meistens schnell wieder, da sie keine Reaktion bekommen. Außerdem gab ein Interviewter an, daß es die Trolle bei dem Großprojekt ziemlich selten gibt. Als ein Grund wurde die dezentrale Struktur des Open Source-Projekts genannt, da es keine Identifikationsfigur gibt, die direkt angegriffen werden könnte, und andere Personen eher unbekannt sind.

---

[72] Eine Gabelung von Software bedeutet, daß sich die Entwicklergruppe aufteilt und ausgehend von einer gemeinsamen Softwarebasis verschiedene Entwicklungswege geht und somit zwei neue ähnliche Programme mit ähnlichen Funktionen schafft.

[73] Die innere Grenzziehung der Zugehörigkeit verschwimmt durch das parallele Arbeiten an ähnlichen Programmen im Dateiablagesystem. Durch die Diversifizierung und Duplizierung folgt direkt die Frage, welche Programme zum Release des Open Source-Projekts gehören, der nur einen begrenzten Umfang haben kann. Die Grenze zwischen dem Projekt und seinem Umfeld der externen Programme ist dagegen durch die technische Hürde des Dateiablagesystemzugriffs schärfer gezogen. Doch existiert eine Bindung zwischen den beiden Bereichen, da diese Programme auf das Großprojekt aufbauen, also davon abhängig sind, und im Gegenzug das Projekt von der Fülle der Zusatzprogramme/-projekte profitiert. Dieses Umfeld garantiert den Nachschub an wichtigen Programmen für den weiteren Ausbau des Großprojekts.

[74] Nach Aussagen eines Interviewten sind Trolle meistens jüngere Personen, d.h. Teenager.

5.3.2.3. Kontrolle der Nutzung des gemeinsamen Produkts

Die Kontrolle der Quellcode-Nutzung bezieht sich auf eine einheitliche, lizenz-bezogene Verwendung. In diesem Fall untersteht der meiste Quellcode der general public licence, weswegen auf die Rückgabe von verändertem Quellcode und die Anwesenheit von copyrights und dem Open Source-Projekt-Logos geachtet wird (s. Kap. 1.6.2.). Die Nutznießer der Verwendung sind projekt-externe Personen und projekt-externe Unternehmen gleichermaßen. Dabei wird ein besonderes Augenmerk auf Unternehmen gelegt, da diese am meisten von dem Quellcode profitieren können. Die Unternehmen werden deswegen kontrolliert, ob sie sich an die Normen der allgemeinen Open Source-Community halten. Dies erfolgt vor allem von den Personen des inneren Kreises, die oft auf Linux-Kongressen sind und dort Übertretungen aufdecken. Das weitere Vorgehen bei solchen Normübertretungen wird dann in der geschlossenen Mailingliste des inneren Kreises besprochen. Auf der anderen Seite unterstützen diese Nutznießer, vor allem die Distributoren, z.T. das Projekt mit Spenden. Aus diesem Grund ist der innere Kreis über den eingetragenen Verein für die Kontakte zu den Sponsoren zuständig und macht auch Werbung für das Projekt[75].

5.4. Motivation oder monetäre und nicht-monetäre Gratifikation

5.4.1. Individuell-orientierte und community-orientierte Motivationsgründe

Es bestehen verschiedene Motivationsgründe, warum Personen aus dem IT-Bereich unentgeltlich Software erstellen. Aus der qualitativen Experteninterviews ergab sich, daß es verschiedene Motivationsgründe für eine Beteiligung gibt. Dies wurde von den Selbstauskünften in den Selbstinterviews gestützt (Anhang A). Diese Gründe können in zwei Bereiche aufgeteilt werden. Der eine Bereich bezieht sich auf das Individuum, der andere auf die Community bzw. auf das Projekt.

Ein Motivationsgrund, der auf das Individuum bezogen ist, ist erstens das Verbessern störender Funktionen der Software, da die Entwickler auch Nutzer des Desktops sind. Ein zweiter Grund bezieht sich auf das Programmieren an sich. Einerseits bekommt der Entwickler bei der Programmierung ständig Erfolgserlebnisse, was ihn zum weiteren Programmieren anregt. Andererseits macht der Prozeß des Programmierens selbst Spaß, da der Entwickler dabei kreativ und erfinderisch tätig ist sowie intellektuell stimuliert wird. Dazu gehört ein spielerischer Umgang mit der Technik bzw. der Software, wurde von einigen Interviewpartnern ausgesagt. Das Programmierte wird als ein Spielzeug[76] gesehen, was sich auch als Bastelleidenschaft beschreiben läßt. Das Programmieren wird von vielen als Hobby gesehen, kann aber auch zu einer „Sucht“, d.h. Realitätsflucht[77], werden. Hier ist ein Zusammenhang zwischen der faszinierenden Tätigkeit des Programmierens und der Kultur bzw. Profession der Arbeitenden festzustellen, denn die überwältigende Mehrheit der Programmierer hat studiert oder studiert noch technisch-ingenieurwissenschaftliche bzw. Informatikstudiengänge.

