Universität Rostock Traditio et Innovatio

Lernen in informellen, internetbasierten Netzwerken

Einflussfaktoren auf den Aufbau von Kompetenzen in Entwicklungsgemeinschaften der Free / Open-Source Bewegung

Dissertation

zur Erlangung des akademischen Grades

Doctor philosophiae (Dr. phil.)

an der Philosophischen Fakultät der

Universität Rostock

vorgelegt von

Klaus Muttray

aus Rostock

23. Mai 2021

https://doi.org/10.18453/rosdok_id00003760

Gutachter:innen:

1. Prof. i. R. Dr. Wolfgang Nieke, Universität Rostock, Institut für Allgemeine Pädagogik und Sozialpädagogik
2. Prof. Dr.-Ing. Alke Martens, Universität Rostock, Institut für Informatik
3. Prof. Dr. Georg Cleppien, Universität Augsburg, Arbeitsbereich Pädagogik der Kindheit und Jugend

Jahr der Einreichung: 2021
Jahr der Verteidigung: 2022

Inhaltsverzeichnis

1 Einleitung

Die Verfügbarkeit von immer günstigeren Computern und stationären beziehungsweise mobilen Breitbandinternetzugängen führt zur einer Durchdringung des Alltags mit digitalen Konsum- und Interaktionsmöglichkeiten. Dies bezieht sich auf verschiedene Bereiche des täglichen Lebens: Lohnarbeit, Freizeitgestaltung oder die Informationsrecherche im Internet zur Lösung von Alltagsproblemen, wie zum Beispiel bei Gesundheitsthemen, Vergleich von Konsumartikeln oder Kochrezepten. Ebenso bei der Verständigung zwischen Familienangehörigen, Freund:innen oder unter Kolleg:innen verdrängen neue Kommunikationskanäle bisher etablierte analoge Formen der Kommunikation: Instant-Messenger, E-Mail oder kommerzielle Soziale Netzwerke wie Facebook, YouTube oder WhatsApp ergänzen analoge Kommunikatioskanäle, verdrängen diese oder schaffen Austauschprozesse zwischen Menschen, die es vorher in dieser Form nicht gab.[1]

Der Zugang zu den neuen technischen Kommunikationsmöglichkeiten transformiert Stück für Stück die alltäglichen Routinen durch den Austausch und Zugriff auf Informationen, die vorher nicht verfügbar waren und ermöglicht damit neue, zeitlich und räumlich verteilte Netzwerke von Menschen mit ähnlichen Interessen. Gleichzeitig ist dies auch ein privilegierter Zugang, da nicht alle Menschen einen Zugang zu den neuen technischen Möglichkeiten besitzen, wie es der Diskurs über die *Digitale Ungleichheit* zeigt.

Neue Informations- und Kommunikationsmöglichkeiten verändern auch Formen des Lernens im Zuge der Nutzung digitaler Medien. Die zunehmende Nutzung digitaler Medien ist nicht nur Bestandteil des lebenslangen Lernens: Die gegenwärtige Corona-Pandemie wirkt wie ein Brennglas auf diese Entwicklung. Das Gebot des sozialen Abstands führt zur digitalen Annäherung im *Global Village* (vgl. McLuhan 1962): Arbeitgeber:innen delegieren ihre Angestellten in die Lohnarbeit in die eigenen vier Wänden und Schulen und Universitäten werden gezwungen, die zuvor stiefmütterlich behandelten Formen des Fernunterrichts als neuen Modus Operandi zu betreiben. Dafür bedarf es geeigneter Methoden, um idealerweise ähnlich starke Austauschprozesse zu ermöglichen, wie es zuvor durch direkte physische Interaktion erfolgt ist. Dezentrales und zeitversetztes Arbeiten und Lernen mit internetbasierten Technologien bedarf anderer Anforderungen an die Lernenden - egal ob es sich um arbeitsbezogene Probleme im betrieblichen Kontext handelt oder ob es sich um die Aneignung von Wissensbeständen im Rahmen von Qualifikationsprozessen geht.

Internetgestützte Technologien ermöglichen neue Arrangements von Lernsituationen. Informationsquellen können unabhängig von Ort und Zeit erschlossen werden. Auch die Rückkopplung zwischen Lernenden und Lehrenden unterliegen nicht mehr geographischen Grenzen. Kommunikation zwischen den Beteiligten einer Lernumgebung kann synchron, also in Echtzeit, erfolgen oder auch zeitversetzt, wie bei Foren oder E-Mail-Diskussionen. Lernen ist weder an die Präsenz an bestimmten Orten noch an die Orientie-

[1] In dieser Forschungsarbeit verwende ich den Gender-Doppelpunkt ':'. Er dient als Platzhalter für alle möglichen Geschlechtsidentitäten, auch jenseits der Binarität von männlich und weiblich.

rung an gemeinsamen Seminarzeiten gebunden. Daraus ergeben sich neue Möglichkeiten für formale Lernumgebungen, wie sie beispielsweise an Schulen und Universitäten bestehen und unter dem Schlagwort E-Learning zusammengefasst werden. Für diesen Zweck werden computergestützte Lernmanagementsysteme[2] beziehungsweise Netzwerke[3] institutionell vorbereitet.

Unter dem Einfluss neuer technischer Möglichkeiten verändern sich aber auch informelle Lernumgebungen. Durch kollaborativ genutzte Werkzeuge und Datenbanken, die über das Internet zugänglich sind, können Lernende sich unabhängig ihres Aufenthaltsortes miteinander vernetzen. Lernen ist dabei nicht immer ein bewusstes Ziel, weil meistens die Lösung von konkreten Problemen angestrebt wird. Dennoch wird dabei gelernt: Bei der beruflichen Bildung liegen diverse Studien vor, die den quantitativen Anteil informeller Lernprozesse am Lernen in einem hohen zweistelligen Prozentbereich verorten (vgl. Frost et al. 2007, S. 3). Lernen ist dabei nicht das eigentliche Ziel, sondern stellt ein Mittel zur Zielerreichung dar. Lernende nutzen dafür thematische Informationsangebote im Internet: Auf einschlägigen Foren, Informationsportalen oder Sozialen Netzwerken finden sie Gleichgesinnte mit ähnlichen Interessen, mit denen sie sich austauschen oder dort Informationen suchen und bewerten können.

Die Erforschung von informellen, digitalen Lernumgebungen und Lernprozessen ist der Schwerpunkt der vorliegenden Arbeit. Gegenstand der Analyse sind Entwickler:innen innerhalb der Free Software beziehungsweise der Open-Source Bewegung. Diese kooperieren mit dem Ziel, gemeinschaftlich neue Soft- und Hardware zu entwickeln. Das verbindende Medium zwischen den Lernenden ist das Internet. Dadurch können dezentral vernetzte Lerngruppen aufgebaut werden, die ohne physische Gruppentreffen auskommen können. Diverse internetbasierte Werkzeuge dienen den Lerngruppen zur Koordination, Diskussion und Dokumentation ihrer Arbeit und schaffen ein soziales Netzwerk zwischen den Beteiligten. Die Lerngruppen sind informell. Dies bedeutet, dass es keine sanktionsfähige Institution gibt, die Ziele, Inhalte oder Arbeitsweisen vorgibt. Die Steuerungsfunktion übernimmt jedes Mitglied der Entwicklungsgemeinschaft selbst.

1.1 Forschungsfragen

Ziel der Arbeit ist es zu analysieren, welche Kompetenzen die Mitglieder bei informellen Lernprozessen erwerben. Zu diesem Zweck werden die öffentlich zugängliche Kommunikations- und Austauschprozesse eines Free Software / Open-Source-Netzwerks mit verschieden Methoden ausgewertet und zusätzlich die Mitglieder in diversen anderen Netzwerken befragt. Mit Hilfe der Interviews wird versucht, informelles Lernen innerhalb

[2] Für Studieninhalte im Medizinstudium stehen verschiedene kommerzielle Lernportale zur Verfügung, welche zum Beispiel durch Anatomie-Videos und Multiple-Choice-Tests die Prüfungsvorbereitung erleichtern.

[3] In dem Lernmanagementsystem StudIP können für jede universitäre Vorlesungen und Seminare Ordner angelegt werden, auf welche Lehrende und Lernende Zugriff haben, um Inhalte zu teilen beziehungsweise selbst Inhalte erstellen können.

solcher Lernumgebungen zu beschreiben. Es werden Rahmenbedingungen abgeleitet, die für das Lernen in internetbasierten Netzwerken von entscheidender Bedeutung sind und wie sie sich auf den Kompetenzerwerb auswirken.

Bei der Betrachtung von informellen, internetbasierten Netzwerken stehen die jeweiligen Lernprozesse der Mitglieder im Mittelpunkt. Die zentralen Forschungsfragen dieser Arbeit lauten wie folgt:

1. *Warum beteiligen sich Personen an den Interaktionen in solchen Netzwerken?*

   Die Forschungsfrage zielt auf das Ergründen der Interessen der jeweiligen Personen, die durch die Partizipation in solchen Strukturen verfolgt werden. Hier ist die Frage von Bedeutung, ob sich die Interessen im Verlauf des Engagements verändern und welche biographischen Bezüge und Einstellungen die Personen vor dem erstmaligen Engagement in solch einem Netzwerke aufweisen. Nicht zu vernachlässigen ist der übergeordnete Kontext dieser Forschungsfrage: Welche Bedeutung hat die Partizipation für die Lebensqualität und möglichen Perspektiven der Lebensgestaltung?

2. *Wie wird in informellen, internetbasierten Netzwerken gelernt?*

   Durch die Betrachtung der Aktivitäten einzelner Personen im Rahmen des Lernnetzwerks werden die Lernformen analysiert und die jeweils genutzten Informationsquellen identifiziert. Ein Schwerpunkt der Betrachtung liegt auf der Netzwerkstruktur dieser Lernumgebung. Dadurch wird die Frage des Einflusses von anderen Lernenden auf die individuellen Lernprozesse relevant. Die Verknüpfung des Lernens mit der Struktur des Netzwerkes eröffnet eine neue Analyseebene: Wie beeinflussen unterschiedliche organisierte Netzwerke die Lernprozesse ihrer Mitglieder?

3. *Was lernen die Mitglieder durch ihr Engagement in den Netzwerken?*

   Der individuelle Kompetenzaufbau durch die Partizipation in solchen Netzwerken ist für das Forschungsinteresse hochgradig interessant, denn dieser geschieht ohne eine pädagogische Steuerung von außen. Weiterhin soll untersucht werden, wie sich Gelerntes anhand existierender Kompetenzmodelle in eine Ordnungssystem überführen lässt.

Eine Evaluation des individuellen Kompetenzaufbaus einzelner Personen aufgrund der Aktivitäten in solchen Netzwerken wird nicht angestrebt, da diese Arbeit sich auf die Rahmenbedingungen des Lernens in informellen Kontexten fokussiert. Dies erfolgt unter spezieller Betrachtung des kooperativen und kollaborativen Lernens unter den strukturellen Rahmenbedingungen von internetgestützten Netzwerken.

1.2 Aufbau der Arbeit

Die Forschungsarbeit gliedert sich in zwei Teile: Erstes dem theoretischen Teil mit den grundlegenden Begriffen samt einer Beschreibung des Gegenstands dieser Forschungsar-

beit und notwendigen Lerntheorien. Im zweiten empirischen Teil erfolgt eine Darstellung der Forschungsergebnisse und eine Einordnung in den bildungswissenschaftlichen Diskurs.

Der erste theoretische Teil besteht aus fünf Kapiteln. Im Anschluss an die Einleitung erfolgt im zweiten Kapitel eine Beschreibung des Forschungsgegenstandes der Freien Software beziehungsweise Open-Source Bewegung. Diese ist eine Soziale Bewegung, die auf eine jahrzehnte lange Historie zurückblickt und sich in dieser Zeit global vernetzt hat und sich auch diverse Interessensvertretungen herausdifferenziert haben. Bei der Freien Software / Open-Source Bewegung handelt es sich auch um eine stark fragmentierte Bewegung, die sich in tausenden Gruppen mit variabler Größe und Struktur jeweils für sich ein selbstgestecktes Ziel innerhalb dieser Bewegung verfolgen.

Im daran anschließendem Kapitel erfolgt eine Beschreibung des bisherigen Forschungsstandes über solche informellen Netzwerke und deren Akteur:innen.

Im dritten Kapitel wird ein Lernbegriff für diese Arbeit entwickelt. Dafür wird das Lernen aus zwei unterschiedlichen Perspektiven beschrieben: Lernen ist sowohl ein individueller Prozess als auch ein sozial verankerter Prozess. Es werden grundlegende Konzepte und Lerntheorien in ihrer historischen Entwicklung vorgestellt und auf ihre Eignung geprüft. Ein weiterer Schwerpunkt dieses Kapitels gilt den besonderen Rahmenbedingungen des Lernens in internetbasierten Netzwerken. Lernen in dezentralen Lernnetzwerken wird in den Kontext des E-Learnings eingeordnet.

Das vierte Kapitel beschäftigt sich mit dem Forschungsstand über Freie Software beziehungsweise Open-Source Projekte und deren Akteur:innen. Es werden typischen Strukturformen solcher Netzwerke betrachtet und oft genutzte Werkzeuge und Arbeitsweisen beschrieben. Weiterhin werden die für diese Arbeit relevanten Forschungsergebnisse von vorherigen Studien vorgestellt. Diese beziehen auf die kommunikativen Austauschprozesse innerhalb solcher Netzwerke und den Interessen der Akteur:innen sowie deren soziodemographischen Merkmale.

Im letzten Kapitel des theoretischen Teils wird das Forschungsdesign vorgestellt. Es handelt sich um einen Mixed-Method-Ansatz mit einer Beitragsanalyse der öffentlich zugänglichen E-Mail-Kommunikation und die Analyse der individuellen Beiträge zur gemeinschaftlichen Entwicklungsarbeit einer ausgewählten Projektgruppe. Dieser Datenkorpus wird mit Leitfadeninterviews mit Akteur:innen von verschiedenen anderen Gruppen ergänzt.

Mit dem sechsten Kapitel beginnt der empirische Teil dieser Forschungsarbeit. Zur Einführung werden die Interaktionen in einem Freie Software / Open-Source Projekt im Längsschnitt betrachtet. Die Analyse der E-Mail-Kommunikation auf der projektweiten Mailingliste bildet den Hauptteil dieses Kapitels. Exemplarisch werden zwei Personen dieses Projekts anhand ihrer Interaktionen näher untersucht. An ihnen werden Austauschprozesse mit den anderen Akteur:innen des Projekts vorgestellt, um typische Kommunikations- und Lernprozesse in solchen Netzwerken zu demonstrieren.

Das siebte Kapitel dient der Beantwortung der Forschungsfrage nach den Interessen der engagierten Personen. Es beginnt mit einer Vorstellungen der Gründe, die die Akteur:innen zu einem Engagement in solchen Netzwerken bewegen und wie diese sich im Verlauf der Zeit verändern. Dabei wird zwischen Vorbedingungen, Auslösern für ein erstes Engagement und dem Verfolgen von konkreten Interessen während des Engagements unterschieden.

Anschließend erfolgt im achten Kapitel der Fokus auf die Lernprozesse der Akteur:innen. Dafür werden zunächst typische Handlungen im Rahmen der Softwareentwicklung beschrieben. Dadurch kann gezeigt werden, dass ein Großteil der Lernprozesse einem problemorientierten Anwendungsbereich entspringt. Bei der Lösung nutzen die Entwickler:innen unterschiedliche Strategien, die unter anderem vom Zugriff auf unterschiedliche Informationsquellen abhängig sind. Diese werden im im Einzelnen vorgestellt.

Der Kompetenzaufbau wird im daran anschließenden neunten Kapitel thematisiert. Anhand der Kontrastierung von zwei Kompetenzmodellen werden unterschiedliche Kompetenzbereiche differenziert. In diese erfolgt die Einordnung des Kompetenzaufbaus durch die Entwicklungsarbeit. Fallbezogen wird eine entsprechende lerntheoretische Zuordnung vorgenommen. Als weiteren Analyseschritt erfolgt eine Typologisierung der Entwickler:innen: Anhand von Interessen, präferierten Lernstrategien und aufgebauten Kompetenzen werden vier charakteristische Typen vorgestellt und anhand von typischen Vertreter:innen veranschaulicht.

Im Schlusskapitel dieser Arbeit werden die Beiträge dieser Arbeit zur Lerntheorie herausgearbeitet. Aufgrund der Forschungsergebnisse werden Konsequenzen für die pädagogischen Gestaltung solcher Lernarrangements abgeleitet. Zum Abschluss wird noch einmal der Bezug zu kritischen Bildungstheorien hergestellt und mögliche Erweiterungsvorschläge diskutiert.

2 Forschen in informellen, internetbasierten Netzwerken

Zur Beschreibung des Forschungsgegenstands werden im Folgendem die notwendigen Begriffe erläutert. Dies bezieht sich insbesondere auf die Begriffe von *Freier* beziehungsweise *Open-Source* Software und ihrer Abgrenzung zu proprietärer[4] Soft- und Hardware. Daran anknüpfend werden die besonderen Bedingungen und Strukturen beschrieben, die das Umfeld solcher Software prägen. Zu diesem Zweck wird die Gemeinschaft von Nutzer:innen und Entwickler:innen, die sich für Freie und Open-Source Soft- und Hardware einsetzen, als soziale Bewegung herausgearbeitet. Anschließend wird die Entwicklungsgemeinschaft beschrieben, die sich als eigentliche Keimzelle um eine konkrete Soft- oder Hardware bildet und in der die Entwickler:innen kooperativ und kollaborativ sich im gegenseitigen Austausch engagieren.

2.1 Freie Software und Open-Source Software

Als Freie oder Open-Source wird Soft- oder Hardware bezeichnet, deren Aufbau und Funktionsweise offen dokumentiert ist. Bei Software, drückt sich in der Offenlegung des Quellcodes[5] aus, was als Open-Source bezeichnet wird. Im Gegensatz zur reinen Auslieferung des Maschinencodes[6], wie es bei proprietärer Software (zum Beispiel Microsoft Windows) geschieht, ist es damit auch möglich, die Funktionsweise des Programms nachzuvollziehen und zu verändern. Dieser Aspekt ist jedoch nicht nur ein Mehrwert, der nur für Programmierkundige relevant ist. Hiermit entspannt sich auch eine politische Dimension, die für alle Anwender:innen relevant ist:

> „Die Chancen zur Teilhabe an gesellschaftlichen Prozessen werden in zunehmendem Maße nicht mehr nur durch den Zugang zu klassischer Bildung vermittelt, wie es in der fordistischen Gesellschaftsstruktur noch der Fall war. Vielmehr ist im informationalen Modus der Zugang zu wertvollen Informationen mit Hilfe von ausgefeilten Softwaretechnologien im Rahmen eines lebenslangen Lernprozesses von entscheidender Bedeutung. [...] Und Software ist die Technologie, die Zugang ermöglicht, steuert und begrenzt. Die freie

[4] Das Adjektiv 'proprietär' bedeutet, dass sich etwas in Eigentum befindet. In diesem Fall des Unternehmens, welches die Soft- oder Hardware hergestellt hat. Die Nutzung solcher Soft- und Hardware ist durch das betreffende Unternehmen juristisch streng reglementiert und Nutzungsrechte müssen in der Regel finanziell erworben werden. Um das Monopol auf solche Produkte zu schützen, werden Quellcode oder die Konstruktionspläne geheim gehalten.

[5] Der Quellcode sind die von Programmierer:innen verfassten Anweisungen, welche die Funktionsweise eines Programms beschreiben. Sie sind in einer Programmiersprache formuliert und erst für den Computer ausführbar, wenn der Quellcode zuvor durch einen Compiler oder einen Interpreter in Maschinencode umgewandelt wurde.