---

[75] Es gibt ein eigenes Subprojekt, das Gelder und ideelle Unterstützung von Sponsoren organisiert.

[76] Dies kann man an den lustigen Fehlermeldungen sehen, die von einem Interviewten erwähnt wurden.

[77] So erzählte ein Interviewter, da diese Erfolgserlebnisse negative Erlebnisse, wie die Trennung vom Partner, auffangen können.

Drittens kann der Erwerb von sozialen oder technischen Kompetenzen zur Erhöhung späterer Arbeitplatzmöglichkeiten ein Motivationsgrund sein. Dabei unterstützt die Reputation die Arbeitsmarktchancen. Projektbeteiligte, die Releasekoordinatoren waren oder die besonders anerkannt sind, werden von projektnahen Firmen gern angeworben. Doch ist dieser Motivationsgrund laut Aussage der Interviewpartner eher unwichtig.
Als vierter Grund kann die Bezahlung für die Arbeit an dem Großprojekt genannt werden. Dabei haben die angestellten Open Source-Mitarbeiter ihr Hobby zum Beruf gemacht, wobei sie oftmals uneingeschränkt über die Arbeit an dem Großprojekt entscheiden können. Aber monetäres Einkommen stellt bei den dafür angestellten Open Source-Mitarbeitern keinen wichtigen Motivationsgrund dar.

Motivationsgründe, die auf das Großprojekt und seine Community bezogen sind, spielen aber auch eine Rolle. So ist erstens die Hilfsbereitschaft unter den Projektbeteiligten bei der Softwareentwicklung ein Grund, die in der kooperativen Kultur in Erscheinung tritt. Dabei motiviert die Genugtuung jemandem helfen zu können, genauso wie geholfen zu bekommen.
Zweitens wurde die produktive Teamarbeit genannt, die eng mit der kooperativen Kultur zusammenhängt. Dazu gehören auch Absprachen mit anderen Mitarbeitern. Es wird z.B. abgesprochen, daß einer die Erstellung einer neuen Funktionen verspricht, wenn der andere bestimmten Code schreibt bzw. Veränderungen durchführt.
Das Kennenlernen von Gleichgesinnten und die Identifikation mit der Community ist der dritte Motivationsgrund. Dabei befördert das persönliche Kennenlernen die weitere Identifikation mit der Community. Dabei vermindert die Identifikation mit dem Projekt die Häufigkeit der Konflikte. So sind Konflikte in und zwischen verschiedenen Softwareunterprojekten in dem Open Source-Großprojekt selten. Dies gilt auch für die Übersetzungsteams.
Viertens stellt die Erlangung von Reputation, d.h. von sozialer Anerkennung, ein Motivationsgrund dar. So gibt es Konflikte über die Qualität einer Programmierung, die auch motivierenden Charakter haben können, wenn ein zurechtgewiesener Kritiker den anderen zeigen möchte, daß er besser ist und ein Problem in kurzer Zeit besser lösen kann.
Fünftens fühlen sich einige der Community verpflichtet und möchten dieser etwas zurückgeben, da sie die Software kostenlos von dieser erhalten haben.
Sechstens sind Langeweile und die Sehnsucht nach Gesellschaft weitere Motivationsgründe.
Ein siebter Grund stellen Sachspenden oder monetäre Spenden dar, die aber von den Interviewten als unwichtig bezeichnet wurden. Dabei handelt es sich um die Wertschätzung der Arbeit durch Außenstehende. So werden durch Spenden von Endnutzern an Projektbeteiligte deren Motivation erhöht. Ein Entwickler schrieb: “I did get a gift certificate from a thankful end-user a while back which was cool. It made my month” (s. Anhang A). Doch es wurde berichtet, daß solche Spenden selten sind. Auch Spenden von Firmen an das Großprojekt erhöhen die allgemeine Motivation in der Community.