[6] Maschinencode ist die für eine spezifische Computerarchitektur übersetzen Arbeitsanweisungen. Sie sind für den Computer direkt ausführbar. Der Maschinencode ist binär kodiert und ist für Programmier:innen nicht oder nur unter Nutzung von speziellen Programmen (Disassembler) auf sehr niedrigem Niveau zu verstehen.

> bzw. offene Verfügbarkeit von digitalisiertem Wissen in Programm- und/oder Datenform führt also auch für den reinen Anwender zu einer Steigerung seiner Möglichkeiten, an wissenszentrierten Prozessen nachhaltig teilzuhaben und damit zu einer Steigerung der 'informationalen Chancengleichheit'." (Zimmermann 2004, S. 363)

Freie Software und Open-Source Software ermöglichen die Nutzung von Computern und den Zugang zu Informationsressourcen ohne die Zahlung von Lizenzgebühren, wie es proprietäre Software der Fall ist. Dies gilt in gleicher Linie für die auf dem Computer gespeicherten Daten.[7] Andere Programmierer:innen können durch offen dokumentierte Dateiformate und Programmierschnittstellen kompatible Software entwickeln. Nutzer:innen wie auch Entwickler:innen von Freier und Open-Source Software sind dadurch nicht auf die Einwilligung des:der ursprünglichen Entwickler:in einer Software angewiesen. Im Kern geht es dabei um Freiheitsrechten, welche sich im Begriff von *Freier Software* niederschlagen:

> „When we call software 'free,' we mean that it respects the users' essential freedoms: the freedom to run it, to study and change it, and to redistribute copies with or without changes. This is a matter of freedom, not price, so think of 'free speech', not 'free beer'." (Stallman 2015, S. 75).

Dadurch, dass die Weitergabe Freier Software immer unter der gleichen Lizenz durchgeführt werden muss, ist sichergestellt, dass sich der Anteil der zirkulierenden Freien Software stetig vermehrt.[8] Auf diese Weise entstand das Betriebssystem GNU/Linux oder der Browser Firefox als Agglomerat von Open-Source Projekten. Die Offenheit des Quelltextes ermöglicht es Programmierkundigen die Programme nach eigenen Wünschen anzupassen oder auch anhand der Nachvollziehbarkeit der Funktionsweise das Programmieren zu erlernen. Dies ist allerdings keine Voraussetzung, um diese Software zu nutzen. Wer nicht selbst die Validität und Qualität der Software prüfen oder verändern kann, profitiert davon, dass andere es können und ihre Ergebnisse der Allgemeinheit wieder zur Verfügung stellen. Daraus entsteht eine Gemeinschaft, die stetig im gegenseitigem Austausch Soft- und Hardware weiterentwickelt und damit eine Wissensallmende aufbaut. Offenheit des Quellcodes oder der Bauanleitungen ermöglicht gegenseitige Hilfe. So profitieren alle von den Kenntnissen Einzelner und jede:r hat die theoretische Möglichkeit, dieselben Kompetenzen zu erlangen.

Methodisch wird dies durch eine Lizenz abgesichert, unter der die Soft- und Hardware veröffentlicht wird. In ihr sind die Rechte und Pflichten dokumentiert, unter denen die

[7] Die Daten z.B. Texte, Tabellen oder Audio/Video-Aufnahmen die auf einem Computer mit proprietärer Software eingegeben werden, werden oft auch in proprietären Dateien auf dem Computer gespeichert. Insofern können diese Daten dann auch nur wieder mit der betreffenden Software auch wieder bearbeitet werden, da die Art der Komprimierung oder Verschlüsselung der Dateien auf dem Speichermedium nur dem Hersteller der Software bekannt ist und er deswegen ein Monopol darauf hat.

[8] Wird eine Software, die unter einer freien Softwarelizenz veröffentlicht wurde, verändert, muss diese veränderte Version auch wieder unter der gleichen Lizenz veröffentlicht werden. Eine Monopolisierung von ursprünglich Freier Software durch eine Veränderung ist dadurch ausgeschlossen.

Nutzung stattzufinden hat. Größere Popularität erlangten beispielsweise die *GPL (GNU General Public License)*, die ursprünglich 1998 von Richard Stallman in ihrer ersten Version veröffentlicht wurde. Der Linux-Kernel[9] ist beispielsweise eine Software, die auf der GPL basiert. Daneben gibt es zahlreiche andere freie Lizenzen wie beispielsweise die Lizenzen der Organisation *Creative Commons*, mit der beispielsweise Wikipedia arbeitet.

Auch der Aspekt der Sicherheit von quelloffener Software ist nicht zu vernachlässigen: Erst die Offenlegung des Quellcodes ermöglicht die Kontrolle, dass die Software auch die zugesicherte Funktionalität beinhaltet und keine geheimen Hintertüren eingebaut sind. Mit diesen könnten staatliche oder nicht-staatliche Akteur:innen Daten überwachen und manpipulieren (vgl. Grassmuck 2004, S. 361). Damit entspannt sich ein weiterer Freiheitsbegriff für quelloffene Softaware: Freiheit als Schutz vor Herrschaft.

2.2 Open Source und Freie Software als Soziale Bewegung

Open-Source heißt zunächst einmal nur die Lesbarkeit, also damit die Möglichkeit zum Verständnis vom Aufbau und der inneren Funktionen und Prozesse. Es ist damit ein Attribut, was Soft- oder Hardware haben kann oder nicht. Freie Software ist oft auch Open-Source Software und umgekehrt. Beide unterliegen jedoch unterschiedlichen Lizenzen, die im Detail Unterschiede aufweisen und einen ideologischen Bruch innerhalb der Open-Source-Gemeinschaft wiederspiegeln.

Freie Software ist eine Bezeichnung, die durch die 1985 von Stallman gegründete *Free Software Foundation* vorangetrieben wird. Sie betont die Freiheitsrechte, die den Nutzenden durch die Lizenz garantiert wird. Per Definition ist Freie Software dann gegeben, wenn die Nutzung der Software zu jedem Zweck möglich ist, das Programm in seiner Funktionsweise studiert und nach eigenem Ermessen verändert werden kann und Kopien der Software, auch in modifizierter Form ohne Einschränkungen verteilt werden können (vgl. Stallmann 2015, S. 3).

1998 spaltete sich ein Teil der Free Software Gemeinschaft ab und die *Open Source Initiative* wurde gegründet. Deren Definition entspricht weitgehend der der Free Software Foundation, jedoch nutzt diese stattdessen den Begriff *Open-Source*. Stallman kontrastiert die ideologischen Unterschiede beider Organisationen wie folgt:

> „The two terms describe almost the same category of software, but they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement. For the free software movement, free software is an ethical imperative, because only free software respects the users' freedom. By contrast, the philosophy of open source considers issues in terms of how to make software 'better'—in a practical sense

[9] Der Linux-Kernel ist der Betriebssystemkern, der die Prozess- und Datenorganisation steuert. Zusammen mit der auf dem Kernel aufbauenden GNU-Software (GNU steht für das rekursive Akronym *GNU's Not Unix*) bildet es zusammen das GNU/Linux-Betriebssystem, was fälschlicherweise oft nur als Linux-Betriebssystem bezeichnet wird. Bekannte GNU/Linux-Distributionen sind beispielsweise Ubuntu, Fedora oder Debian.

> only. It says that nonfree software is an inferior solution to the practical problem at hand. For the free software movement, however, nonfree software is a social problem, and the solution is to stop using it and move to free software.“ (ebd., S. 76)

Beide Organisationen verfolgen das Ziel, Open-Source Software beziehungsweise Freie Software zu fördern. Trotz ideologischer Unterschiede tragen beide zur Verbreitung quelloffener Soft- und Hardware bei. Der gemeinsame Gegenspieler ist proprietäre, also unfreie, Soft- und Hardware. Deren Funktionsweise wird den Nutzenden vorenthalten. Deshalb ist eine Anpassung an andere Bedürfnisse nicht möglich oder nur mit Hilfe des ursprünglichen Herstellers durchführbar. Die Nutzung proprietärer Software ist meist an die Zahlung einer Gebühr gekoppelt und das Kopieren und die Weitergabe der Software wird durch das Copyright rechtlich eingeschränkt.

Um den gemeinsamen Prinzipien und Zielen beider Strömungen Rechnung zu tragen, wird im Folgendem von Free / Libre[10] and Open-Source Software, abgekürzt *FLOSS*, gesprochen. Da die Quelloffenheit mit der Möglichkeit der Veränderung und uneingeschränkte Weitergabe für jegliche Zwecke prinzipiell auch auf Hardware[11] übertragbar ist, wird diese in der Abkürzung subsumiert.

Die Wahrung der Freiheitsrechte durch die Förderung von FLOSS wird aber nicht nur durch diese Organisationen bewerkstelligt. Sie beschränken sich auf Öffentlichkeitsarbeit, Kapitalakquise und juristische Beratung bei Lizenzverstößen. Die hauptsächliche FLOSS-Entwicklung wird mittlerweile mehrheitlich durch ehrenamtliches Engagement etlicher Entwickler:innen weltweit geleistet. Aufgrund dieser Struktur lässt sich die FLOSS-Gemeinschaft auch als soziale Bewegung beschreiben. Eine Definition von sozialen Bewegungen die den Netzwerkcharakter betont, liefert die Politikwissenschaftlerin Donatella della Porta:

> „Social movements are informal networks linking a plurality of individuals and groups, more or less structured from an organizational point of view. Whereas parties or pressure groups have somewhat well-defined organizational bounderies, with participation normally verified by a membership card, social movements are instead composed of loose, weakly linked networks of individuals who feel part of a collective effort. [...] One distinctive characteristic of a social movement is the possibility of belonging and feeling involved

[10] Das Wort „Libre“ wird oft als Zusatz verwendet, da es in der französischen und spanischen Sprache unmissverständlich für Freiheit steht und damit nicht mit frei/free, im Sinne von kostenfrei, verwechselt zu werden.

[11] Freie bzw. Open-Source Hardware zeichnet sich durch die Offenlegung der Konstruktions- und Funktionsdokumentation aus und der Garantie von zusätzlichen Nutzungsrechten. Im Gegensatz zu Software ist die Vervielfältigung von Hardware jedoch mit bedeutend höheren Kosten verbunden, da die einzelnen Bauteile beschafft, ggf. verändert und zusammengefügt werden müssen. Softwarekopien brauchen nur Speicherplatz und die entsprechenden Schreib- und Lesegeräte. Freie Hardware sind beispielsweise die 3D-Drucker vom RepRap-Projekt oder der Arduino Mikrocontroller. Auch für Freie Hardware gibt es angepasste Lizenzen.

> in collective action without necessarily being a member of a specific organization.“ (della Porta 2007, S. 7)

Sie legt ihren Fokus auf die organisatorischen Strukturen von interagierenden Gruppen und Individuen, wie sie für die FLOSS-Bewegung typisch ist. Diese werden im folgenden Abschnitt näher ausgeführt. Wiederkehrende Merkmale in anderen Definitionen von Sozialen Bewegungen sind: gemeinsame Ziele, Dauerhaftigkeit und variable Organisations- und Aktionsformen (vgl. Raschke 1988, S. 77; McAdam & Snow 1997, S. XVIII; Brand et al. 1986, S. 36f).

Gemeinsame Ziele

Die Beweggründe, weshalb sich Individuen und Gruppen innerhalb der FLOSS-Bewegung engagieren sind vielfältig. Von der institutionellen Ebene her betrachtet, steckt sich die Free Software Foundation dezidiert politische Ziele zum Schutz der Freiheit von Soft- und Hardware-Nutzer:innen:

> „As our society grows more dependent on computers, the software we run is of critical importance to securing the future of a free society. Free software is about having control over the technology we use in our homes, schools and businesses, where computers work for our individual and communal benefit, not for proprietary software companies or governments who might seek to restrict and monitor us. The Free Software Foundation exclusively uses free software to perform its work. The Free Software Foundation is working to secure freedom for computer users by promoting the development and use of free (as in freedom) software and documentation—particularly the GNU operating system—and by campaigning against threats to computer user freedom like Digital Restrictions Management (DRM) and software patents.“ (Free Software Foundation o.D.)

Im Gegensatz dazu betont die Open Source Initiave den pragmatischen Nutzen[12] von Open-Source Software und hat es sich zum Ziel gesetzt, Aktivist:innen der FLOSS-Gemeinschaft zu vernetzen und Aufklärungsarbeit über den Nutzen von Open-Source Software zu leisten (vgl. Open Source Initiative o.D.). Auf individueller Ebene haben die Mitglieder der weltweit agierenden FLOSS-Gemeinschaften verschiedene Beweggründe für ihr Engagement. Diese reichen von gesellschaftspolitischen Ansprüchen, über die individuelle Freude beim Problemlösen und Austausch innerhalb der Gemeinschaft bis hin zu monetären Anreizen. Eine genaue Beschreibung individueller Anreizstrukturen wird in Kapitel 7 bei der Betrachtung der Interessen von Entwickler:innen in der FLOSS-Bewegung ausgeführt.

[12] Für die Open Source Initiative besteht der Mehrwert von Open-Source Software gegenüber proprietärer Software in höherer Qualität, bessere Reliabilität und Flexibilität, geringeren Kosten und den Abbau von Herstellerabhängigkeiten (vgl. Open Source Initiative o.D.).

Kontinuität

Um soziale Bewegungen von singulären kollektiven Ereignissen wie beispielsweise Aufständen abzugrenzen, wird ein langfristiger Protestzyklus von mehreren Jahren vorausgesetzt (vgl. Raschke 1988, S.78). Als institutionalisierter Teil der FLOSS-Bewegung sind die Free Software Foundation und die Open Source Initiative bereits seit mehreren Jahrzehnten aktiv. Durch Publikationen, Kampagnen und Tagungen wird versucht, FLOSS zu fördern. Daneben gibt es Zusammenschlüsse, die jeweils mit einem spezifischen Fokus regelmäßig Konferenzen veranstalten. In Deutschland sind dies beispielsweise die regionalen *Linux-Tage*[13], die *FrOSCon (Free and Open Source Conference)* oder der *OpenTechSummit*.

Darüber hinaus gibt es eine ständig wachsende Zahl von neuen FLOSS-Projekten, die die Einsatzmöglichkeiten von FLOSS ständig erweitern und die teilweise auch eigene Konferenzen organisieren. Eines der größeren FLOSS-Internetportale *SourceForge*, welches kleineren FLOSS-Projekten eine Entwicklungsumgebung bietet, zählt nach Selbstaussage momentan ca. 502.000 Projekte, in denen sich mehrere Millionen Entwickler:innen engagieren (vgl. SourceForge o.D.). Eine bestimmte Auswahl an FLOSS wird in verschiedenen GNU/Linux-Distributionen aufeinander abgestimmt und in einem Paket zusammengestellt. Debian, eine der ältesten Distributionen auf der auch die Ubuntu-Distribution aufbaut, wird seit 1993 kontinuierlich weiterentwickelt (vgl. Debian o.D.).

Variable Organisations- und Aktionsformen

Auf lokaler Ebene organisieren sich vielerorts *Linux User Groups* meist in Form von offenen Stammtischen. Sie tauschen sich über Themen rund um GNU/Linux aus und sind eine treibende Kraft bei der Organisation von regionalen Linux-Tagen. Momentan gibt es in Deutschland circa 260 solcher Gruppen (vgl. LinuxUser o.D.).

Thematisch fokussierter arbeiten die Entwicklungsgemeinschaften, die sich um ein konkretes FLOSS-Projekt herum bilden. Durch deren Beiträge vergrößert sich das Spektrum an Einsatzbereichen, in denen FLOSS eingesetzt werden kann. Die bereits beschriebenen Lobbyorganisationen kümmern sich mit ihren institutionellen Möglichkeiten zum Schutz von FLOSS durch Lizenzierung, Pressearbeit und Kampagnen zu Schlüsselthemen.

2.3 FLOSS-Entwicklungsgemeinschaft

Innerhalb des heterogenen Feldes der FLOSS-Bewegung gilt der Fokus dieser Arbeit den Gruppen von Aktiven, die konkret an der Herstellung von FLOSS beteiligt sind. Bei FLOSS-Entwicklungsgemeinschaften handelt es sich um "kooperative Softwareentwicklung innerhalb von sozialen Netzwerken [...] [als] deterritoriale, politische Vergemein-

[13] Die Linux-Tage sind Konferenzen mit Fokus auf Vorträge und Seminare rund um FLOSS. Etabliert haben sich beispielsweise die jährlichen stattfindenden *Chemnitzer Linux-Tage* und die *Kieler Open Source und Linux Tage*.

schaftung mit weitergehenden Zielen" (Tepe & Hepp 2008a, S. 172). Die Arbeit innerhalb einer solchen Entwicklungsgemeinschaft erfolgt in der Regel unbezahlt in der Freizeit. Die Projektbeiträge der Mitglieder reichen von der Einreichung von Programmcode, Übersetzung von Dokumentation und Bedienungsoberflächen bis Grafikdesign, Testen der Funktionalität bis zur Einsendung von Verbesserungsvorschlägen.

Wie die Gemeinschaft sich strukturiert und organisiert kann sehr verschieden sein. Von Projektteams von zwei Entwickler:innen mit flacher Hierarchie bis zu arbeitsteilig organisierten Großteams mit schwankenden Zahlen von mehreren Hundert Projektbeteiligten mit ausgeprägten Machtunterschieden, sind alle Mischformen vertreten. Das verbindende Element der FLOSS-Entwicklungsgemeinschaften in dieser Arbeit ist die internetgestützte Arbeitsweise. Damit sind die einzelnen Projektbeteiligten nicht mehr notwendigerweise an einen Ort gebunden. Durch die gemeinsame Nutzung von E-Mail, Online-Foren, Wikis und anderen internetgestützten Werkzeugen wird die kooperative Entwicklung von der geographischen Herkunft entkoppelt. Einzig ein Internetzugang, Zeit für die Projektarbeit und die Bereitschaft, sich auf die projektspezifischen Kommunikations-, Dokumentations- und Entscheidungsverfahren einzulassen, sind die einzigen Voraussetzungen, um sich in offenen FLOSS-Entwicklungsgemeinschaften zu engagieren.

3 Rahmenbedingungen von Lernprozessen

Nachdem der Forschungsgegenstand und die Forschungsfragen dieser Arbeit dargelegt wurden, gilt es nun, theoretisch fundierte Analysekriterien abzuleiten. Sie bilden das Grundgerüst für das Design der Forschungsmethoden. Dazu braucht es einen Lernbegriff, der weit genug gefasst ist, um möglichst viele Facetten des Lernens der Entwickler:innen erfassen zu können. Gleichzeitig muss er aber auch konkret genug sein, um die Besonderheiten des Lernens innerhalb der konkreten Lernumgebung am Beispiel einer FLOSS-Entwicklungsgemeinschaft herausarbeiten zu können.

Um dies leisten zu können, wird Lernen in solchen Entwicklungsgemeinschaften im Folgenden aus drei unterschiedlichen Perspektiven betrachtet: Lernen als individueller Prozess, Lernen im sozialen Kontext und die Rahmenbedingungen des Lernens in internetgestützten, informellen Lernumgebungen.

Die erste Perspektive betrachtet Lernen als individuellen Prozess. Dies ist nicht so selbstverständlich, wie es zunächst klingt. Auch heute noch nutzen viele Personen, die beruflich in Lehr-Lern-Situationen arbeiten, einen mechanistischen, unterkomplexen Lernbegriff. Dieser geht davon aus, dass Lernen bei allen Menschen gleich abläuft: Lernen ist von außen initiiert und wird mittels geeigneter Methoden und Unterrichtsmaterialien moderiert. Diese Herangehensweise ist oftmals auch durch ökonomische Sachzwänge verursacht. Dennoch lassen sich auch finanzielle Rahmenbedingungen von Bildungsinstitutionen nur legitim ändern, wenn sich auch außerhalb der Bildungswissenschaft ein adäquater Lernbegriff im Alltagsverstand durchsetzt. In dem ersten Teil dieses Kapitels wird ein Lernbegriff erarbeitet und die beiden wichtigsten Faktoren, Interesse und Situiertheit, beschrieben. Anschließend wird das Ziel von Lernprozessen genauer spezifiziert. Fokus dieser Arbeit ist der Aufbau von Kompetenzen in Anlehnung an den *Fähigkeitenansatz* (vgl. Nussbaum 2012). Zur Einordnung der Rahmenbedingungen des Lernens werden abschließend die Lernformen betrachtet, die beim Lernen in solchen Lernumgebungen vorherrschend sind.

Die zweite Perspektive betrachtet Lernen im sozialen Kontext. Es ist sozusagen die komplementäre Perspektive zum individuellen Prozess. Ohne diese zweite Perspektive bleibt der Lernbegriff unvollständig, weswegen diese Komponente des Lernens zwingend in ihrer vollen Bedeutung dargestellt werden muss. Da der Stellenwert der sozialen Dimension erst ein Produkt der jüngeren historischen Entwicklung von Lerntheorien ist, wird bei der Beschreibung dieser Perspektiven die Entwicklung dieser Dimension anhand der größten Lerntheorien nachvollzogen. Anhand der Entwicklungsgeschichte von Lerntheorien wird die Evolution der sozialen Dimension von Lernen herausgearbeitet. Als Resümee wird die Lerntheorie und mit ihr verbundene Konzepte vorgestellt, die für diese Arbeit leitend sind.

Die dritte Perspektive ist den konkreten Bedingungen der Lernumgebung gewidmet, in der sich die FLOSS-Entwickler:innen bewegen. Es erfolgt eine Einordnung in die Typologie des E-Learnings unter besonderer Betrachtung des kooperativen und kollaborativen

Lernens in Lerngemeinschaften.

3.1 Lernen als individueller Prozess

Für die Erarbeitung eines Lernbegriffs für diese Arbeit werden zunächst als Problemaufriss die Unzulänglichkeiten eines vereinfachten Lernbegriffs dargestellt, wie er lange durch die Lernpsychologie geprägt wurde. Durch die kognitive Wende fand ein Ausbruch aus der reinen Betrachtung des Lernens als von außen beobachtbarer Prozess statt und bewegte sich hin zu immer komplexer werdenden Lernbegriffen, die sich insbesondere auf die Dimensionen von Interesse und Situiertheit beziehen.

Anschließend wird das Ziel von Lernprozessen umrissen. Anhand des Fähigkeitenansatzes von Martha Nussbaum soll dargestellt werden, inwieweit das Engagement in Entwicklungsgemeinschaften dazu geeignet ist, Grundfähigkeiten aufzubauen.

Im darauf folgenden Abschnitt werden Lernformen betrachtet, die für solche FLOSS-Lerngemeinschaften dominierend sind. Hierbei handelt es sich vor allem um das informelle Lernen. Diese Lernform wird in ihren Grundzügen erläutert und im konkreten Anwendungsszenario im Rahmen von gemeinnütziger Tätigkeit dargestellt.

3.1.1 Suchbewegungen der Psychologie: Die Entdeckung des Individuums

Biologistische Lerntheorien, die auf Verhaltensänderung bei sich ändernden Umweltbedingungen zur Sicherung des Überlebens abzielen, versuchen das Lernen von Tieren auf den Menschen zu übertragen. Der darauf begründete Behaviorismus der Reiz-Reaktions-Psychologie beschreibt Lernen als einen Prozess, der unabhängig vom Individuum erfolgt. Lernen ist in dieser Lesart ein zu verallgemeinernder Prozess mit festgelegten Ursache-Wirkungs-Mechanismen. Dieser kann losgelöst von Lerninhalten und den Lernenden selbst betrachtet werden und äußert sich als von außen beobachtbares Verhalten. Das Individuum als *tabula rasa* wird zwar durch den Lernprozess geprägt, die kognitive Gesamtstruktur des Individuums hat aber keinen Einfluss auf das Lernen.

Im Rahmen der kognitiven Wende erfolgte ein Paradigmenwechsel der sich für die kognitiven Prozesse wie Wahrnehmung, Gedächtnis, Denken und Lernen öffnete. In Nordamerika erfolgte diese Entwicklung ab den 1950er Jahren und wurde maßgeblich von Jerome Bruner und Noam Chomsky beeinflusst, während in Europa die Arbeiten von Jean Piaget oder Alexander Lurija richtungsweisend waren (vgl. Miller 2003, S. 141f). Daraus resultierte die heutige Kognitionspsychologie.

Klaus Holzkamp als Vertreter der Kritischen Psychologie forderte, dass die Psychologie ihr Selbstverständnis als Wissenschaft der Außenbeobachtung aufzugeben habe. Nur durch die Betrachtung von Lernprozessen aus der Perspektive des Lernsubjektes könne das Lernproblem, welches „als psychologisch abgegrast und institutionell befriedet" (Holzkamp 1995, S. 14) galt, wieder für neue Erkenntnisdimensionen geöffnet werden. Er beschreibt Lernhandlungen im Kontext von Vorwissen und Erfahrungen sowie der Kör-

pergebundenheit des Lernens (vgl. ebd., S. 206 - 294. Diese zu begrüßende Sensibilisierung für den Kontext von Lernprozessen scheint im Zuge der Neurowissenschaften wieder verloren zu gehen.

Durch neue Untersuchungsverfahren können Veränderungen im Gehirn auf elektrischer und chemischer Ebene sichtbar gemacht werden. Dies verleitet dazu, Rückschlüsse auf Lernprozesse ausschließlich aus der Außenbeobachtung heraus zu ziehen. Diese sehen Lernen als Veränderung synaptischer Übertragungsstärke an (vgl. Spitzer 2002, S. 146). Die technische Messung der Hirnaktivität geht mit dem Anspruch einher, daraus allgemeine Prinzipien für das Verständnis von Lernprozessen ableiten zu können (vgl. ebd., S. XIV). Dass dies wieder zu einer Verengung des Blickes auf das Lernen führt, beschreiben Michael Göhlich und Jörg Zirfas:

> „Problematisch aus pädagogischer Sicht ist, dass die Lernpsychologie das menschliche Lernen zu erklären, nicht jedoch zu verstehen sucht. Der Sinn des Lernens spielt auch für diese Wissenschaft keine entscheidende Rolle. Hier liegt die große Aufgabe für die pädagogische Sicht auf menschliches Lernen.“ (Göhlich & Zirfas 2007, S. 13)

Das Problem der Außenbetrachtung des Lernens als messbarer und in diesem Sinne als extern kontrollierbarer Prozess wird noch einmal ausführlich bei der Betrachtung der Hauptströmungen der Lerntheorien in Kapitel 3.2 aufgegriffen.

Im weiteren Verlauf dieser Arbeit wird die von Göhlich und Zirfas aufgeworfene Sinnfrage weiter erörtert. Sie ist für das Verständnis von Lernprozessen elementar, da sie statt des behavioristischen Begriffs der Motivation auf individuelle Interessen- und Sinnstrukturen zurückgreift.

3.1.2 Pädagogische Sichtweisen auf das Lernen

Erste Nachweise für pädagogisches Denken reichen bis in die Antike zurück. Die Philosophie beschäftigte sich schon damals mit Sinnfragen und versuchte, menschliches Lernen zu verstehen. So beschrieb Aristoteles, dass das Lernen aus der Bewunderung heraus entspringt und lusterzeugend[14] ist (vgl. Aristoteles 2007, S. 56).

Die Kategorie Sinn lässt sich auf Grundlage einer reinen Außenbeobachtung nicht erheben. Darüber können nur die Lernenden selbst Auskunft geben. Da der Sinn von etwas zu Lernendem jeweils vor dem Hintergrund der individuellen Erfahrungswelt bestimmt wird, lässt sich dieser nicht verallgemeinern. Generalisierte Motivationsstrategien als Impulsgeber für das Lernen müssen deswegen ins Leere laufen. Doch wie genau lässt sich das Lernen definitorisch beschreiben?

[14] Konkret schreibt er: „Lernen und Staunen ist in den meisten Fällen angenehm, denn im Staunen steckt die Begierde zu lernen, so daß das Bestaunte etwas Begehrtes ist“ (Aristoteles 2007, S. 56). Damit beschreibt er das Phänomen, dass Lernen ein lusterzeugender Prozess ist, bei dem sich Wissen und Können angeeignet wird. Lernen dient aber auch dem Ziel der Unlustvermeidung in unangenehmen Situationen. Dies wird auch als *defensives Lernen* bezeichnet.

Exemplarisch werden im Folgenden einzelne Definitionen mit den zentralen Begriffen aufgezeigt, welche für den Lernbegriff dieser Arbeit ausschlaggebend sind. Eine umfassende Darstellung des pädagogischen Grundbegriffes kann im Rahmen dieser Arbeit nicht geleistet werden.

> „Lernen bezeichnet die Veränderungen von Selbst- und Weltverhältnissen sowie von Verhältnissen zu anderen, die nicht aufgrund von angeborenen Dispositionen, sondern aufgrund von zumindest basal reflektierten Erfahrungen erfolgen und die als dementsprechend begründbare Veränderungen von Handlungs- und Verhaltensmöglichkeiten, von Deutungs- und Interpretationsmuster und von Geschmacks- und Wertstrukturen vom Lernenden in seiner leiblichen Gesamtheit erlebbar sind; kurz gesagt: Lernen ist eine erfahrungsreflexive, auf den Lernenden sich auswirkende Gewinnung von spezifischem Wissen und Können.“ (Göhlich & Zirfas 2007, S. 17)

Lernen ist also das Reflektieren über gemachte Erfahrungen und verändert das Verhältnis des Lernenden zu sich selbst, wie auch zur Umgebung und manifestiert sich in neuem Wissen und Können. In Abgrenzung zum Bildungsbegriff sehen die Autoren Lernen als kleinteiligeren Veränderungsprozess, der eher die subjektive Erfahrung im Blick hat, statt sich in einen kulturellen Gesamtrahmen mit Autonomie als normativem Bildungsziel, einzuordnen (vgl. ebd., S. 15).

Auf diesen wiederum zielt die kritisch-pragmatische Lerntheorie von Peter Faulstich ab. Im Anschluss an die Kritische Theorie beschreibt er Lernen als Möglichkeitserweiterung, die die Faktizität des Vorgegebenen überschreitet, um die Potentiale des *guten Lebens* auszuschöpfen. Lernen ist in Sinn und Situiertheit eingebunden und führt zu einer Veränderung des Individuums. Lernende verändern sich selbst, ihre Identität und den Möglichkeitsraum ihrer Handlungen. Als eine Kritische Theorie, nutzt Faulstich einen Freiheitsbegriff, der nicht losgelöst von gesellschaftlichen Lebensbedingungen betrachtet werden kann. Kritisch heißt für ihn, die Perspektive auf das Mögliche zu richten, zu dem Lernen befähigt (vgl. Faulstich 2013, S. 88 - 95).

3.1.3 Dimensionen des Lernens

Wie in den Ausführungen von Faulstich bereits anklang, ist Lernen ein Prozess, der individuell verschieden abläuft. Er ist eingebettet in Rahmenbedingungen, die von der lernenden Person selbst und der Lernsituation abhängen. Die Wichtigsten mit Bezug auf selbstgesteuertes Lernen sollen hier kurz beleuchtet werden. Dabei handelt es sich um Interesse und Situiertheit.

Der Vollständigkeit halber muss hier noch erwähnt werden, dass auch ohne Interesse und Lustgewinn gelernt werden kann. Defensives Lernen ist nicht auf den intendierten Erwerb von Fähigkeiten und Fertigkeiten ausgerichtet, sondern auf die Vermeidung von Strafen oder anderen negativen Erfahrungen.

Interesse

Erst zu Beginn des 20. Jahrhunderts fand eine systematische pädagogische Forschung bezüglich des Einflusses von Interesse beim Lernen statt. Die bis dahin genutzten psychologischen Motivationstheorien wiesen aus pädagogischer Sicht erhebliche Mängel auf, da sie der Gegenstandsspezifität und dem Inhaltsbezug beim Lernen nicht ausreichend Rechnung trugen. Andreas Krapp beschrieb Interesse als wichtige, empirisch fundierte Bedingungsvariable für Lernen (vgl. Krapp 1992a, S. 9ff):

> „Lernen aus Interesse führt zu vergleichsweise umfangreichen, differenzierten und tief verankerten Wissenstrukturen, die sich unter bestimmten Voraussetzungen auch in entsprechenden Leistungsnachweisen in Schule und Universität bemerkbar machen. Trotz objektiv hoher Anstrengungen erlebt der Lerner die Auseinandersetzung mit dem Gegenstand seines aktuellen Interesses als angenehm. Es fällt ihm leicht, seine Aufmerksamkeit auf den Lernstoff zu konzentrieren und er tendiert stärker als sonst dazu, ganz in der Beschäftigung mit einer Sache aufzugehen und Flow-ähnliche Zustände zu erleben.“ (ebd., S. 41)

Das Interessenkonstrukt von Krapp unterscheidet zwei Ursachen für eine Interessenhandlung: aktualisiertes und situationales Interesse. Aktualisiertes Interesse beschreibt persönliche Interessen, also individuelle Vorlieben für bestimmte Wissens- und Handlungsgebiete. Es ist von ’innen’ veranlasst und resultiert aus früheren Erfahrungen mit dem Interessengegenstand. Wohingegen das situationale Interesse allein durch ’äußere’ Reize entsteht. Durch entsprechende Gestaltung des Handlungskontextes wird ein situationsspezifisches Interesse angeregt, was vorher nicht vorhanden war. Die gezielte Gestaltung von Lernarrangements und Lernmaterialien dienen dem Zweck, situationales Interesse zu wecken (vgl. Krapp 1992b, S. 309).

Situiertheit

Neben dem Punkt, dass selbstgesteuertes Lernen sich an Sinn und Interesse orientiert, gibt es noch einen anderen entscheidenden Faktor. Lernen erfolgt situiert, ist also eingebunden in einen individuellen Kontext. Dies ist zum einen der eigene Körper und zum anderen die vorherigen Erfahrungen, die in Bezug auf den Lerngegenstand schon gemacht wurden. Lernen ist demnach geprägt durch Körperlichkeit und biografische Erlebnisse (vgl. Faulstich 2007, S. 20). Die Möglichkeiten des eigenen Körpers befördern oder verunmöglichen bestimmte Erfahrungen. Ebenso verhält es sich mit dem Vorwissen und der Biografie. Lernschwierigkeiten können das Resultat von unangenehmen Erfahrungen in der Vergangenheit sein.

> „Bei Erwachsenen ist von Anfang an klar, dass sie in vielfältigen Kontexten stehen, Situationen unterschiedlich wahrnehmen und in ihrer Biografie eigene Vergangenheit verarbeiten. Erwachsenenlernen kommt nicht aus ohne Rückbezug auf Lebens-, insbesondere auf Lernerfahrungen aus Kindheit, Schule,

> Arbeitsplatz, Familie usw. [...] Lernen ist immer schon Anschlusslernen an vorher Gelerntes.“ (ebd., S. 20)

Das Konzept von Lernen als situiertem Prozess geht zurück auf Jean Lave und Etienne Wenger. Dieser Ansatz wurde nach der Beobachtung der informellen Handwerksausbildung von Schneider:innen in Liberia entwickelt. Es geht von der Grundannahme aus, dass Lernen ein integraler Bestandteil einer jeden sozialen Praxis ist und nicht unabhängig davon untersucht werden kann. Dies gilt aber nicht nur für das Lernen Erwachsener, denn auch Kinder sind legitime periphere Teilnehmende an der sozialen Welt der Erwachsenen. Hierbei klingt schon die Definition von situiertem Lernen an, die von Lave und Wenger durch die Verknüpfung von zwei Konzepten in ihrem kulturell-historischen Lernbegriff zusammenfassen. Sie charakterisieren Lernen als legitime periphere Partizipation in Praxisgemeinschaften (vgl. Lave & Wenger 1991, S. 30 - 33).

Hierbei deutet sich schon die soziale Einbettung des Lernens an. Aus diesem Grund werden diese beiden Konzepte noch einmal explizit beim Lernen im sozialen Kontext in Kapitel 3.2.5 aufgegriffen. Die Konzepte der legitimen peripheren Partizipation und der Praxisgemeinschaften haben für diese Arbeit eine fundamentale Bedeutung, da sie sich bestens für die Analyse von Lernen in Entwicklungsgemeinschaften der FLOSS-Bewegung eignen.

3.1.4 Lernen als Befähigung

Der Lernbegriff von Göhlich und Zirfas, wie auch der von Faulstich, benennen als Resultat des Lernens eine Erweiterung des Handlungsrepertoires und des Reflexionsvermögens. Diese Definition des Lernbegriffs ist für sich genommen noch recht allgemein formuliert, da die neuen Handlungsmöglichkeiten sich auf alle möglichen Handlungen beziehen können. Faulstich mit seinem kritisch-pragmatischen Lernbegriff engt die Möglichkeitserweiterung durch Lernen jedoch weiter ein:

> „Dies ist der Kern kritischer Theorie, dass sie nicht bestehende ’Tatsachen’ abbildet, sondern nach Möglichkeiten fragt, die als Potentiale ’guten Lebens’ die Faktizität des Vorgegebenen überschreiten. Nicht, wie es ist, ist zu untersuchen, sondern, wie es sein könnte. [...] Lernen ist in Entwürfe des Möglichen in der Zukunft einbezogen: Ich kann etwas nicht, will es aber können; also lerne ich, damit ich es danach kann. Ich frage danach, was anders sein könnte, also nach den Möglichkeiten. Um diese zu ergreifen erweitere ich meine Handlungsspielräume, lerne also. Zugleich besitze ich die Freiheit innerhalb bestehender Verhältnisse zu lernen oder aber nicht zu lernen. Ich lerne in bedingter Freiheit. Unter dem Horizont des Möglichen und Zukünftigen stoßen wir hier unausweichlich auf den wissenschaftlich, politisch und philosophisch hoch belasteten Begriff der Freiheit.“ (Faulstich 2007, S. 91)

Mit dem Hinweis auf das *gute Leben* gibt Faulstich der Erweiterung der Handlungsmöglichkeiten eine normative Richtung vor, ohne diese weiter zu spezifizieren. Durch den Verweis auf die Freiheit und die Möglichkeiten zur Veränderungen, wird die Frage aufgeworfen, wozu Lernen konkret befähigen sollte. Der Fähigkeitenansatz von Martha Nussbaum und Amartya Sen versucht darauf eine Antwort zu geben.

Der Fähigkeitenansatz von Nussbaum

Der von Martha Nussbaum entwickelte Fähigkeitenansatz[15] ist im Kern ein Ansatz, um den Grad menschlicher Entwicklung messen zu können. Die rein ökonomische Betrachtung durch das Bruttoinlandsprodukt von Volkswirtschaften hat viele blinde Flecken. Mit einem solchen Index ist es nicht möglich soziale Ungleichheit, Bildungsungerechtigkeit, Gesundheitszustand der Bevölkerung oder ökologische Probleme abzubilden. Aus diesem Grund wurde von den Vereinten Nationen der *Human Development Index* (HDI) eingeführt. Nussbaums Fähigkeitenansatz hat viel Potential den HDI zu bereichern. Relevant für diese Forschungsarbeit ist aber seine Implikation des Fähigkeitenansatzes für Lernen und Bildung. Deren normative Begründung und die Ausgestaltung von Nussbaums Ansatz wird in den kommenden Abschnitten ausgeführt.