5.4.2. Bewertung der Motivationsgründe

Eine erste Annäherung an die Wichtigkeit der Motive läßt die Auswertung der Ergebnisse der offenen Frage (Fragenr. 20) der quantitativen Befragung aus Anhang C zu, die auf einem Linux-Konferenz an anwesende Community-Mitglieder ausgeteilt wurde. Als Zentral wurden die Community des Open Source-Projekts mit kooperativen Strukturen ohne zentrale Leitfi-

gur(en) und die technische Qualität des Produkts genannt. Als weitere Motivationen wurden noch der Spaß am Programmieren an sich, die selbstbestimmte Arbeit, die Konkurrenz zur proprietären Software und die Nutzerorientierung erwähnt. Dies ist aber eher als eine sehr grobe Annäherung zu sehen.
Die Wichtigkeit der Motive wurden in einem Fragebogen abgefragt. Aus den qualitativen Experteninterviews und einigen Angaben aus der Literatur wurde eine Itembatterie zur Motivation erstellt, die im Durchschnitt von 53 Personen beantwortet wurde (s. Anhang C). Dabei zeichnete sich ab, daß einige Motivationsgründe für die Beteiligung als besonders wichtig und einige als besonders unwichtig empfunden wurden.
Über 85 % der Antwortenden nannten die Freude am Ergebnis, der Spaß am Programmieren und Intellektuelles Interesse (Probleme lösen) von einer großen Mehrzahl als wichtig. Aus diesem Grund steht der Spaß am intellektuellen Rätsel lösen, vergleichbar dem Basteln, im Zentrum der Motivationsgründe.
Ca. 80 % der Antwortenden sehen die Qualifizierung/Weiterqualifizierung als wichtig an. Die Qualifizierung/Weiterqualifizierung wird dabei mehr im Kontext des Spaßes an der Produktion als in Beziehung zu Verwertungsinteressen gesehen, da nur für ca. 48 % der Antwortenden die Verbesserung der Arbeitsmarktchancen wichtig sind.
Zwischen 70 % und 80 % der Antwortenden fühlen sich der Community verpflichtet und wollen dieser etwas zurückgeben, fühlen sich der Community zugehörig, möchten ein störendes Problem für die private Anwendung lösen und erfreuen sich der Freiheit der Arbeitsteilung, also an dem selbstbestimmten Arbeiten. Das bedeutet, daß an zweiter Stelle einerseits Motivationsgründe mit Bezug zur Community, andererseits mit Bezug zum Arbeits- und Nutzungsprozeß stehen.
66 % bzw. 68 % empfinden Selbstbestätigung und politisch-gesellschaftlichen Idealismus als wichtige Motivationsgründe.
Bei der Lizenzierung des Open Source-Projekts, der Ablehnung von Microsoft, Ablehnung von kommerzieller Software und der Verbesserung der Arbeitsmarktchancen ergab sich keine Mehrheit, die diese Gründe für wichtig oder unwichtig hielten. Aus diesem Grund haben diese Motivationsgründe eine gewisse Relevanz, wirken aber im Hintergrund.

Als unwichtig wurden die erhöhte Aufmerksamkeit der Community, ein störendes Problem für die Arbeit im Beruf lösen, die erhöhte Aufmerksamkeit der Open Source-Community außerhalb des Großprojekts und der erhöhte Status im Beruf von ca. 60 % der Personen gesehen. Die Motivationsgründe Langeweile, viel Freizeit und Hoffnung auf spätere Entlohnung spielen bei vielen keine größere Rolle.
Sachspenden und Geld/Entlohnung wurden in der überwältigenden Mehrheit als unwichtig erachtet.

5.4.3. Wichtigkeit der Motivationsgründe bei Eintritt und nach Eintritt in das Großprojekt

Die Motivationsgründe verändern sich nach der Aussage der Interviewten. Dabei ist die Wichtigkeit der Motivationsgründe beim Eintritt von denen nach dem Eintritt zu unterscheiden.
Beim Eintritt sind die meisten Nutzer, die in das Projekt eintreten, angetrieben von einem Problem mit der Software. Es fehlt ihnen eine spezielle Funktion oder ein spezielles Pro-

gramm oder es stört sie ein spezifischer Fehler oder eine Eigenheit eines Programms, so daß sie anfangen selbst an dem Problem zu arbeiten. Die neuen Teilnehmer wenden sich dabei den Tätigkeitsbereiche zu, in denen sie Vorkenntnisse haben. Das Lösen eines eigenen Problems mit der Software wurde als der wichtigste Eintrittsgrund genannt.
Es gibt noch andere Gründe, die aber nicht so wichtig sind, wie z.B. die Langeweile und die Sehnsucht nach Gesellschaft die von einem einzelnen Interviewten erwähnt wurde.
Ein weiterer periphererer Grund ist, der community etwas für die kostenlose Nutzung der Software zurückzugeben. So schrieb ein Programmierer: „I was so impressed with the work of such a high quality that others did for nothing, that I decided to respond with same kind of work, both for XXX and Linux" (s. Anhang A).