Nussbaum versucht Antworten auf die Frage nach dem *guten Leben* zu geben, eines der Kernfragen der Philosophie des Aristoteles, auf die sich Nussbaum bezieht. Aus der Beantwortung dieser Frage leitet sich die Aufgabe des Staates und seiner Institutionen ab:

> „[...] jedem Bürger[16] die materiellen, institutionellen und pädagogischen Bedingungen zur Verfügung zu stellen, die ihm einen Zugang zum guten menschlichen Leben eröffnen und ihn in die Lage versetzen, sich für ein gutes Leben und Handeln zu entscheiden." (Nussbaum 2012, S. 24)

Doch was genau ist ein gutes Leben? Über diese normative Frage divergieren die Meinungen. Nussbaums essentialistischer Ansatz unterscheidet sich grundlegend von liberalen Ansätzen, auf die sich viele Wohlfahrtssysteme stützen.

Theorie des Guten

Um dies zu bestimmen, entwickelte die Philosophin Nussbaum auf der Grundlage der Theorie des Guten von Aristoteles die *starke vage Konzeption des Guten*. Ihr Kern sind die Grundfähigkeiten von Menschen. Diese zu entwickeln, sei Ziel staatlichen Handelns. Menschen sollen dazu befähigt werden, folgende Fähigkeiten aktiv ausüben zu können (vgl. ebd., S. 57f):

- ein volles Menschenleben zu leben, ohne vorzeitigen Tod beziehungsweise keinen Tod, bevor das Leben nicht mehr lebenswert erscheint

[15] Teilweise wird der Begriff Befähigungsansatz verwendet. Im englischen Original: *Capabilities Approach*.

[16] In der attischen Demokratie konnten nur Männer die Vollbürgerschaft innehaben. Frauen, Fremde und Sklaven waren von der politischen Mitbestimmung ausgeschlossen.

- sich guter Gesundheit zu erfreuen (angemessene Ernährung, Unterkunft, sexuelle Befriedigung, Mobilität)
- unnötigen Schmerz zu vermeiden und freudvolle Erlebnisse zu haben
- Nutzung aller fünf Sinne, Vorstellungsvermögen, Denk- und Urteilsvermögen
- Aufbau von Bindungen zu Dingen und Personen (zu lieben, zu trauern, Sehnsucht und Dankbarkeit empfinden)
- eine Vorstellung vom Guten zu haben (kritische Reflexion der eigenen Lebensplanung)
- verschiedene Formen von familiären und sozialen Bindungen zu führen
- Verbundenheit mit der Natur, pfleglicher Umgang mit Tieren und Pflanzen
- zu lachen, spielen, Freude an erholsamen Tätigkeiten haben
- das eigene Leben zu führen und nicht das von jemand Anderem

Die Konzeption ist stark, weil sie konkrete menschliche Ziele beschreibt, aber dennoch vage, um Raum für individuelle Spezifikationen und unterschiedliche Traditionen zu lassen (vgl. ebd., S. 46). Ziel ist es, Menschen die Möglichkeit zu geben, das für sie gute Leben führen zu können. Dazu bedarf es der praktischen Vernunft für die eigene Lebensplanung, die Verbundenheit mit anderen Menschen und eine Entscheidungsfreiheit, die sich auch auf die politische Konzeption auswirkt (vgl. ebd., S. 59, 65).

Diesen Fähigkeitenkatalog entwickelte Nussbaum in Abgrenzung zur liberalen Theorie der Gerechtigkeit von John Rawls. Seine Theorie basiert auf der Verteilung von Grundgütern statt auf Beschreibung von Fähigkeiten. Rawls Konzeption einer gerechten Gesellschaft baut auf zwei Prinzipien auf (vgl. Rawls 1971, S. 60f):

1. Politische Grundrechte und Freiheiten: Wahl-, Rede-, Versammlungs- und Gedankenfreiheit (zum Beispiel Religionsfreiheit); Schutz des Eigentums; Schutz vor willkürlicher Verhaftung, Unverletzlichkeit der Person
2. Sozioökonomische Verteilung: eine Ungleichverteilung von Reichtum wird nur dann geduldet, wenn sie für eine Mehrheit von Vorteil ist und der Zugang zu Ämtern und Positionen allen offen steht[17]

Seine *Theorie des Guten* bezeichnet er als schwach, weil sie auf die essentiellen Grundgüter beschränkt ist. Gut ist für ihn das, was eine Person für eine vernünftige Lebensführung

[17] In diesem Punkt grenzt sich Rawls von rein utilitaristischen Verteilungsstrategien ab, in dem er im Zweifel der Freiheit (1. Gerechtigkeitsprinzip) Vorrang einräumt (vgl. Rawls 1971, S. 243f). Ansonsten wäre ein Gesellschaft nach utilitaristischen Gesichtspunkten immer noch gerecht, wenn eine Minderheit durch die Mehrheit ausgebeutet werden würden. Diese Ungleichbehandlung würde durch den Gewinn an Lebensqualität von Vielen die Ausbeutung von Wenigen rechtfertigen.

benötigt. Dies schlägt sich in der Anwendung der Gerechtigkeitsprinzipien nieder. Doch was genau für eine Person an Gütern notwendig ist, kann die Person nur allein entscheiden. Aus diesem Grund kann keine allgemeingültige Liste an Gütern erstellt werden (vgl. ebd., S. 395 - 408) beziehungsweise eine solche Liste könnte als Paternalismus interpretiert werden.

Der gegenwärtige Sozialstaat der BRD ist nach dieser liberalen Theorie von Rawls gestaltet. Den Staatsbürger:innen werden politische Grundfreiheiten zugestanden und sie haben zumindest theoretisch die Chance, mittels eines Arbeitseinkommens ihren Entwurf von einem vernünftigen, guten Leben zu leben. Ist ihnen dies aus diversen Gründen nicht möglich, sorgt der Staat mit Transferleistungen dafür, dass zumindest formal die politischen Grundrechte erhalten bleiben. Ob sie allerdings das für sie bestmögliche *gute Leben* führen (können), bleibt dem Staat verborgen.

Hier setzt Nussbaums Kritik an der liberalen Theorie des Guten an. Sie arbeitet heraus, dass starre Sätze von monetären Transferleistungen nicht die unterschiedlichen Ressourcenbedarfe von Menschen abdecken, um eine bestimmte Lebensqualität zu erreichen. Beispielsweise brauchen Kinder aus benachteiligten Minderheitengruppen mehr finanzielle Unterstützung, um den gleichen Zugang zu Bildungsmöglichkeiten zu bekommen als Kinder aus der Mittelschicht (vgl. Nussbaum 2012, S. 36). Weiterhin ist die menschliche Fähigkeit, sich Dinge vorzustellen und zu wünschen, stark von den aktuellen Lebensbedingungen geprägt. Menschen in prekarisierten Lebensverhältnissen verlieren oder entwickeln gar nicht erst die Idee von einem besseren Leben. Mit dem Beispiel von unterprivilegierten Analphabetinnen in Bangladesch schreibt Nussbaum:

> „Manchmal verhindert die Kombination von Unwissenheit und kulturellem Druck tatsächlich, daß überhaupt der Wunsch nach Bildung entsteht; manchmal verhindert sie nur die öffentliche Artikulation dieses Wunsches. In beiden Fällen wird eine nützlichkeitsorientierte Herangehensweise an gesellschaftspolitische Fragen unfähig sein, den Status quo zu kritisieren, wie ungerecht der im Hinblick auf die Bandbreite menschlicher Möglichkeiten auch sein mag.“ (ebd., S. 43)

Die latente Gefahr, dass die Grundfähigkeiten bei Menschen, die aus dem Arbeitsmarkt gedrängt wurden oder in entfremdeten Arbeitsverhältnissen beschäftigt sind, verkümmern, bleibt im liberalen Staatsentwurf unerkannt. Einzig nach einer diagnostizierten psychischen oder physischen Arbeitsunfähigkeit greift das Hilfesystem des Sozialstaates. Der Staat hat damit eine Nachsorgefunktion. Dies aber auch nur dann, wenn überhaupt ein Defizit formell erkannt wurde.

Dieses Problem würde sich mit einem auf den Aufbau von Fähigkeiten orientierten Gesellschaftsentwurf nicht stellen. Der Staat hätte dann die Aufgabe, die Grundfähigkeiten bei Kindern zu entwickeln und bei Erwachsenen zu erhalten und die entsprechenden Umstände zu schaffen, dass die Menschen aktiv ihre Fähigkeiten nutzen können (vgl. ebd., S. 107, 86). Dies weist dem Staat eine aktive Vorsorgefunktion zu. Lebens- und Ar-

beitsbedingungen, die sich negativ auf die Grundfähigkeiten auswirken könnten, müssten präventiv beseitigt werden. Eine konsequente Anwendung des Fähigkeitenansatzes würde demnach zu tiefgreifenden Veränderungen der aktuellen Bildungs- und Arbeitsbedingungen führen.

Probleme des Fähigkeitenansatzes

Wie Nussbaum selbst feststellt, weist der Fähigkeitenansatz bei seiner Anwendung noch erhebliche Fragen auf. Diese ergeben sich aus der mangelnden Operationalisierung der Grundfähigkeiten und den moralischen Konsequenzen. Zum einen sind die Grundfähigkeiten so allgemein formuliert, dass erst geeignete Indizes für deren Messung gefunden werden müssen. Zum anderen fehlt ein Maßstab, ab welchem Fähigkeitenniveau eine Grundfähigkeit als ausreichend entwickelt gilt. Sind Antworten auf diese Fragen gefunden, muss definiert werden, welche Menge und welcher Zugang zu Mitteln als ausreichend gilt.

Bezüglich der moralischen Implikationen muss gefragt werden, welche Verpflichtungen des Staates sich ergeben, wenn die Grundfähigkeiten zum Beispiel durch Alter oder durch eine Form der Beeinträchtigung eingeschränkt sind. Noch weiter zugespitzt stellt Nussbaum sich die Frage, ob Menschen, welche die Grundfähigkeiten nicht ausbilden können, dass Menschsein aberkannt werden könnte (vgl. ebd., S. 117ff).

Fähigkeitenansatz als Zielbestimmung pädagogischen Handelns

Für den Aufbau und den Erhalt der Grundfähigkeiten braucht es zum einen die internen Fähigkeiten durch Ausbildung des Geistes, des Charakters und des Körpers und zum anderen die externen Bedingungen, wie zum Beispiel Abwesenheit von Armut und monotoner Arbeit oder Zugang zu gesunden Nahrungsmitteln. Aufgabe des Staates ist es, bei so vielen Menschen wie möglich den Aufbau der Grundfähigkeiten zu realisieren, statt die zu unterstützen, die diese Schwelle bereits erreicht haben (vgl. ebd., S. 103, 63). Um dies zu erreichen, bedarf es eines tiefgreifenden Umbaus der Art und Weise, wie Bildungsinstitutionen organisiert sind.

Externe Bedingungen

Das Bildungssystem nimmt im Fähigkeitenansatz eine Schlüsselposition ein (vgl. Nussbaum 2006, S. 387). Statt Elitenförderung, müsste auf eine konsequente Breitenförderung gesetzt werden. Das gegliederte Schulsystem müsste zugunsten eines einheitlichen Schulabschlusses umgebaut werden. Der Rahmenlehrplan mit dem Fächerkanon müsste an die neuen Anforderungen angepasst werden und das pädagogische Personal in allen Bildungseinrichtungen massiv aufgestockt werden, um Fähigkeitenentwicklung nicht nur im Klassendurchschnitt, sondern individuell bei jedem Kind beziehungsweise Erwachsenen umsetzen zu können.

Auch das Thema Evaluation von Lernfortschritten bedarf einer umfassenden Neubewertung. Die Art und Weise wie lernförderliche Situationen gestaltet werden können, ge-

hört auf den Prüfstand. Ansätze wie projektorientierter Unterricht und Tandem-Teaching könnten dazu beitragen, Bildungsungerechtigkeit zu reduzieren. Dem Trend im Bildungssystem, Naturwissenschaften und Technologieforschung auf Kosten von Geisteswissenschaften und Kunst zu bevorzugen, sollte Einhalt geboten werden (vgl. Nussbaum 2009, S. 7).

Interne Fähigkeiten

Eine umfassende Feinzielplanung für das Bildungssystem liegt von Nussbaum nicht vor. Allerdings betont sie in Anlehnung an Rabindranath Tagore und John Dewey die Bedeutung der künstlerischen und geisteswissenschaftlichen Unterrichtsfächer für den Bereich Demokratielernen. Die Fächer können kritisches Denk- und Vorstellungsvermögen fördern, welche sich wiederum in folgenden Fähigkeiten manifestieren (vgl. Nussbaum 2006, S. 388ff; Nussbaum 2009, S. 4ff):

- Kritisches Bewusstsein über die eigene Person sowie über Traditionen und Lebensweisen:

  Dies beinhaltet die Fähigkeit zu logischem Denken und der Prüfung von Argumentationen auf Konsistenz und ihren Wahrheitsgehalt. Um daraus eigene Positionen entwickeln zu können, bedarf es eines präzisen Urteilsvermögens.

- Gefühl der Verbundenheit mit anderen Menschen außerhalb der eigenen Region / Gruppe:

  Solidarität gegenüber anderen Gruppen innerhalb und außerhalb der eigenen Region oder des Staates erfordert zum einen Kenntnisse in Bezug auf Geschichte, Religion, Ethnien und Gender. Zum anderen erfordert sie Kenntnisse über gemeinsame Interessen und Bedürfnisse.

- Fähigkeit der Perspektivenübernahme und Einfühlungsvermögen:

  Diese interne Fähigkeit erfordert Kreativität und Vorstellungsvermögen.

Melanie Walker (2006) beschreibt für den Bereich der Hochschulbildung Ansatzpunkte für die Anwendung des Fähigkeitenansatzes. Sie dokumentiert diskriminierende wie auch bestärkende Erfahrungen von Studierenden an Hochschulen. Diskriminierendes Verhalten entsteht seitens des Lehrpersonals und auch durch andere Studierende. Daraus ergibt sich in der Konsequenz, diskriminierende Verhaltensweisen als eine Ursache von Lernschwierigkeiten zu verhindern. Sexismus, Rassismus und Klassismus in Lehr-Lern-Situationen, aber auch in ihrer institutionalisierten Form, stehen der Bildungsgerechtigkeit entgegen. Um die Entwicklung von internen Fähigkeiten zu fördern, muss auf diskriminierungsfreie Lernmöglichkeiten geachtet werden.

Für die Anwendung des Fähigkeitenansatzes in der Hochschulbildung, isoliert Walker

aus den von Nussbaum und anderen erstellten Fähigkeitenkataloge[18] acht pädagogisch relevante Ziele (vgl. ebd., S. 120, 128f):

1. Entscheidungsfähigkeit (begründete, kritisch reflektierte, sozial verantwortliche Entscheidungen treffen; Lebensplanung in einer ungewissen Welt)

2. Intellektuelle Resilienz (Work-Life-Balance; Risiken abschätzen; Ausdauer; Anpassungsfähigkeit; Selbstständigkeit; Hoffnung)

3. Vorstellungsvermögen und Wissen (Wissensaneignung aus Freude; Wissensaneignung mit Ziel der Partizipation im beruflichen und gesellschaftlichen Kontext; Perspektivenübernahme, kritische Bewertungen vornehmen können; komplexe Sachverhalte darstellen; Bewusstsein über Problemfragen in Ethik und Moral; Verständnis von Wissenschaft und Technologie im öffentlichen Diskurs)

4. Lerndispositionen (Neugier erleben und Begierde nach Lernen haben; Selbstvertrauen in eigene Lernfähigkeiten haben)

5. Soziale Beziehungen und Netzwerke (gemeinschaftliches Lösen von Problemen; effektives Arbeiten in Gruppen; freundschaftliche Beziehungen für Lernunterstützung und gemeinsame Freizeit aufbauen können; gegenseitiges Vertrauen)

6. Würde, Respekt, Anerkennung (Schutz vor psychischer Gewalt und Entfremdung, Aufbau von Selbstvertrauen und Hoffnung)

7. Emotionale Integrität (Abwesenheit von Angst und Sorgen, Empathiefähigkeit, Emotionen entwickeln für Vorstellungskraft, Bewusstsein und Einsichten)

8. Körperliche Integrität (Schutz vor allen Formen von körperlichen und verbalen Übergriffen beziehungsweise Belästigungen)

Die Bedeutung des Fähigkeitenansatzes für diese Arbeit

Das Erkenntnisinteresse dieser Forschungsarbeit zielt auf den Aufbau von Kompetenzen seitens der Entwickler:innen durch ihr Engagement innerhalb von FLOSS-Projekten ab. In Abhängigkeit von dem Umfang des Engagements ist davon auszugehen, dass diverse Grundfähigkeiten aus Nussbaums Katalog bei der gemeinschaftlichen Entwicklungsarbeit erworben oder ausgebaut werden. Dies betrifft vor allem das Erleben von freudvollen Erlebnissen, erholsame Tätigkeiten ausüben, soziale Bindungen ausbauen und das Vorstellungs-, Denk- und Urteilsvermögen zu verbessern.

[18] Einer der von Walker genutzten Fähigkeitenkataloge ist der von Narayan und Petesch, der anhand einer international durchgeführten Studie der Weltbank über Armut erstellt wurde. Er benennt neben Fähigkeiten auch materielle Güter als Voraussetzung für ein gutes Leben. Folgende Kategorien wurden erstellt: Gesundheit, körperliche und physische Integrität, Würde und Respekt, soziale Zugehörigkeit, kulturelle Identität, Vorstellungs- und Urteilsvermögen, Bildung, Fähigkeit sich und Andere zu organisieren beziehungsweise Partizipation in Organisationen, politische Repräsentation und die Fähigkeit, die politische Führung zur Rechenschaft zu ziehen. Als materielle Voraussetzung werden Arbeitsplatz, Produktionsmittel oder finanzielle Rücklagen genannt (vgl. Narayan & Petesch 2002, S. 463).

Der von Walker erstellte Katalog ist in weiten Teilen dazu geeignet, auch auf die Prozesse in den Praxisgemeinschaften von FLOSS-Projekten übertragen zu werden. Anhand der Analyse der Kommunikation und durch die Befragung der Entwickler:innen wird geprüft, welche Kompetenzen dadurch beeinflusst werden. Damit dient der Fähigkeitenansatz als Analyseraster, um die vielschichtigen Handlungen und sozialen Prozesse bei der gemeinschaftlichen FLOSS-Entwicklung in Erfahrung zu bringen.

3.1.5 Befähigung und Kompetenzaufbau

Zur Anwendung des Fähigkeitenansatzes für den erziehungswissenschaftlichen Kontext soll das Kompetenzkonstrukt herangezogen werden. Denn, wie es bereits Walker mit ihrem Katalog von Zielen versuchte, bedarf die Konzeption eines *guten Lebens* in irgendeiner Form eine Operationalisierung, um empirisch erfasst werden zu können.

Der operationale Lernzielansatz führte jedoch in der Vergangenheit zu einer Vielzahl scheinbar beliebiger Lernzielkataloge mit erwünschten Verhaltensweisen. Sie waren in ihrer Begründung fragwürdig und boten keine Orientierung. Im Gegensatz dazu ermöglicht das Konstrukt von Kompetenzen die Beschreibung von kohärenten Fähigkeitenbündeln, die sich auch in Bezug auf spezifische Lernwege voneinander abgrenzen lassen (vgl. Nieke 2012, S. 2f). Heinrich Roth führte 1971 eine Taxonomie von eigenständig begründbaren Kompetenzbereichen ein. Er unterschied Selbst-, Sach- und Sozialkompetenz.

> „Mündigkeit, wie sie von uns verstanden wird, ist als Kompetenz zu interpretieren, und zwar in einem dreifachen Sinne: a) als Selbstkompetenz (self competence), d.h. als Fähigkeit, für sich selbst verantwortlich handeln zu können, b) als Sachkompetenz, d.h. als Fähigkeit, für Sachbereiche urteils- und handlungsfähig und damit zuständig sein zu können, und c) als Sozialkompetenz, d.h. als Fähigkeit, für sozial, gesellschaftlich und politisch relevante Sach- oder Sozialbereiche urteils- und handlungsfähig und also ebenfalls zuständig sein zu können.“ (Roth 1971, S. 180)

Mündigkeit als Ziel von Erziehung und Entwicklung schlägt sich in verantwortliche Handlungsfähigkeit nieder. Dabei sind die einzelnen Kompetenzen aber nicht voneinander losgelöst zu betrachten. Selbstkompetenz für die moralisch-mündige Handlungsfähigkeit kann beispielsweise nicht ohne Sach- und Sozialkompetenz entwickelt werden (vgl. ebd., S. 388f).

Nieke erweiterte dieses Modell um zwei weitere Kompetenzbereiche. Die Sprachkompetenz dient als Basis für alle anderen Kompetenzen. Sämtliche Kognitionen und Orientierungsstrukturen ausdifferenzierter Kulturen können sprachlich repräsentiert werden und sind durch Enkulturation und Bildung gelernt. Das Erlernen von Sprachen ist damit ein eigenständiger Kompetenzbereich, ohne den der Kompetenzaufbau in anderen Kompetenzbereichen nicht möglich ist. Als weiteren Kompetenzbereich führte er die Leibkompetenz ein. Diese zielt auf den Anteil von bewusst gestaltbaren Körperfunktionen

wie Gestik, Mimik, Stimmmodulation und die Körperbeherrschung. Im Zuge dieser weiterentwickelten Theorie baut die Selbstkompetenz auf der Sprachkompetenz auf. Diese ist wiederum die Basis für die Sach-, Leib- und Sozialkompetenz, die in ihrer Gesamtheit zur Handlungskompetenz zur Aufgabenbearbeitung und Werterealisierung konvergieren (vgl. Nieke 2012, S. 3ff). Im Rahmen dieser Forschungsarbeit soll diese Kompetenztheorie als Ordnungssystem für die Betrachtung des Kompetenzaufbaus von Entwickler:innen der FLOSS-Bewegung dienen. Insbesondere im Auswertungskapitel über die Fähigkeiten und Fertigkeiten, welche die Entwickler:innen durch ihr Engagement in der FLOSS-Entwicklung erworben haben, erfolgt die Präsentation der Erkenntnisse entsprechend der Kompetenzbereiche.

Die gesellschaftlichen Implikationen des Fähigkeitenansatzes sind jedoch viel weitreichender. In dieser Arbeit soll mitnichten der Eindruck vermittelt werden, dass ein subjektivierter Kompetenzaufbau dem gerechtigkeitstheoretischen Anspruch von Nussbaum in Gänze entsprechen könnte. Denn „Befähigungen oder Realfreiheiten lassen sich nicht auf individuelle Eigenschaften, Dispositionen oder Kompetenzen reduzieren, sondern verweisen auf das komplexe Zusammenspiel von Infrastrukturen, Ressourcen, Berechtigungen und Befähigungen“ (Ziegler 2018, S. 80). In diesem Sinne kann eine empirische Betrachtung des Kompetenzaufbaus innerhalb der FLOSS-Bewegung nur einen Teil der gesellschaftlichen Implikationen abdecken, die sich aus einer kritischen Bildungstheorie nach dem Fähigkeitenansatz ergeben.

Zudem lässt sich der Fähigkeitenansatz in der Erziehungswissenschaft als „Basis für eine empirisch fundierbare grundlagentheoretische wie praktisch-evaluative Neuorientierung“ (Oelkers et al. 2010, S. 85) verstehen, in einer dominierenden empirischen Bildungsforschung, die einem verkürzenden Humankapitalansatz verhaftet ist[19]. Diese zielt mit ihren Evaluationstudien hauptsächlich auf *Employability*, im Sinne von Beschäftigungsfähigkeit, statt die Bedingungsfaktoren für soziale Ungleichheit mit einzubeziehen. Durch den Fokus auf Verwirklichungschancen werden Wissensaneignung und Bildung nicht nur mit Blick auf ökonomisches Wachstum und gesellschaftlicher Innovationsfähigkeit betrachtet. Der Fähigkeitenansatz bietet die Möglichkeit die reale Autonomie von Individuen empirisch zu fassen und das Spektrum von effektiv realisierbaren und voneinander unterscheidbaren Handlungsmöglichkeiten zu beschreiben, derer sich die Akteur:innen für ihre individuelle Form eines guten Lebens tatsächlich bedienen können (vgl. ebd., S. 85ff).

3.1.6 Lernformen in der FLOSS-Entwicklung

Wenn die Arbeit an FLOSS-Projekten durch die Entwickler:innen in deren Freizeit erfolgt, sind die dabei stattfindenden Lernprozesse außerhalb formaler Lernorte anzusiedeln. Doch wie genau kann der Aufbau von Kompetenzen in solchen informellen, internetbasierten Netzwerken charakterisiert werden?

[19] Dies zeigt sich beispielsweise bei den PISA-Schulleistungsuntersuchungen der OECD.

Informelles Lernen

Günther Dohmen zeigt die Schwierigkeiten der begrifflichen Bestimmung von informellem Lernen auf, wenn es nur eindimensional als Lernen außerhalb institutionalisierter Lernarrangements wie (Berufs-)Schule, Universität oder betrieblicher Weiterbildung gesehen wird. Dadurch wird diese Lernform zu einer reinen Sammelbezeichnung für nicht näher spezifizierte Lernformen und Lernorte (vgl. Dohmen 2018, S. 56). Erschwerend kommt hinzu, dass alle Lernprozesse auch informelle Anteile in unterschiedlicher Gewichtung haben. Also auch im Rahmen von formal organisierten Lernangeboten mit definierten Lernzielen und aufbereiteten Lernmaterialien wird informell gelernt (vgl. Overwien 2016, S. 400). Die exakte Definition von informellem Lernen gestaltet sich also schwierig. Konsens scheint nur dahingehend zu herrschen, was es nicht ist.

> „1. Es handelt sich um »Lernen«, das heißt um das Verständnis suchende, konstruktive Verarbeiten von Informationen und Erfahrungen. Das schließt zum Beispiel unbewusste Sozialisationsvorgänge aus.
> 2. Es geht um »informelle« Prozesse, d.h. um Lernprozesse, die sich mehr spontan und »natürlich« außerhalb fremdorganisiert-planmäßiger Lehr/Lernveranstaltungen vollziehen. Das schließt hier besonders das Lernen im Rahmen traditioneller schulischer Unterrichtsformen aus.“ (Dohmen 2018, S. 53)

In Abgrenzung zum formalen Lernen wird teilweise auch das non-formale Lernen unterschieden. Es beschreibt ebenfalls Lernen außerhalb von Bildungseinrichtungen, ist aber im Gegensatz zu informellen Lernen zielgerichtet. Es handelt sich also um intentionales Lernen außerhalb formal organisierter Lernangebote. Informelles Lernen erfolgt unstrukturiert und ist inzidentell, also zufällig (vgl. Europäische Kommission 2001, S. 33ff). Da aber der Übergang zwischen diesen Lernformen fließend ist, verzichten viele Autor:innen auf diese Unterscheidung.

Käte Meyer-Drawe und Christian Grabau weisen zudem darauf hin, dass es auch große Schnittmengen zu den Konzepten vom lebenslangen Lernen oder selbstgesteuerten Lernen gibt (vgl. Meyer-Drawe & Grabau 2018, S. 62). Dies arbeitete bereits schon Dohmen heraus. Er gliedert dies jedoch strukturierter auf und beschreibt Ausprägungen von informellem Lernen beim impliziten Lernen, Erfahrungslernen, Alltagslernen und kompetenzentwickelndem Lernen (vgl. Dohmen 2001, S. 27 - 49).

Bei der Bestimmung des Gegenstands von informellem Lernen verweist er auf eine wichtige Unterscheidung zum formalem Lernen:

> „Informelles Lernen ist ein instrumentelles Lernen, ein Mittel zum Zweck. Der Zweck ist [...] nicht das Lernen selbst, sondern die bessere Lösung einer außerschulischen Aufgabe, einer Situationsanforderung, eines Lebensproblems mit Hilfe des Lernens“ (ebd., S. 19)

Es geht also nicht darum, etwas Bestimmtes zu lernen, sondern sich selbst in die Lage zu versetzen, ein bestimmtes Ziel zu erreichen. Das heißt, dass „das informelle Lernen

sich meist im Zusammenhang mit anderen Tätigkeiten und anderen Zielsetzungen als sinnvolle und notwendige Hilfe zum besseren Zurechtkommen in der Umwelt ergibt“ (ebd., S. 23). Informelles Lernen ist damit eine Lösungsstrategie, um eine gewünschte Zustandsänderung zu erreichen.

Informelles Lernen und gemeinnützige Tätigkeit

Der Fokus dieser Arbeit gilt dem informellen Lernen im Rahmen von gemeinnütziger Tätigkeit weil der Zugang zu den Produkten der FLOSS-Entwicklung zumindest theoretisch[20] allen Menschen möglich ist. Mit Blick auf die Lernerfahrungen ist dieser Bereich im deutschsprachigen[21] Kontext bislang nur wenig beforscht. Eine der wenigen Studien über den Kompetenzerwerb bei Jugendlichen als Zielgruppe liefern beispielsweise Wiebken Düx et al. (2008) oder Heinz Reinders (2014). Dabei legen die Autor:innen ihren Schwerpunkt auf institutionalisierte Strukturen, in denen Kinder und Jugendliche freiwillig Verantwortung für andere übernehmen (vgl. Düx et al. 2008, S. 12). Dies schließt aber einen bedeutenden Anteil an gemeinnützigem Engagement aus, der bei der reinen Betrachtung von Jugendverbänden und anderen organisierten Trägern außer Acht bleibt. Statt den Blick auf die Strukturen zu richten, wählt Reinders einen ressourcenorientierten Zugang zu gemeinnütziger Tätigkeit: „Gemeinnützige Tätigkeit hat den Handlungsanreiz, eigene Ressourcen zur Verfügung zu stellen, um den Zustand einer Person, einer Gruppe oder eines Objektes zu verbessern“ (Reinders 2014, S. 19). Schwerpunkt bildet hier die Verteilung von unterschiedlichen Kapitalarten, statt den Blick auf standardisierten Handlungen in Organisationen zu richten. Die Nähe zwischen informellem Lernen und gemeinnütziger Tätigkeit zeigt er durch drei Gemeinsamkeiten auf: Freizeit, Selbststeuerung und individuelle Interessen (vgl. Reinders 2018, S. 442).

Um die Lernprozesse durch die gemeinnützige Tätigkeit zu erklären, bietet Reinders zwei unterschiedliche Modelle. Zum einen ist da der Reflexionsprozess im Rahmen von gemeinnütziger Tätigkeit zu nennen: Die Lernenden erleben durch ihre Erfahrungen eine Diskrepanz zwischen Gegebenheiten der Lebensrealität und theoretischen Überlegungen. Dies stößt einen Reflexionsprozess an, der zu einem neuen Theorie-Praxis-Verständnis führt (vgl. ebd., S. 445). Dies geht auf das Konzept vom Erfahrungslernen von John Dewey und Kurt Lewin sowie auf das Entwicklungsmodell von Jean Piaget zurück (vgl. Kolb 1984, S. 21ff). Darauf aufbauend wurde das Fünf-Phasen-Modell des Reflexionsprozesses abgeleitet (vgl. Reinders 2016, S. 42):

1. Erinnerung an die Erfahrung: Was habe ich erlebt?

[20] Der Zugang ist nur theoretischer Natur, weil die Nutzung von FLOSS natürlich an den Zugang zu bestimmter Hardware, dem Vorhandensein spezifischer Kenntnisse und teilweise auch an den Zugang zum Internet gekoppelt ist.

[21] In den USA ist oftmals gemeinnützige Tätigkeit als *Service Learning* fest in das Curriculum am College integriert. Als Unterrichtsmethode wird sie im Rahmen der Demokratiepädagogik eingesetzt (vgl. Coelen et al. 2016, S. 336f). Aus diesem Grund ist dieser Bereich besser beforscht. Allerdings stellt sich die Frage, inwieweit dies schon eine Grauzone vom informellen Lernen darstellt, da die Student:innen nur bedingt selbstgesteuert lernen, denn die Teilnahme ist teilweise obligatorisch.

2. Bewusstmachung von Kognitionen und Emotionen: Was habe ich in der Situation gedacht und gefühlt?

3. Theoretische Kontextualisierung: Wie lässt sich die Situation theoretisch erklären?

4. Theoriegeleitete Bewertung der Situation: Wie bewerte ich die Situation nach der theoretischen Einordnung?

5. Eröffnung von Handlungsoptionen: Was kann ich in neuen Situationen anders machen?

Das Phasenmodell der Reflexion muss nicht immer in vollem Bewusstsein durchlaufen werden. Oftmals handelt es sich dabei um einen un- oder halbbewussten Prozess. Dieser wird insbesondere dann initiiert, wenn die Diskrepanz zwischen den gemachten Erfahrungen und den eigenen Erwartungen besonders groß ist (vgl. Reinders 2018, S. 447).

Zum anderen gibt es das Modell zur Wirkung von gemeinnütziger Tätigkeit auf die dort handelnden Personen. Aus den Resultaten von verschiedene empirische Studien mit Jugendlichen (Reinders 2014 oder Reinders & Youniss 2006) wurde die *Theorie Gemeinnütziger Tätigkeit* entwickelt. Nach den Studien führten diese zu messbaren Veränderungen bei den Teilnehmenden, auf deren Grundlage diese Theorie immer weiter ausdifferenziert wurde. Die Theorie besagt, dass sich gemeinnütziges Handeln entlang von zwei Pfaden auswirkt: Auf dem kognitiven Pfad kommt es kurzfristig zu einer Veränderung des Selbstbildes im Rahmen der Persönlichkeitsentwicklung. Dies führt langfristig zu Reflexion über soziale Stereotype. Die Jugendlichen sehen soziale Randgruppen nicht mehr als gleichförmig an, sondern erkennen hinter den Stereotypen differenzierte individuelle Biografien. Auf dem behavioralen Pfad kommt es durch das gemeinnützige Handeln zum Erleben von sozialer Handlungswirksamkeit. Dies stärkt langfristig die Bereitschaft von zukünftigen prosozialen Handlungen und die Beteiligung an politischen Mitbestimmungsprozessen (vgl. Reinders 2018, S. 448).

3.1.7 Individuelles Lernen in FLOSS-Projekten

Zusammenfassend kann festgehalten werden, dass sich das Lernen von der reinen Außenbetrachtung hin zum subjektorientierten Lernen entwickelt hat. Freiwilliges Lernen erfolgt entlang der Interessen von Lernenden und ist situiert. Das bedeutet, Lernen hängt von Vorbedingungen ab, die die Lernenden selbst mitbringen. Dies sind zum Beispiel biographische Erlebnisse, Vorwissen oder körperliche Wahrnehmungs- und Handlungmöglichkeiten. Jedes Lernen ist eingebettet in eine soziale Praxis, die kulturell geprägt ist.

Lernen dient der Möglichkeitserweiterung durch den Aufbau von Wissen und Können. Damit ist eine Vorstellung von einer Zukunft verbunden, in der jemand etwas tun oder über etwas urteilen kann, was der Person vorher nicht möglich war. In der kritisch-pragmatischen Bestimmung sieht Faulstich im Lernen die Möglichkeit ein *gutes Leben*

führen zu können. Im Fähigkeitenansatz führt Nussbaum im Rückgriff auf Aristoteles aus, was die Ziele für ein gutes Leben sind. Dies ist ein Katalog von normativen Zielen, die durch eine Kombination von externen Bedingungen und individuellen Fähigkeiten erreicht werden sollen. Um diese in formalen Lernumgebungen umzusetzen, hat Walker daraus einen genaueren Lernzielkatalog für die Hochschulbildung abgeleitet.

Nach diesen Ausführungen soll der Erwerb von Fähigkeiten und Fertigkeiten im Sinne des guten Lebens auch auf seine Anwendung in Bezug auf internetgestützte, informelle Lernumgebungen geprüft werden. Durch die Datenanlyse werde ich zeigen, welche Kompetenzen Entwickler:innen bei ihrer Entwicklungsarbeit aufbauen. Im Unterschied zu formalen Lernumgebungen, für die Walker ihren Fähigkeitenkatalog erstellt hat, handelt es sich bei Entwicklungsgemeinschaften der FLOSS-Bewegung um informelle und non-formale Lernumgebungen. Dies bedeutet, dass die Entwickler:innen zum einen instrumentell lernen, also Lernen ein Mittel ist, um einen bestimmtes Ziel zu erreichen. Dies kann auch als problembasiertes Lernen bezeichnet werden, denn die Problemlösung stellt das Lernziel dar. Zum anderen handelt es sich um inzidentelles Lernen. Es werden zufällig Dinge gelernt, die ursprünglich nicht intendiert waren.

3.2 Lernen im sozialen Kontext

Nach der Erarbeitung eines Lernbegriffs der die individuellen Prozesse im Fokus hat, fehlt die komplementäre Perspektive auf die soziale Dimension vom Lernen. Untersuchungsgegenstand dieser Arbeit sind Individuen in Lerngruppen, die sich nicht explizit als Lerngruppe verstehen. Sie sind entweder gar nicht oder nur sehr lose in einen formal organisatorischen Rahmen integriert. Dennoch teilen sie ein Ziel, eine Gruppenidentität und nutzen gemeinsame Werkzeuge und Informationsquellen. Solche Lerngruppen können beispielsweise in projektorientierten Arbeitsgemeinschaften entstehen, wie sie bei politischen Kampagnen in sozialen Bewegungen anzutreffen sind oder sich informell aufgrund gemeinsamer Interessen in beruflichen Kontexten bilden.

Eine Lerntheorie muss sensibel für die Austauschprozesse zwischen den Lernenden sein, um wichtige Rahmenbedingungen für das Lernen des Individuums beschreiben zu können. Nur damit kann sich eine Lerntheorie dem Anspruch auf Vollständigkeit annähern.

Zu diesem Zweck werden im Folgenden die wichtigsten Lerntheorien vorgestellt und auf ihre Tauglichkeit in Bezug auf das Lernen in sozialen Kontexten überprüft. Dies sind der Behaviorismus, der Kognitivismus, der Konstruktivismus und die neurobiologischen Lerntheorien. Letztere orientieren sich jedoch an Prinzipien der etablierten Lerntheorien und sind streng genommen nur eine Erweiterung bekannter Konzepte. Auch die netzwerkorientierten Lerntheorien können als Erweiterung sozialkonstruktivistischer Lerntheorien aufgefasst werden. Dabei wird ein besonderer Fokus auf die *Communities of Practice* gelegt, da sie wichtige Impulse für das Lernen in Netzwerken liefern .