Nach dem Eintritt werden andere Motivationsgründe wichtiger. Der Konflikt um die Applikationsduplikationen (s. Kap. 4.2.1.) zeigt, daß in dem Großprojekt eine Spannung zwischen Eigeninteresse und Gemeinschaftsgefühl besteht. Dabei wird die Community mit ihrem Gemeinschaftsgefühl wichtiger, sagte ein Interviewter. Die Entwickler mit hoher Reputation identifizieren sich stärker mit der Community als Projektbeteiligte aus der Peripherie. Dabei ziehen die Personen aus dem inneren Kreis einen großen Teil ihrer Motivation aus der Zugehörigkeit zur Community, weswegen sie das Allgemeinwohl der Community als Ganzes vertreten.
Die Interviewten berichteten von einer Entwicklung des Großprojekts, die eine Verschiebung der Wichtigkeit von Motivationsgründen nach sich ziehen kann. Am Anfang der Erstellung des Desktops gab es einige wenige Basisprogramme, an dem alle arbeiteten. Diese wenigen Programme wurden auf ein gemeinsames Ziel hin bearbeitet, was das Gemeinschaftsgefühl erhöhte. Mittlerweile hat sich das Großprojekt weiter entwickelt, die zentralen Programme sind zum größten Teil fertiggestellt worden. Es gibt eine Konzentration auf die Entwicklung von nicht-zentralen Anwendungen und einen immensen Zuwachs an Mitentwicklern. Dies wirkt sich in einer Diversifizierung und Duplizierung der Programme aus, was sich wiederum in einer Diversifizierung der Ziele und einer Aufsplitterung in Einzelinteressen niederschlägt. Dies deutet auf eine Stärkung des Eigeninteresses als Motivationsgrund hin.

5.4.4. Demotivierende Faktoren

Es gibt demotivierende Faktoren bei Eintritt und nach Eintritt in das Großprojekt. Die Konflikte auf den Mailinglisten der Community spielen dabei aber eine geringe Rolle als demotivierende Faktoren.

Beim Eintritt in das Großprojekt kann die manchmal rüde bzw. ablehnende Kommunikation der Entwickler die Motivation der Nutzer für einen Eintritt senken. Dies ist Ausdruck einer Spannung zwischen Nutzern, die sich nicht an dem Projekt beteiligen, und den Entwicklern. Demgegenüber werden Nutzer, die sich beteiligen wollen und dies signalisieren, eher ernst genommen (s. Kap. 1.6.3.).
Der Einstieg in ein Unterprojekt läuft i.d.R. über patches. Doch können zwei patches zur gleichen Zeit zu einem Problem erstellt werden, weswegen ein patch ausgewählt wird. Es wurde von einem Projektbeteiligten berichtet, daß sein patch mit Begründung abgelehnt wurde. Mit

der Begründung konnte er wenig anfangen. Die Begründung und die Ablehnung verringerte seine Motivation, so daß er kein weiteres Interesse an einer Mitarbeit hatte.

Nach dem Eintritt wirken verschiedene demotivierende Faktoren. Demotivierend wirken sich z.B. negative Bewertungen des Quellcodes oder des Arbeitseinsatzes aus.
Bei der gemeinsamen Arbeit und Kommunikation in den Subprojekten lernen sich die Projektbeteiligten kennen, wobei sich Sympathien und Antipathien zwischen den Mitarbeitern entwickeln können. Bei Antipathien zwischen zwei Mitarbeitern wird einer der beiden, falls eine Schlichtung keinen Erfolg hatte, auf kurz oder lang sein Betätigungsfeld in dem Großprojekt in ein anderes Subprojekt verlegen, aber nicht das Großprojekt verlassen.