3.2.1 Behaviorismus

Der klassische Behaviorismus als Lerntheorie reduziert Lernen auf ein von außen beobachtbares Verhalten, welches auf einen Reiz folgt. Das lernende Individuum ist eine *black box*, dessen innere Prozesse als nicht relevant angesehen werden, weil diese einer Außenbeobachtung nicht zugänglich sind. Lernen wird als Reiz-Reaktions-Lernen definiert, wobei auf einen Reiz aus der Umwelt mit einem neuen Verhalten reagiert wird. Das Konzept des Behaviorismus wurde erstmalig durch John B. Watson beschrieben. Er definierte 1913 die Psychologie als objektive Naturwissenschaft, deren Ziel die Vorhersage und Kontrolle von Verhalten ist. Das individuell verschiedene Denken und Erleben seien dabei für den Erkenntnisprozess nicht von wissenschaftlichem Wert (vgl. Watson 1913, Abs. 1).

Als Wegbereiter dieser Lerntheorie gilt Iwan P. Pawlow, der zu Beginn des 20. Jahrhunderts die Veränderung bedingter Reflexe bei Hunden mittels der Konditionierung beschrieb. Bei seinen Experimenten mit Hunden konditionierte er diese, auf einen zuvor neutralen Reiz mit Speichelfluss zu reagieren, der zuvor keine spezifische Reaktion bei den Tieren auslöste. Diesen Ansatz führte später Burrhaus F. Skinner fort und prägte den Begriff der Operanten Konditionierung. Er beschränkte sich im Gegensatz zu Pawlow nicht nur auf die Veränderung von einem vorhandenen Verhaltensrepertoire. Mittels gezielter Reaktion auf bestimmtes Verhalten, zum Beispiel durch Futterbelohnung, wurden neue Verhaltensmuster bei Versuchstieren erzeugt (vgl. Faulstich 2013, S. 38ff). Dies beschreibt er wie Folgt:

> „Wir sorgen dafür, daß Wirkungen im Sinne von positiven Nacheffekten tatsächlich eintreten und unter solchen Bedingungen eintreten, die besonders günstig sind für die Verhaltensänderungen, die wir „lernen“ nennen. Wenn wir erst einmal die besondere Art einer Folgeerscheinung, die wir Verstärkung nennen, hergestellt haben, erlauben es unsere Methoden, das Verhalten eines Organismus fast beliebig zu formen.“ (Skinner 1971, S. 17)

Im Verlauf seiner Forschung dehnte Skinner mit seinen Kolleg:innen die Experimente auf diverse Tierarten und Menschen aus. Dabei stellte er für die Übertragbarkeit dieser Methodik Folgendes fest:

> „Bei all diesen Arbeiten war die Gattung des Organismus überraschend unwesentlich. Zwar waren die untersuchten Lebewesen alle Wirbeltiere aber dennoch von sehr verschiedener Art. Vergleichbare Ergebnisse erzielten wir mit Tauben, Ratte, Hunden, Affen, Kindern und in letzter Zeit auch mit psychisch Erkrankten [...]. Trotz großer phylogenetischer Unterschiede zeigt der Lernprozeß dieser Lebewesen erstaunlich ähnliche Eigenschaften.“ (ebd., S. 20)

Wesentlich für den Lernprozess sind hiernach die Wahl der geeigneten Verstärker, um ein erwünschtes Verhalten zu formen. Für die Übertragung dieser Prinzipien auf den Schulun-

terricht hat Skinner diverse Möglichkeiten für Verstärkungen vorgestellt. Dies können beispielsweise automatische Verstärker sein, etwa durch besondere Eigenschaften des Lerngegenstandes, der zu einer Beschäftigung mit ihm animiert oder durch Belohnungen in der Form, dass Kinder im direkten Anschluss an das Zeigen des gewünschten Verhaltens eine für sie als freudvoll erlebten Tätigkeit nachgehen können. Eine andere Form der positiven Verstärkung kann auch durch die Zuneigung der Lehrkraft erreicht werden.[22] Daneben gibt es auch Formen der negativen Verstärkung. Die gewünschte Verhaltensänderung wird dadurch erreicht, dass Kinder damit eine Form der Bestrafung vermeiden können (vgl. ebd., S. 25).

Einfluss anderer Personen im Behaviorismus

Die Rolle von anderen Personen spielt bei psychologischen Lerntheorien, die sich am Behaviorismus orientieren, keine besondere Bedeutung. Diese fungieren lediglich als Quelle von Reizen, wozu potentiell alle Elemente in der unmittelbaren Umgebung von Lernenden geeignet sind. Der Lernbegriff ist mit Fokus auf Verhaltensänderung stark generalisiert und vereinfacht. Lernen wird als eine von außen initiierte und dadurch fremdgesteuerte Tätigkeit angesehen. Die komplexen inneren Prozesse von Lernenden werden reduziert oder ganz ausgeblendet.

3.2.2 Kognitivismus

Die Lerntheorie des Kognitvismus bricht das starre Reiz-Reaktions-Lernen auf und öffnet sich für innere Prozesse, die bei den Lernenden stattfinden. Die Prozesse, die zwischen Reiz und Reaktion stattfinden, werden schrittweise unter die Lupe genommen und ausdifferenziert. Dieser intrapsychischen Bereich, der zuvor als *black box* keine Relevanz hatte, wird mit Prozessen der Informationaufnahme, Informationsspeicherung und Informationsverarbeitung charakterisiert. Konkret werden folgende Erweiterungen gemacht (vgl. Faulstich 2013, S. 42f):

- Informationsaufnahme: Ständig ist unser Körper mit Reizen aus der Umwelt konfrontiert. Nicht jeder Reiz führt zu einer Reaktion. Schon bei der Aufnahme von Reizen durch unsere Sinnesorgane wird nach wichtigen und unwichtigen Reizen unterschieden. Die Auswahl der Reize, die von uns als wichtig eingestuft werden, ist auch von inneren Prozessen gesteuert und veränderlich.

- Informationsspeicherung: Ohne Erinnerung an zuvor Gelerntes ist kein aufbauendes Lernen möglich. Das Gedächtnis sammelt Erfahrungen, wodurch Wissensbestände aufgebaut und Bewertungen ermöglicht werden.

[22] Da der zeitliche Zusammenhang zwischen gewünschter Reaktion und Verstärkung wichtig ist, sieht Skinner die Kapazitäten einer einzelnen Lehrkraft in der Klasse als nicht ausreichend an. Stattdessen empfiehlt er den Unterricht mit Lehrmaschinen zu ergänzen, die den Schüler:innen bei einer richtigen Antwort automatisch belohnen (vgl. Skinner 1971, S. 27).

- Informationsverarbeitung: Die Informationen in unserem Gedächtnis unterliegen einer ständigen Reorganisation. Durch unsere Gefühle werden diese nach Relevanz geordnet und die Handlungsmöglichkeiten bewertet.

Die begriffliche Analogie zur Informationsverarbeitung im Kontext der Informatik kommt nicht zufällig. Verschiedene Entwicklungen in der Wissenschaft führten zur kognitiven Wende, die in vielen Bereichen zu einem umfassenden Umdenken führten. Howard Gardner beschrieb fünf maßgeblich Einflüsse, aus denen später die interdisziplinäre Kognitionswissenschaft hervorging: (1) die Mathematik und Informatik[23], (2) das neuronale Netzwerkmodell[24], (3) die Kybernetik[25], (4) die Informationstheorie[26] und (5) neurophysiologische Pathologien[27] (vgl. Gardner 1985, S. 16 - 23).

Auch in der Psychologie hielt die kognitive Wende Einzug. Wichtige Erweiterungen auf dem Gebiet der behavioristischen Lerntheorie hin zum Kognitivismus erarbeitete zum Beispiel durch Albert Bandura. Seine Lerntheorie baut auf Lernen durch Verstärkung auf. Werden die differenzierten Konsequenzen einer Handlung als positiv erlebt, werden diese Handlungen beibehalten, andernfalls verworfen. Diese Bewertung der Konsequenzen erfolgt allerdings entlang der kognitiven Fähigkeiten. Lernen ist damit kein unbewusster, mechanistischer Vorgang, wie es im radikalen Behaviorismus postuliert wurde. Vielmehr haben die Konsequenzen eine wichtige Funktion für das kognitive Lernen.

Dies ist zum einen der Informationsgehalt einer Handlungskonsequenz. Durch Beobachtung der Konsequenzen durch die Lernenden bauen diese Hypothesen über ihre Umwelt auf. Sie lernen, welche Handlungen in bestimmten Situationen zur Erreichung positiver Ergebnisse am Geeignetesten sind. Zum anderen beschreibt Bandura auch eine

[23] Wichtige Entwicklungen waren unter anderem 1936 das theoretische Modell der Turingmaschine von Alan Turing. Auf dessen Grundlage beschrieb John von Neumann die Fundamente der Computerarchitektur, die auch in heutigen Computern Anwendung finden.

[24] In einer gemeinsamen Arbeit stellten der Neurophysiologe Warren McCulloch und der Logiker Walter Pitts 1943 das Modell eines neuronalen Netzes vor. Sie zeigten, wie das Zusammenspiel von Neuronen in einem Nervensystem mit Elementen der Logik nachgebildet werden kann. Dies inspirierte die wissenschaftliche Gemeinschaft zu der theoretischen Möglichkeit, die Arbeitsweise vom menschlichen Gehirn mit logischen Elementen nachzubauen.

[25] Der Mathematiker und Philosoph Norbert Wiener widmete sich der Natur rückgekoppelter Systeme. 1961 etablierte er den Begriff Kybernetik. Diese neue Wissenschaftsdisziplin widmet sich den Gemeinsamkeiten in der Arbeitsweise von biologischen Nervensystemen, Computern und anderen Maschinen.

[26] Ein wichtiges Element bei der Funktionsweise von Computern war die Art und Weise, wie Informationen theoretisch modelliert, technisch übertragen und gespeichert werden können. Der Elektrotechniker und Mathematiker Claude Shannon zeigte Ende der 1930er Jahre, dass Informationen als logische Zustände (wahr und falsch) in Form von elektromechanischen Relais maschinell umgesetzt werden können. Als elektrisch gesteuerte Schalter schließen oder öffnen diese eine Schaltkreis und ermöglichen damit die Abbildung logischer Funktionen in Form von Schaltkreisen. Er etablierte die Einheit *Bit* als kleinste Basiseinheit einer Information. Diese bezeichnet einen definierten Zustand zwischen zwei Alternativen. Mit der Informationstheorie können Informationen unabhängig von ihrer technischen oder biologischen Basis verarbeitet werden.

[27] Im Rahmen der beiden Weltkriege boten Kriegsopfer mit Verletzungen am Gehirn neue Einsichten in seine Funktionsweise. Der internationale wissenschaftliche Austausch über Pathologien wie Wortfindungsstörungen, Störungen bei der Verarbeitung sensorischer Reize und andere mentale Störungen warfen neue Fragen auf, die auf Grundlage des einfachen Reiz-Reaktions-Lernens nicht zu erklären waren. Dies führte zu einem wachsenden Verständnis über Aufbau und Funktionsweise des menschlichen Gehirns und seiner kognitiven Funktionen.

motivationale Funktion. Menschen haben aufgrund ihres Gedächtnisses antizipatorische Fähigkeiten. Durch vergangene Erfahrungen können Erwartungen an zukünftige Ereignisse schon in der Gegenwart zu zielgerichteten Handlungen führen. Ein:e Autofahrer:in wird nicht jedes Mal den Tank ihres:seines Autos bis auf den letzten Tropfen leerfahren, um dann erst an das Tanken zu denken. Bandura geht davon aus, dass die meisten menschlichen Handlungen auf Grundlage dieser antizipatorischen Kontrolle erfolgen. Weiterhin identifizierte Bandura die Verstärkungsfunktion. Experimentell konnte gezeigt werden, dass Verstärkungen bei Menschen ineffektiv sind, solange sie nicht die Bedingungen verstehen, unter denen eine Belohnung erfolgt. Das Zeigen der erwünschten Handlung steigt dagegen sprunghaft an, sobald die Proband:innen sich der Belohnungsstrategie bewusst waren. Daraus resultiert, dass Lernen durch Einsicht ein wichtiger Moderator für Lernprozesse ist. Bandura resümiert, dass Lernen durch Verstärkung gut dafür geeignet ist, bereits gelerntes Verhalten zu festigen. Allerdings ist es für das Erlernen neuer, komplexer Handlungen ungeeignet (vgl. Bandura 1977, S. 17 - 22).

Auch Jean Piaget hat mit seiner Theorie der geistigen Entwicklung (vgl. Piaget & Inhelder 2000) einen wichtigen Beitrag zur Bereicherung kognitiver Lerntheorien geleistet. Er betont ebenfalls die aktive Beteiligung der Lernenden im Lernprozess. Piaget beschreibt zwei grundlegende Prozesse, mit denen sich das Individuum seiner Umwelt anpassen kann: Assimilation und Akkommodation. Diese werden von Faulstich folgendermaßen zusammengefasst:

> „Assimilation bedeutet die kognitive Integration von Umwelteinflüssen in kognitive Schemata der eigenen, verfügbaren und bevorzugten Art, über diese Dinge zu denken. Beispiel: Wenn für das Kind ein Holzstück zum Schiff wird, so assimiliert es das Holzstück an sein kognitives Schema vom Schiff. Akkommodation meint die Modifikation der Interpretation der Umwelt durch deren Einflüsse. Akkomodation tritt dann auf, wenn es eine Diskrepanz oder Störung der Auseinandersetzung mit der Welt gibt, für die der Organismus noch kein bewährtes Schema besitzt. Dann muss der lernende Mensch gewissermaßen denkend 'im Kopf' umräumen, d.h. seine bisherigen kognitiven Schemata neu ordnen. Beispiel: Ein Säugling hat irgend ein visuelles Element in das invariante Muster einer Rassel assimiliert (Schütteln ergibt Lärm) und ergreift nun gerade ein ähnlich geformtes Holzstück, doch bleibt das auditive Element des Rasselns aus. Nun wird der Säugling seine Aufmerksamkeit auf vorhandene taktile oder visuelle Schemata richten, die eine Unterscheidung zwischen Rassel und Holzstück ermöglichen. Ist diese Unterscheidung getroffen, dann werden die neuen Elemente mit den alten in der Akkommodation zu einem erneuerten Schema verknüpft, das dann seinerseits Ausgangspunkt für weitere Assimilationen zukünftiger Erfahrungen darstellt.“ (Faulstich 2013, S. 44)

Piaget sieht Lernen als einen Prozess, wodurch das Individuum neues Fähigkeiten und

Fertigkeiten durch aktive Auseinandersetzung mit seiner Umwelt aufbaut.

Einfluss anderer Personen im Kognitivismus

Während bei behavioristischen Lerntheorien der Einfluss anderer Personen vom Stellenwert als gleichrangig wie jegliche andere Reize aus der Umwelt eingeschätzt werden, kommt es bei kognitiven Lerntheorien zu einem Umdenken.

Bandura beschreibt in seiner Sozialen Lerntheorie verschiedene Komponenten, wie Lernprozesse durch Einflüsse aus der Umwelt in einer besonderen Art und Weise von anderen Personen kommen. Er verweist auf den Umstand, dass ein Lernen, welches nur auf dem Erleben der Konsequenzen von eigenen Handlungen beruht, mühselig und teilweise sogar gefährlich wäre. Neue Verhaltensweise können auch dadurch erlernt werden, dass andere Personen bei diesen Handlungen beobachtet werden. Diese durch Observation gewonnenen Erkenntnisse dienen später als Grundlage für eigene Handlungen. Der als Modelllernen bezeichnete Prozess bildet die Grundlage für die meisten menschlichen Verhaltensweisen (vgl. Bandura 1977, S. 22).

Andere Personen dienen aber nicht nur als Modelle für Handlungen, sie steuern auch die Auswahl von spezifischen Verhaltensweisen in einer bestimmten Situation. Dies liegt darin begründet, dass es Erwartungen in Bezug auf die Folgen von individuellen Handlungen gibt. Sie wurden in der Vergangenheit gelernt und durch wiederholte Verstärkung gefestigt. Durch reziproke Handlungen von Individuen wird damit ein soziales Milieu geformt, in dem die Beteiligten in einer vorhersehbaren Art und Weise miteinander interagieren (vgl. ebd., S. 198).

3.2.3 Konstruktivismus

Der Konstruktivismus rückt die kognitiven Leistungen von Menschen in den Mittelpunkt. Er verkörpert einen vom behavioristischen und kognitivistischen Ansatz abweichenden, epistemologischen Ansatz. Realität ist nicht mehr eine objektives Phänomen, welches von unseren Sinnesorganen erfasst und deckungsgleich in unserem Geist abgebildet wird. Vielmehr ist Realität eine individuelle Konstruktionsleistung. Diese hängt von unterschiedlichen Einflüssen ab, in denen sich einzelne Denkschulen des Konstruktivismus voneinander unterscheiden. Die Bedeutendsten sollen hier in ihren wichtigsten Aussagen zueinander in Beziehung gesetzt werden, da sie für für diese Arbeit von entscheidender Bedeutung sind.

Radikaler Konstruktivismus

Wegbereiter des Konstruktivismus war der Radikale Konstruktivismus. Dieser vertritt einen fundamental anderen erkenntnistheoretischen Ansatz als die beiden zuvor beschriebenen Lerntheorien. Er lehnt den Determinismus des Behaviorismus und des Kognitivismus ab, der die Umwelt mit all seinen Reizen als objektive Realität definiert. Stattdessen vertritt er die Auffassung, dass Realität individuell konstruiert wird. Jeder Mensch

verfügt über eine eigene Realitätskonstruktion, die individuell verschieden ist. Einer der wichtigsten Vertreter war Ernst von Glasersfeld, der den Begriff des Radikalen Konstruktivismus prägte:

> „Der Radikale Konstruktivismus ist unverhohlen instrumentalistisch. Er ersetzt den Begriff der Wahrheit (im Sinne von der wahren Abbildung einer von uns unabhängigen Realität) durch den Begriff der *Viabilität* innerhalb der Erfahrungswelt der Subjekte. Er verwirft folglich alle metaphysischen Verpflichtungen und beansprucht nicht mehr zu sein, als ein mögliches Denkmodell für die einzige Welt, die wir »erkennen« können, die Welt nämlich, die wir als lebende Individuen konstruieren." (Glasersfeld 1996, S. 55)

Glasersfeld greift auf die genetische Erkenntnistheorie von Piaget zurück. In ihm sah er die ersten Ansätze einer konstruktivistischen Erkenntnistheorie. Er beruft sich auf Piagets Aussagen, dass es kein objektiv richtiges Wissen gibt, sondern nur mehr oder weniger angemessenes Wissen. Daraus folgt, dass Erkenntnis kein Bild der realen Welt ist. Erkenntnis ist ein Prozess der Anpassung (vgl. ebd., S. 42f). Da aber Begriffe wie Anpassung oder Angemessenheit häufig utilitaristisch genutzt werden und so zu Missverständnissen führen, schlägt Glasersfeld stattdessen den Begriff der Viabilität vor:

> „Handlungen, Begriffe und begriffliche Operationen sind dann viabel, wenn sie zu den Zwecken oder Beschreibungen passen, für die wir sie benutzen. Nach konstruktivistischer Denkweise ersetzt der Begriff der Viabilität im *Bereich der Erfahrung* den traditionellen philosophischen Wahrheitsbegriff, der eine »korrekte« *Abbildung der Realität* bestimmt. [...] Der Radikale Konstruktivismus ist also, wie ich schon eingangs sagte, eine besondere Art, Wissen zu begreifen, und zwar Wissen nicht nur als Ergebnis, sondern auch als Tätigkeit." (ebd., S. 43)

Wissen wird dabei nicht passiv aufgenommen, es wird vom Subjekt aktiv aufgebaut. Die menschliche Kognition hat die Funktion, die individuelle Erfahrungswelt zu organisieren, da es keine objektive Realität gibt, die erkannt werden kann (vgl. ebd., S. 48). Beim Wissen handelt es sich um Abstraktionen von Erfahrungen, für die auch in der Zukunft Gültigkeit beansprucht wird. Von unseren Mitmenschen nehmen wir an, dass sie bei ähnlichen Erfahrungen auch ähnliches Wissen aufbauen (vgl. Glasersfeld & Fischer 2013, S. 142).

Beim Wissen macht Glasersfeld eine wichtige Unterscheidung. Wissen kann zum einen durch Training aufgebaut werden, wie zum Beispiel durch Auswendiglernen oder Nachahmung. Zum anderen ist Wissen jedoch auch ein Produkt des Verstehens. Lernen durch Verstehen kann nur über reflexive Abstraktion erfolgen. Dies ist ein Prozess, den jeder Mensch selbst durchlaufen muss. Aus diesem Grund ergibt sich für Lernumgebungen der Umstand, dass Menschen, die denselben Einflüssen ausgesetzt sind, höchst unterschiedliches Wissen daraus ableiten (vgl. ebd., S. 159f). Bei der Konzeption von Lernarrangements sollte entsprechend berücksichtigt werden, dass:

> „[...] eine Tätigkeit, die bei dem einen Schüler reflexive Abstraktion auslösen kann, dies bei einem anderen Schüler nicht bewirken muss. Als Didaktiker sollten wir daher nicht mehr erwarten als die Erarbeitung eines möglichst guten und umfassenden Repertoires methodischer Hilfen, keinesfalls aber ein fixiertes Programm unfehlbarer Übungen oder Verfahren. [...] Lehrer dürfen sich selber niemals als Mechaniker der „Wissensübertragung“ sehen, noch auch als solche betrachtet werden. Sie sollten sich vielmehr als die intuitiven Helfer verstehen und so handeln, dass sie, wie Sokrates sagte, bei der Geburt des Verstehens die Rolle der Hebamme spielen.“ (Glasersfeld & Fischer 2013, S. 160)

Weiterhin differenziert er den Begriff der Akkomodation von Piaget. Er unterscheidet zwischen bewusster und unbewusster Akkomodation. Geht ein neues Handlungs- oder Denkschema mit dem Verständnis einher, warum das neue Schema besser funktioniert als das Alte, so ist dies eine bewusste Akkomodation. Fehlt dieses Verständnis der Zusammenhänge bei dem neuen Schema, handelt es sich um eine unbewusste Akkomodation (vgl. ebd., S. 164 f.). Diese Differenzierung verdeutlicht noch einmal, dass Glasersfeld dem Akt des Verstehens eine besondere Bedeutung zukommen lässt und es mehrere Konstruktionsverfahren für Wissen gibt.

Kritik am Radikalen Konstruktivismus

Der Radikalen Konstruktivismus verfügt nur über eine äußerst rudimentäre Theorie des Lernens. Er liefert keine Aussagen darüber, was wichtige Triebfedern des Lernens sind. So bleibt unklar, unter welchen Voraussetzungen Lernen durch Training in Lernen durch Verständnis umschlägt. Auch die Bedeutung anderer Personen beim Lernen wird nicht thematisiert. Durch den ausschließlichen Fokus auf die individuelle Realitätskonstruktion, verliert die soziale Komponente beim Lernen jegliche Bedeutung. Aus diesem Grund verliert sich der Radikalkonstruktivismus in der Isolation des Individuums. Dies reicht bis zum Vorwurf vom Solipsismus[28] (vgl. Gergen & Wortham 2001, S. 122; Faulstich 2013, S. 51)

> „Die Welt bleibt unerkennbar: Die Wahrheit sei die Erfindung eines Lügners. Das Denkmuster dreht sich: Die Welt (nicht mehr das Ich) wird zur blackbox. Lernen wird nicht von außen angestoßen, sondern von innen.“ (Faulstich 2013, S. 51).

Damit verlieren Begriffe wie Bildung und Mündigkeit ihre Bedeutung, da sich jegliches Wissen am Kriterium der Viabilität auszurichten hat. So fehlt es dem individualistischen Konzept des Radikalen Konstruktivismus an der Analyseebene der Gesellschaft, in dessen Spannungsfeld das Handeln aller Individuen eingebettet ist (vgl. ebd., S. 54 f). Ähnlich

[28] In der Philosophie beschreibt der Solipsismus die Anschauung, dass nur das eigene Ich existiert. Dadurch gibt es keine Gewissheit über die Existenz einer objektiven Realität außerhalb des eigenen Bewusstseins.

sehen es Jochen Gerstenmaier und Heinz Mandl, die dem Konstuktivismus vorwerfen, die Sicht auf die Akteure mit ihren individuellen Bedeutungskonstruktionen beim Wissenserwerb verloren zu haben. In ihren Augen beschreibt der Ansatz eher das Verhältnis, in dem Wissen zur Wirklichkeit steht, statt Antworten auf die Fragen nach der Entstehung des Wissens oder dem Prozess der Wissenskonstruktion und dessen Bedeutung zu liefern (vgl. Gerstenmaier & Mandl 1995, S. 870).

Aufgrund der fundamentalistischen Ansichten des Radikalen Konstruktivismus haben sich viele Ansätze eines moderaten Konstruktivismus entwickelt. Sie beanspruchen nicht eine eigenständige Erkenntnistheorie zu sein, bedienen sich aber einzelner Fragmente des Radikalkonstruktivismus und stellen diesen ins Zentrum ihrer Theorie. Einige dieser Ansätze, die für den Bereich des gemeinschaftlichen Lernens eine besondere Bedeutung haben, werden im Folgenden dargestellt.

Sozialer Konstruktivismus

Eines der Schlüsselwerke des Sozialkonstruktivismus stammt von Peter Berger und Thomas Luckmann mit ihrem Werk *The Social Construction of Reality* aus dem Jahr 1966. Sie vertreten darin die These, dass wir innerhalb der Gesellschaft über einen gemeinsam konstruierten Wissensbestand verfügen. Ein Beispiel für dieses gesellschaftliche Wissen sind die kollektiven Bedeutungszuschreibungen in Bezug auf Sprachbegriffe oder Zeichen, die von einer Generation an die nächste weitergegeben werden. Dies schlägt sich in dem Begriff vom Alltagsverstand beziehungsweise *common sense* nieder. Unser alltägliches Handeln ist geprägt von diesem kollektiven Wissen[29] beziehungsweise diesen intersubjektiv geteilten Bedeutungszuschreibungen (vgl. Berger & Luckmann 1991, S. 55 f.). Im Gegensatz zum Radikalen Konstruktivismus ist die Realität also nicht mehr eine rein individuelle Konstruktionsleistung, sondern wird in einem sozialen Kontext konstruiert.

> „In other words, common-sense 'knowledge' rather than 'ideas' must be the central focus for the sociology of knowledge. It is precisely this 'knowledge' that constitutes the fabric of meanings without which no society could exist. The sociology of knowledge, therefore, must concern itself with the social construction of reality.“ (ebd., S. 27)

In ihrem Buch sehen sich die Autoren in der Tradition der Wissenssoziologie, deren Fundament die These ist, dass Wissen sozial verteilt ist. Die Aufgabe dieser Disziplin der Soziologie ist es, die Mechanismen dieser Verteilung zu analysieren (vgl. ebd., S. 28). Wissen ist demnach die individuell mehr oder weniger stark ausgeprägte Gewissheit, dass bestimmte Phänomene als real erlebt werden, da sie unabhängig von unserem eigenen Willen existieren. Dementsprechend gibt es keine objektive Realität, aber Phänomene,

[29] Dieser Wissensbestand ist allen Angehörigen zueigen, die dieselbe kulturelle Sozialisation durchlaufen haben. Angehörige anderer Kulturkreise haben keinen Zugang zu diesem kollektiven Wissensbestand und können dadurch die Bedeutungen der Sprache und Symbole einer anderen Kultur erst deuten, wenn sie sich die neuen Bedeutungszuweisungen angeeignet haben.

die von den Mitgliedern der Gesellschaft in unterschiedlichen Nuancen ihrer jeweiligen Merkmale als intersubjektive Realität beschrieben werden. Mit anderen Worten: bestimmte Agglomerationen von Wissen und Realität sind eingebettet in einen bestimmten sozialen Kontext. Eine adäquate Analyse dieser Phänomene ist also ohne eine Analyse der sozialen Beziehungen, in denen dieses Wissen geteilt wird, nicht möglich (vgl. ebd., S. 13 - 15).

Wichtige Einflüsse für den Sozialkonstruktivismus kamen aus der kulturhistorischen Schule. Insbesondere Lev Vygotskij erarbeitete Schlüsselkonzepte mit Bezug auf die sozialen Dimension von Lernprozessen. Mit der *Zone der nächsten Entwicklung* beschreibt er die kognitiven Leistungen, die ein Kind nur mit der Unterstützung von anderen Personen bei der Lösung eines bestimmten Problems erreichen kann. Gemeint sind damit die möglichen Fähigkeiten, die als nächstes vom Kind entwickelt werden, aber mit dem aktuellen Entwicklungsstand und ohne Unterstützung anderer noch nicht realisiert werden können. Lernen durch Nachahmung sei aber kein mechanischer Prozess: Nachahmung bei einer Problemlösung ist nur dann erfolgreich, wenn der Prozess bereits in der Zone der nächsten intellektuellen Fähigkeiten angelegt ist (vgl. Vygotskij 2002, S. 326ff). Vygotskijs Forschung wurden von Jerome Bruner aufgegriffen. Er beschrieb zusammen mit Wood und Ross das Konzept des *Scaffoldling,* wie mit Hilfe gezielter Instruktion durch Tutor:innen Kompetenzaufbau in der Zone der nächsten Entwicklung ermöglicht werden kann (vgl. Wood et al. 1976).[30] 1983 veröffentlichte Bruner eine Forschungsarbeit zum Spracherwerb, der die Bedeutung von zwischenmenschlicher Interaktion für das Erlernen der Erstsprache bei Kindern herausstellt (vgl. Bruner 1983, S. 120).

Social Constructivism & Social Constructionism

Kenneth Gergen und Stanton Wortham differenzieren den Sozialkonstruktivismus in zwei Entwicklungspfade. Dies ist zum einen der *social constructivism*, wie er unter anderem von Vygotskij oder Bruner vertreten wurde. Er zeichnet sich durch einen Dualismus von externer und interner Realität, wie er auch bei dem Radikalkonstruktivismus zu finden ist, aus. Dies wirft bei Gergen und Wortham unweigerlich die unbeantwortete Frage auf, wie diese beiden Realitäten miteinander in Verbindung stehen. Weiterhin werfen sie den Vertretern vor, stark dem empirischen Grundsatz der Wertneutralität verhaftet zu sein (vgl. Gergen & Wortham 2001, S. 123 f.).

Der zweite Entwicklungspfad ist der *social constructionism*, in dessen Tradition die beiden Autoren ihren eigenen Ansatz sehen und der auf den Sozialkonstruktivismus von Berger und Luckmann zurückgeht. Mit dem *social constructivism* teilt er die Einschätzung, dass neben den kognitiven Prozessen auch das soziale Milieu mit für den Wissensaufbau ausschlaggebend ist. Allerdings wird den sozialen Einflüssen eine viel stär-

[30] *Scaffolding* wird als Prozess aus sechs verschiedenen Elementen beschrieben: (1) Interesse der Lernenden auf das Problem richten, (2) Anzahl der möglichen Lösungen einschränken, (3) Anregung, die Konzentration auf die Problemlösung zu richten, (4) Zwischenschritte zur Lösung hervorheben, (5) Frustrationskontrolle und (6) Demonstration der Musterlösung (vgl. Wood et al. 1976, S. 98)

kere Gewichtung beigemessen. Auch ist Erkenntnis für Gergen und Wortham immer perspektivisch und nie werturteilsfrei (vgl. ebd., S. 123 - 127). Damit kritisieren die beiden Autoren den Positivismus, der den bereits beschriebenen Lerntheorien anhaftet.

Sie verweisen auf die soziale Konstruktion von Wissen im Rahmen von Aushandlungsprozessen in einer Kulturgemeinschaft. Wahrheit ist nicht das Produkt einzelner Individuen, sondern wird in sozialen Beziehungen immer wieder neu ausgehandelt oder bestätigt. Damit geht einher, dass die Aushandlung von Bedeutungen auch immer unscharf, mehrdeutig und kontextabhängig sind. Die gesellschaftlichen Akteure, die am Bedeutungsdiskurs teilnehmen, machen dies auch vor dem Hintergrund ihrer eigenen Perspektive und vorangegangener Aushandlungsprozesse. Sprache fungiert dabei als konstitutives Element von Beziehungen und ist dadurch der zentrale Faktor für Aushandlungsprozesse (vgl. ebd., S. 119 - 121). Ihre Position schlägt sich auch in ihrer Kritik des Wissenschafts- und Bildungssystems nieder und mündet in folgenden Forderungen (vgl. ebd., S. 125 - 136):

- *Interdisziplinärer Diskurs und Wissenschaftskommunikation fördern:* Die Ausdifferenzierung der einzelnen Wissenschaftsgebiete führt zu einer Verarmung des Diskurses. Fachfremde sind dadurch strukturell ausgeschlossen und auch die Expert:innen haben die Fähigkeit verloren, ihr spezifisches Wissen in den öffentlichen Diskurs einzubringen. Ihre genutzten Begriffe und Bedeutungen sind so eng mit ihrer wissenschaftlichen Domäne verwoben, dass der Diskurs ohne präzise Kenntnisse dieser Sprache nicht anschlussfähig ist. Der Prozess hat seinen Ursprung schon in vorgelagerten Bildungseinrichtungen, die eine Dequalifizierung der Beteiligten fördern. Lehrkräften wird die Rolle zugewiesen, Lehrpläne umzusetzen statt sie zusammen mit den Lernenden in einem gleichberechtigten Prozess auszuhandeln.

- *Fähigkeiten statt Repetition von Lerninhalten:* Die Evaluation der individuellen Lernleistungen im Bildungssystem erfolgt maßgeblich anhand von Wiederholung der im Lehrplan vorgegebenen Inhalte statt auf Erprobung der angeeigneten Fähigkeiten. Auch die Bedeutung von dialogischem Lernen, die Arbeit in Teams sowie die Präsentation der Ergebnisse außerhalb der Bildungseinrichtung wird noch nicht entsprechend gewürdigt.

- *Verknüpfung von Theorie und Praxis:* Statt Vermittlung von Fakten und Theorien sollte das Ziel pädagogischen und andragogischen Handelns in der Herstellung von Kontexten liegen, in denen wissenschaftlicher Diskurs und praktische Anwendung des Wissens miteinander verknüpft sind. Gelernt werden soll anhand realer Problemstellungen, die auch die Sphäre außerhalb der eigenen Wissenschaftsdomäne mit berücksichtigen.

- *Fokus auf gleichberechtigte Beziehung zwischen Lehrenden und Lernenden:* Aus sozialkonstruktivistischer Perspektive sind Individuen nicht Besitzer:innen von Vernunft per se. Vielmehr sind rationale Argumente ein Ausdruck beziehungsbezo-

gener Errungenschaften. Dementsprechend gilt der Beziehungsgestaltung in Lehr-Lern-Situationen ein besonderer Wert. Beziehungen müssen so gestaltet werden, dass sie einen hierarchiearmen, offenen Diskurs fördern. So führt das monologische Lernen dazu, dass besondere Fähigkeiten und Einsichten von Lernenden nicht in den Bildungsprozess eingebracht werden können. Dies ist aber nötig, um kooperatives Lernen zu ermöglichen. Die herausragende Bedeutung von Kollaboration und Kooperation ist für Gergen und Wortham eine der wichtigsten Erkenntnisse sozialkonstruktivistischen Denkens.

Situiertes Lernen und Legitime Periphere Partizipation

Die Theorie des *Situierten Lernens* ist ein „theoretischer Ansatz zur Entwicklung, Implementierung und Evaluation konstruktivistischer Lernumgebungen [...] die die Kontextgebundenheit des Wissenserwerbs in den Mittelpunkt stellt“ (Gerstenmaier & Mandl 2018, S. 225). Die kontextuelle Einbettung von Lernprozessen findet sich schon bei John Dewey. Bereits zu Beginn des 20. Jahrhunderts formulierte er die These, dass Lernen erfahrungsbasiert sei und die Aushandlung von Bedeutungen das Ergebnis sozialer Konstruktionsprozesse darstellt (vgl. Dewey 2001, S, 145, 16). Damit gehört Dewey auch zu den geistigen Wegbereitern des Sozialkonstruktivismus.

Situiertes Lernen ist vor allem eine Theorie über die Wirkungsweise von Lernumgebungen, die aus konstruktivistischer Perspektive konzipiert wurden. Sie forcieren selbstgesteuerte und kooperative Lernformen und bedienen sich dem problemorientierten und fallbasierten Lernen. Folgende Merkmale zeichnen das Situierte Lernen aus (vgl. Gerstenmaier & Mandl 2018, S. 226):

- Lernen ist ähnlich wie beim Kognitivismus ein individueller, aktiv vollzogener, konstruktiver Prozess
- Lernen erfolgt durch Partizipation der lernenden Person an dem in der Lerngruppe distribuierten Wissen
- Lernen wird als Passung an die Restriktionen und Anregungsinhalte des Kontextes aufgefasst, die erst als periphere und bei zunehmender Expertise als zentrale Partizipation charakterisiert sind

Die Bedeutung der Lerngruppe in den Ausführungen von Gerstenmeier und Mandl geht zurück auf die Theorie der *Legitimen Peripheren Partizipation*. Diese Theorie von Jean Lave und Etienne Wenger ist die Ausdifferenzierung des situierten Lernens.

> „By this we mean to draw attention to the point that learners inevitably participate in communities of practitioners and that the mastery of knowledge and skill requires newcomers to move toward full participation in the sociocultural practices of a community. 'Legitimate peripheral participation' provides a way to speak about the relations between newcomers and old-timers, and

about activities, identities, artifacts, and communities of knowledge and practice.“ (Lave & Wenger 1991, S. 28)

Wie schon beim Sozialkonstruktivismus dargestellt, wird der Schwerpunkt auf den Beziehungscharakter von Wissen und Lernen gelegt. Wissen und Lernen sind beide ein Resultat sozialer Aushandlungsprozesse in Gruppen. Mit der Theorie der *Legitimen Peripheren Partizipation* betonen Lave und Wenger die Bedeutung von erfahrungsbasiertem und informellem Lernen und erweitern sie um eine soziale Dimension der Lerngruppe oder allgemein von Praxisgemeinschaften. Durch legitime Teilhabe an der sozialen Praxis der Lerngruppe werden neue Mitglieder integriert. Sie verfolgen den Ansatz, Situiertes Lernen mit Theorien über die Produktion und Reproduktion von sozialer Ordnung zu verbinden. Demzufolge ist Lernen ein Ergebnis von sozialer Praxis (vgl. ebd., S. 33, 47).

3.2.4 Neurobiologische Lerntheorien

Bezogen auf den Entwicklungsstand sozialkonstruktivistischer Lerntheorien stellen neuere Ansätze neurobiologischer Lerntheorien wieder eine Reduktion des komplexen Bedingungsgeflechts dar, in welches das menschlichen Lernen eingebettet ist. Obgleich die Neurowissenschaften erstaunliche Erkenntnisse über Aufbau und Funktionsweise des Gehirns liefern, bleiben viele Fragen über das Lernen unbeantwortet oder bestätigen die vorliegenden Erkenntnisse der Bildungswissenschaft. Mit Hilfe neuer technischer Messverfahren soll das Gehirn beim Denken beobachtet werden. Dies geht mit der Hoffnung einher, geistige Prozesse durch physiologische Veränderungen im Gehirn erklären zu können. Mit diesem Anspruch verbunden ist das Bestreben, Deutungshoheit in der Diskussion über das Lernen zu erringen.