6. Resümee

Bei diesem Open Source-Großprojekt handelt es sich um eine weltweit verteilte Gruppe von ca. 800-1000 Menschen, die vor allem aus Europa, besonders Deutschland, kommen. Die Gruppe ist recht homogen, da der größte Teil der Personen männlich, eine IT-Ausbildung bzw. –Studium haben und zwischen 20 und 30 Jahren alt ist. Ein großer Anteil der Open Source-Projektbeteiligten sind Studenten, ein weiterer Erwerbstätige. Es wird freiwillig, als Hobby, an der gemeinsamen Erstellung eines Desktops gearbeitet, das vor allem für Linux und Unix-Derivate verwendet wird. Aus diesem Grund überwiegen informelle Normen, wobei die monetäre Gratifikation nicht im Mittelpunkt steht. Zusätzlich sind die Projektbeteiligten zugleich Leistungsersteller wie Nutzer
Der Support für Fehler erfolgt über ein automatisches Fehlerberichtssystem, für die Implementierung in Organisationen, wie Unternehmen, sind dagegen spezielle open source-ausgegründete Unternehmen zuständig.
Es gibt verschiedene Tätigkeitsbereiche in der Open Source-Community, wovon die Softwareentwickler und die Dokumentierer/ Übersetzer, die wichtigsten sind. Die wichtigsten Werkzeuge stellen die Mailinglisten für die Kommunikation und das Dateiablagesystem für die Produktion dar, die von den Projektbeteiligten selbst erstellt wurden und z.T. von projektnahen Unternehmen unterstützt werden. Das Open Source-Projekt besteht einerseits aus wenigen wichtigen und zentralen Unterprojekten, andererseits aus vielen kleinen Ein-Personen-Subprojekten.

Es gibt Konflikte zwischen der Umwelt des Großprojekts und dem Großprojekt selbst. Die Umwelt des Großprojekts besteht einerseits aus anderen Open Source-Projekten, Unternehmen und Endanwendern. Konflikte zwischen verschiedenen Open Source-Projekten sind selten, aber wenn sie auftreten, geht es um die Grundlagen der allgemeinen Open Source-Community, wie dem konkreten Umgang mit den Lizenzen. Die guten Beziehungen zu anderen Open Source-Projekten werden durch die Nutzung von Werkzeugen dieser Projekte im Großprojekt untermauert. Weit mehr Konflikte um die Anwendung der Lizenz des Großprojekts gibt es zwischen dem Großprojekt und Unternehmen, wobei sich die Konfliktthemen um die Übernahme, Veränderung und Verkauf von Quellcode sowie um die Reputation durch copyright-Vermerke drehen.

Der Eintritt in die Community erfolgt nach dem Signalisieren von Interesse an der Mitarbeit durch Zusenden von kleinen Quellcodeveränderungen. Nach mehrmaligem Zusenden von qualitativ-hochwertigen Quellcode erhalten die neuen Teilnehmer die Schreibberechtigung für das Dateiablagesystem. Das bedeutet, daß die neuen Teilnehmer mindestens Basiswissen über Programmieren und Programmiersprachen besitzen müssen. Der Eintritt ist dabei als ein Selbstselektionsprozeß zu sehen, wobei die Selbstmotivation eine große Rolle spielt. Nach dem Eintritt besteht die Möglichkeit durch Reputation in den inneren Kreis aufzusteigen. Der innere Kreis ist eine schwer abgrenzbare Gruppe von Personen, die sich um das Projekt verdient gemacht haben und deswegen Reputation bzw. sozialen Status erlangt haben. Die Reputation besteht aus verschiedenen Elementen, wie Seniorität/ Erfahrung, kontinuierliche Leistung, freundlichem/kooperativen Umgang und Sichtbarkeit. Freiwillige Austritte erfolgen durch die Hinwendung zu anderen Aufgaben, wie Familie oder Erwerbsarbeit, Zwangsaustritte durch das Übertreten von wichtigen Normen, wie z.B. das unabgesprochene Löschen von Quellcode.

Die Produkterstellung bzw. Leistungserstellung steht in dem Open Source-Projekt im Mittelpunkt. Die Softwareentwicklung erfolgt normalerweise in einem Versuch und Irrtums-Prozeß, wobei derjenige über seine Arbeit entscheidet, der die Arbeit macht. Damit fällt die Leistungserstellung und die Arbeitsverteilung zusammen. Dabei gibt es eine operative und eine strategische Entscheidungsebene, wobei sich die operative um die konkrete Softwareerstellung und die strategische um rechtliche Probleme und visionäre Ziele dreht. Die Arbeitsaufteilung erfolgt über die freiwillige Übernahme von Arbeiten, wobei ein Projektkoordinator mit Mitarbeitern das eigene Projekt überwacht. Die Arbeitenden haben normalerweise ein Hauptprojekt, in welchem sie ständig beteiligt sind. Doch helfen sie häufig auch bei anderen Projekten aus. Bei der Arbeitsverteilung werden Absprachen zwischen den Subprojektmitarbeitern getroffen, wobei Vertrauen zur Einhaltung der Absprache eine wichtige Rolle bei dem Open Source-Projekt spielt. Der Projektkoordinator ist für die Fehlerlosigkeit seines Projekts verantwortlich. Vom Projektkoordinator wird z.T. die weitere Bearbeitung seines Projekts erwartet. Entscheidungen über die Übernahme von Quellcode oder die weitere Entwicklung des Projekts werden im Konsens gefällt. Der Projektkoordinator hat dabei keine besonderen Machtbefugnisse. Nur Projektbeteiligte aus dem inneren Kreis genießen wegen ihrer hohen Reputation besonderen Einfluß und Vertrauen. Bei Konflikten wird von diesen eine Schlichtung erwartet.
Die gemeinsame Arbeit wird in einem Release zusammengeführt. Es gibt eine offizielle Veröffentlichung, den Release, der nach einem koordinierenden Releaseplan herausgegeben wird. Zuständig für die Kontrolle dieses Plans und für die Übernahme von stabilen Programmen ist der Releasekoordinator, der diesen speziellen, koordinierenden Posten für den inneren Kreis übernimmt.

Es kristallisieren sich drei Kanäle für Störquellen heraus. Erstens gibt es Konflikte bei der Kommunikation, d.h. auf den Mailinglisten, aber auch beim Chatsystem. Hier sind die meisten Störungen zu finden. Störungen von Projektbeteiligten werden verbal geahndet, projektexterne Störungen werden üblicherweise ignoriert. Zweitens gibt es bei der Nutzung von Quellcode hin und wieder Konflikte und Probleme mit Trittbrettfahrern, die sich um die GPL-

Lizenz drehen. Dies wird normalerweise bei Firmen durch schlechte Publicity geahndet. Drittens kommen Störungen bei der Leistungserstellung im Dateiablagesystem selten vor. Dies wird durch Zurückspielen des vorherigen Quellcodes und u.U. Sperrung des Zugangs sanktioniert.

Die Kontrolle der Arbeit im Open Source-Projekt läuft über die Kontrolle der Qualität des Quellcodes. Dies erfolgt auf der operativen Ebene. Die Qualität wird durch Selbstkontrolle durch gleichzeitige Nutzung der Software, stichprobenartigen Kontrollen durch Projektkoordinatoren und den Einbezug der Endnutzer über ein Fehlerberichtsystem erhöht. Es besteht dabei ein Konflikt zwischen der selbstbestimmten Arbeitsweise und der Endnutzerorientierung der Community, was sich in kleineren Streitigkeiten entlädt. Auf der strategischen Ebene kontrolliert der innere Kreis die Einhaltung von Normen bei der Produktion, Kommunikation und Nutzung der gemeinsamen Software. Dabei ist der innere Kreis für die Einheit des Großprojekts bzw. für das einheitliche Auftreten zuständig.

Die Motivation der Projektbeteiligten setzt sich aus einer Mischung intrinsischer und extrinsischer Elemente zusammen. Intrinsische Motivationen sind eigene Probleme zu lösen oder der Spaß am Programmieren. Diese Motivationsgründe wurden von einer großen Mehrheit als wichtig bezeichnet. Die extrinsische Motivation ist auf die Community und ihrer kooperativen Kultur bezogen. Dazu zählen Anerkennung und gegenseitige Hilfe. Diese Motivationsgründe werden auch von einer Mehrheit für wichtig erachtet. Konflikte können dabei einerseits die Motivation im Projekt durch Beleidigungen senken, aber andererseits durch das wecken von Ehrgeiz erhöhen.

7. Literatur

- Jörgensen, N. (2001): Putting it all in the trunk: incremental software development in the FreeBSD open source project. In: Info Systems J (2001) **11**, No. 4, 321–336
- Mockus, A., Fielding, R., Herbsleb, J. (2002): Two Case Studies of Open Source Software Development: Apache and Mozilla. In: ACM Trans. Software Engineering and Methodology, 11(3), 309-346, 2002, URL: http://www.research.avayalabs.com/techreport/ALR-2002-003-paper.pdf