Ein Vertreter neurobiologischer Lerntheorien ist Gerhard Roth. Er vertritt einen biologistischen Lernbegriff, in dem er Lernen als „ [...] eine universell verbreitete Fähigkeit zur mittel- und langfristigen Anpassung eines Organismus an seine Umwelt“ (Roth 2011, Kapitel 4: Arten des Lernens) beschreibt. In einer früheren Publikation bezeichnete Roth sich als Vertreter eines neurobiologischen Konstruktivismus (vgl. Roth 2001, S. 9). Er stellt fest:

„Wahrnehmung beruht also nicht auf einer direkten Abbildung der Welt, einer bloßen Kopie, aber doch auf einer systematischen, wenngleich ausschnitthaften, hervorgehobenen und abgeschwächten Repräsentation der Welt im Gehirn, die mit der Überlebenssituation des Organismus eng zusammenhängt. Wie könnte der Organismus auch überleben, wenn er nicht das *Wesentliche* seiner Umwelt erfasste?“ (Roth 2003, S. 72)

Damit stellt er klar, das wir die Welt nicht klar erkennen können, sondern nur ein Abbild von dem konstruieren können, was unsere Sinne wahrnehmen. Es gibt für ihn einen Ausweg aus dem erkenntnistheorethischen Dilemma, dass wir jegliche Sinneseindrücke immer nur durch den Filter unserer eigenen Sinnesorgane wahrnehmen, selbst wenn es

sich um die wirkliche Realität handelt. Mit Hilfe von sinnesphysiologischen Messmethoden kann von subjektiven Sinneseindrücken abstrahiert werden. Dies ermöglicht einen Austausch über Messwerte. Beispielsweise ist die Farbwahrnehmung individuell unterschiedlich, doch ein Konsens lässt sich über die Messung der Frequenz von Licht herstellen. Dennoch gibt er zu, dass auch diese messbare Wahrnehmungswelt keinen Anspruch auf Objektivität erheben kann. Allerdings ermöglichen naturwissenschaftliche Messmethoden sowie die Hypothesenbildung und Prüfung eine genauere Abbildung der Wirklichkeit, als es uns ohne diese möglich wäre (vgl. ebd., S. 73f).

Bei den für ihn ausschlaggebenden Lernformen bedient er sich bei allen großen lerntheoretischen Strömungen. Als elementare Lerntheorie sieht er den Behaviourismus: klassische, operante und instrumentelle Konditionierung sowie Sensitivierung und Habituation.[31] Beim Kognitivismus bescheinigt er dem Imitationslernen und dem Lernen durch Einsicht eine besondere Bedeutung (vgl. Roth 2011, Kapitel 4: Weitere Lernformen). Wie schon bereits beim Kognitivismus stehen wieder die messbaren kognitiven Leistungen im Mittelpunkt der Hirnforschung. Die Gemeinsamkeiten mit dem Konstruktivismus ergeben sich nur aus der These, dass die Wahrnehmung der Welt eine individuelle Konstruktionleistung darstellt. Hier grenzt sich Roth aber strikt vom Radikalen Konstruktivismus ab. Die Konstruktion von Wirklichkeit ist kein beliebiger Akt, denn:

> „Es ist aber ein großer Irrtum zu glauben, das »denkende Subjekt« konstruiere seine Weltsicht, denn es konstruiert »denkend« nur sehr wenig von dieser Welt. Vielmehr handelt es sich um unbewusste oder vorbewusste Prozesse, deren Produkt und nicht Konstrukteur das denkende Subjekt ist.“ (Roth 2011, Kapitel 10: Wie ist Verstehen zwischen autonomen Systemen möglich?)

Die Konstruktionsleistung verlaufe in einem engen Korridor. Dies stellt sicher, dass wir als Individuen miteinander in Austausch treten können und gemeinsame Verständigung beziehungsweise gegenseitiges Verstehen möglich ist. Dieser Konstruktionsspielraum wird durch folgende Mechanismen begrenzt (vgl. Roth 2011, Kapitel 10: Wie ist Verstehen zwischen autonomen Systemen möglich?):

- die Nähe in der stammesgeschichtlichen Entwicklung: je näher verschiedene Spezies miteinander verwandt sind, desto eher gleichen sich deren Bedürfnisse und kognitive Leistungen
- als *homo sapiens* besitzen wir die gleiche genetische Grundausstattung für Intelligenz, Emotion und Verhalten
- geteilte Kultur und spezifische gesellschaftliche Verhältnisse, die unsere Wahrnehmung steuern

[31] Sensitivierung bedeutet, dass unerwartete Sinneseindrücke (z.B. eine mögliche Gefahr) unsere Aufmerksamkeit auf die Reizquelle lenken. Dadurch werden schnelle Schutzmaßnahmen ermöglicht, falls die Reize auf eine Gefahr hindeuten. Mit Habituation beschreibt er das Phänomen, dass wir uns an einen Reiz gewöhnen, wenn ein unerwarteter Reiz wiederholt keine Bedrohung anzeigt. Die Habituation ist das Gegenstück zur Sensitivierung (vgl. Roth 2011, Kapitel 4: Habituation und Sensitivierung).

- das Erleben von ähnlichen Ereignissen, zum Beispiel gemeinsame Interessen oder Lebenserfahrungen

Die Frage aber, wie sich Lernen in einem sozialen Kontext abspielt, wird nicht beantwortet. Die Antworten, die neurobiologisch orientierte Wissenschaftler:innen in Richtung Erziehungswissenschaft geben, gleichen denen der geisteswissenschaftlich orientierten Pädagogik. Dies bestätigt auch Roth (vgl. ebd., Kapitel 11: Neurodidaktisch-neuropädagogische Konzepte). Dennoch versucht er in seinem Buch *Bildung braucht Persönlichkeit* Ratschläge für eine gute Unterrichtspraxis zu geben, die den Kenntnisstand der Neurobiologie allerdings weit übersteigen. Deswegen muss er unweigerlich auf die Erkenntnisse der erziehungswissenschaftlichen Lehr- und Lernforschung zurückgreifen.

Kritik neurophysiologischer Thesen

Einer der Hauptkritikpunkte neurophysiologischer Thesen ist der biologische Determinismus. Dieser stellt die Handlungs- und Willensfreiheit des Menschen in Frage. Unser Ich und der freie Wille sei, mehr oder weniger, nur eine Illusion.[32] Unser Handeln und Denken sei neurologisch determiniert, das heißt durch die innere Strukturen festgelegt und durch biochemische Prozesse vorgegeben, die ohne jede willkürliche Kontrolle auf Reize reagieren (vgl. Speck 2008, S. 43f). Eine solche Einschätzung würde viele Grundfeste unserer Gesellschaft erschüttern. Konzepte wie Verantwortung, Schuld, Mündigkeit oder Selbstbestimmung würden ihre Bedeutung in Bezug auf individuelles Handeln verlieren, da sich die Handelnden nicht aktiv in einem bewussten Prozess für oder gegen eine Handlung entscheiden können. Weiterhin wird oft das Gehirn als einziger Bezugsrahmen für das Lernen genommen. Es wird ein Bild vom Menschen als Kopfwesen entworfen, während die Bedeutung des Körpers vernachlässigt wird. Seine Funktion wird darauf reduziert, das Gehirn am Leben zu erhalten und es mit Reizen zu versorgen (vgl. ebd., S. 46f). Dies blendet aber die Körpergebundenheit des Lernens aus. Die Gestaltung und Leistungsfähigkeit des eigenen Körpers liefert eine individuelle Perspektive auf die Wahrnehmung der Welt, die sich stark von anderen Menschen unterscheiden kann. Insofern greift die reine Betrachtung von Aktivitätsmustern im Gehirn beziehungsweise auf Änderungen von Übertragungsstärke an Synapsen zu kurz.

Faulstich verweist darauf, dass Lernfähigkeit vor allem ein psychosoziales Phänomen ist und nicht durch die ausschließliche Betrachtung der physikalischen, chemischen und physiologischen Prozesse im Gehirn erklärt werden kann (vgl. Faulstich 2013, S. 60). Er fasst die Kapazität der Hirnforschung wie folgt zusammen:

> „Neurophysiologische Forschungen suchen Antworten auf ganz andere Fragen als sie die Erziehungs- und Bildungswissenschaften beschäftigen – jedenfalls wenn es darum geht, die psychischen Prozesse des Lernens zu be-

[32] Über die Existenz bzw. der Definition des freien Willens gibt es keinen Konsens in der Hirnforschung. Die Diskussion erstreckt sich über die völlige Negation des freien Willens bis hin zu einer Einschränkung des freien Willens als Kontrollinstanz, die jeweils einem, aus dem Gehirn 'hervorsprudelnden' Handlungsinitiativen, den Vorzug gibt oder diese hemmt (vgl. Speck 2008, S. 43ff).

> greifen. Wenn die Hirnforschung bildungswissenschaftliche Antworten gibt, bleiben diese meist im Bereich neuro-terminologisch überformten Alltagswissens. Sie schweben in der Gefahr, hinter den in der Konstruktivismus-Debatte erreichten Stand zurückzufallen und als überholt geglaubte naiv-behavioristische Positionen zu reaktivieren.“ (ebd., S. 61)

Die Einflüsse anderer Personen beim Lernen liegen außerhalb des Fokus neurophysiologischer Lerntheorien. Erkenntnisse über physiologische Vorgänge im Gehirn sind ohne Zweifel ein faszinierendes Forschungsfeld, an die sich viele Hoffnungen aus dem Bereich der Medizin knüpfen. Sie liefern aber keine neuen Impulse zum Verständnis von Lernprozessen in ihrem sozialen Kontext.

3.2.5 Netzwerkorientierte Lerntheorien

Im Folgendem werden zwei weitere Lerntheorien vorgestellt, die Lernen als sozialen Prozess begreifen. Diese verorten sich teilweise im Sozialkonstruktivismus oder grenzen sich von ihm ab. Zentraler Ansatz für diese Arbeit ist die Theorie der *Communities of Practice*, da sie für die Betrachtung von Lernprozessen in sozialen Bewegungen wichtige Analysekategorien liefert. Im Anschluss daran wird der Konnektivismus erläutert, da dieser die Elemente von der Communities of Practice aufgreift und diese auf den Anwendungsfall von internetbasierten Netzwerken überträgt.

Communities of Practice

Das Konzept geht zurück auf Étienne Wenger und ist eine Weiterentwicklung vom Ansatz der *Legitimate Peripheral Participation* (vgl. Lave & Wenger 1991). Lernen beschreiben sie als Aufbau von Kompetenzen in all jenen Kompetenzbereichen, welche sich auf das Reflexionsvermögen auswirken. Dies steht im Gegensatz zum reinen Erwerb von Wissen und Fertigkeiten als isolierte Handlungsweisen. Nach Wenger bedeutet Lernen:

> „It is what changes our ability to engage in practice, the unterstanding of why we engage in it and the resources we have at our disposal to do so. [...] Such learning has to do with the development of our practices and our ability to negotiate meaning. It is not just the acquisition of memories, habits, and skills, but the formation of an identity. Our experience and our membership inform each other, pull each other, transform each other. “ (Wenger 1998, S. 95f)

Lernen bedeutet also Aufbau von Kompetenzen, um in irgendeiner Art und Weise mit anderen Individuen soziale Beziehungen eingehen zu können und eine Einsicht darin, warum dies wichtig ist. Die Austauschprozesse wirken sich auf die Identitätsentwicklung aller Beteiligten aus. Kernthesen vom Ansatz der *Communities of Practice* ist die sozial verteilte Aushandlung von Bedeutung und Sinn sowie die soziale Praxis in einer Gemeinschaft.

Sinn und Bedeutungsaushandlung

Jede Erfahrung, die wir machen, ordnen wir in einen Sinnzusammenhang ein. Ob etwas sinnvoll ist oder nicht, wird immer wieder neu bestimmt. Dies ist eine wichtige Einflussgröße, die unser Handeln bestimmt. Eine Praxis nach Wenger ist ein Prozess, bei dem wir die Welt und unser Handeln in ihr als bedeutsam für uns erleben. Jedes von Menschen geschaffene Artefakt ist oder war Teil einer solchen sozialen Praxis. Dies gilt ebenso für die notwendigen Werkzeuge und das Wissen, um dieses Artefakt zu produzieren. Dies impliziert auch immer einen historischen Hintergrund, vor dem Sinnhaftigkeit erfahren wird (vgl. Wenger 1998, S. 51f).

Die Aushandlung von Bedeutung beziehungsweise die Bestimmung von Sinn unterliege immer zwei Prozessen: Partizipation und Verdinglichung. Partizipation ist die Teilnahme an sozialen Aktivitäten. Die Teilnehmer:innen formen durch Handlungen gegenseitig ihre Bedeutung, welche sie diesem Austauschprozess beimessen. Dies muss nicht nur positiv für die Beteiligten sein. Auch die Erfahrung von Ungleichheit in sozialen Beziehungen ist ein prägender Prozess für die eigenen Sinnstrukturen. Verdinglichung meint, dass sich soziale Praxen mit ihren Sinnstrukturen zu kulturellen Objekten manifestieren. Dies können physische Gegenstände wie auch Prozeduren oder Wissen sein. In ihnen ist eine Vielzahl von sozialen Austauschbeziehungen der Vergangenheit geronnen und sie beeinflussen wiederum soziale Aktivitäten in der Gegenwart und Zukunft (vgl. ebd., S. 58 - 61).

Praxisgemeinschaften

Wie der Name suggeriert, wird eine Gemeinschaft in diesem Konzept durch eine gemeinsame Praxis definiert. Beispielsweise ist eine nachbarschaftliche Gemeinschaft nicht unbedingt durch eine gemeinsame Praxis gekennzeichnet, obwohl alle nah beieinander wohnen und sich viele Erfahrungen ähneln. Wenger geht es darum, die Praxisgemeinschaft als besondere Gemeinschaft von gemeinschaftlichen Aktivitäten, Traditionen oder Organisationen abzugrenzen (vgl. Wenger 1998, S. 72f).

In diesem Sinne definiert das aufeinander bezogene, alltägliche Handeln einer Gruppe von Leuten, ob es sich um eine Praxisgemeinschaft handelt oder nicht. Eine Organisation oder ein Unternehmen kann eine Praxisgemeinschaft darstellen oder aus vielen Praxisgemeinschaften bestehen. Ebenso ein Zusammenschluss von Leuten, die sich nicht unter dem formalen Rahmen eines Vereins, einer Firma oder einer Interessenvertretung versammeln, können unbewusst eine Praxisgemeinschaft sein. Für Wenger sind für eine Praxisgemeinschaft folgende Merkmale ausschlaggebend: wechselseitiges Engagement, gemeinschaftliche Unternehmung und das gemeinsames Repertoire.

Wechselseitiges Engagement

Essentiell für eine Praxisgemeinschaft nach Wenger sind alle Dinge, die eine auf Gegenseitigkeit beruhende Praxis ermöglichen. Ziel ist der Austausch untereinander, egal in welcher Form dieser geschieht und wodurch dieser entfacht und lebendig gehalten wird.

Wesentlich ist, dass die Beteiligten an wichtigen Prozessen der Gemeinschaft einbezogen sind und sie durch ihr Engagement ihre Zugehörigkeit ausdrücken können (vgl. Wenger 1998, S. 74).

> „Mutual engagement involves not only our competence, but also the competence of others. It draws on what we do and what we know, as well as on our ability to connect meaningfully to what we don't do and what we don't know - that is, to the contributions and knowledge of others." (ebd., S. 76)

Die gegenseitige Unterstützung in einer Praxisgemeinschaft kann sich in zwei Arten ausdrücken: entweder durch komplementäre Handlungen oder durch sich überlappende Tätigkeiten. Beide Formen können auch zusammen auftreten (vgl. ebd., S. 76).

In der kollegialen Fallberatung helfen sich beispielsweise Sozialarbeiter:innen bei der Betreuung ihrer Klient:innen gegenseitig. Die Tätigkeitsbereiche ähneln sich und die Praxisgemeinschaft profitiert von den unterschiedlichen Berufserfahrungen und dem Austausch darüber unter den Kolleg:innen. Ein Operationsteam im Krankenhaus basiert auf komplementären Handlungen. Jede Person ist Spezialist:in auf einem Gebiet und nur durch aufeinander abgestimmte Handlungen wird die Operation effizient durchgeführt. Niemand im Team macht das Gleiche, alle ergänzen sich gegenseitig. Die Praxisgemeinschaft funktioniert durch die Kombination verschiedener Handlungen, die auf unterschiedlichen Wissensbeständen beruhen.

Gemeinschaftliche Unternehmung

Eine gemeinschaftliche Unternehmung heißt nicht unbedingt, dass alle mit allem einverstanden sind oder dieselben Ansichten vertreten. Bei einigen Praxisgemeinschaften wird der Dissens als treibende Kraft angesehen. Wichtig ist allerdings, dass dies das Ergebnis eines gemeinschaftlichen Aushandlungsprozesses ist. Dabei bringt sich jede Person vor dem Hintergrund ihrer eigenen Lebenssituation ein. Die Gemeinschaft muss täglich neu einen Umgang mit Unterschieden und abweichenden Bestrebungen innerhalb ihrer Mitglieder finden. Die Gemeinschaft ist bei diesem Prozess stets in ein größeres Spannungsfeld eingebunden, das aus institutionellen sowie kulturellen beziehungsweise gesellschaftlichen Anforderungen besteht, die sich historisch entwickelt haben (vgl. Wenger 1998, S. 78f).

Das Geflecht aus Rechenschaftspflichten materialisiert sich in Form von Regeln, Richtlinien, Standards und Zielen. Sie geben vor, in welchem Rahmen die Gemeinschaft und ihre Individuen handeln und Ereignisse interpretieren. Einige von ihnen sind explizit ausformuliert, andere sind nur als ungeschriebene Gesetze implizit präsent. Diese sind nicht starr, sondern unterliegen immer einer gewissen Interpretation derer, die täglich ihr Handeln danach ausrichten. Um in einer Praxisgemeinschaft ein erfolgreiches Mitglied zu werden, bedarf es der Fähigkeit, zwischen verdinglichten Handlungsstandards und flexiblen Engagement in der Praxis unterscheiden zu können (vgl. ebd., S. 81f).

Geteiltes Repertoire

Weiterhin wichtig für den Zusammenhalt der Gemeinschaft ist das gemeinsame Repertoire. Es besteht aus materiellen wie auch aus immateriellen Dingen. Dies sind beispielsweise Routinen, Begriffe, Werkzeuge, Gesten, Symbole, Konzepte, Geschichten oder Denk- und Handlungsweisen. Diese sind mit der geschichtlichen Entwicklung der Praxisgemeinschaft verknüpft und prägen wiederum die zukünftigen Prozesse. Das Repertoire verkörpert verdinglichte und partizipative Aspekte. Damit ist es das Resultat vergangener Bedeutungsaushandlungen der Mitglieder und spiegelt damit auch deren Identität und Ausdrucksweisen wieder. Da die Elemente des Repertoires im Kontext ihrer geschichtlichen Entwicklung immer wieder verschieden interpretiert werden, wohnt ihnen eine Mehrdeutigkeit inne. Wenger spricht von der Ambiguität des Repertoires. Dies kann zu Problemen führen, andererseits aber auch eine Quelle für neue Sinn- und Bedeutungszuschreibungen sein. Durch diese Ambiguität unterliegt das geteilte Repertoire einem ständigem Wandel. Die Probleme der interpretativen Unschärfe führen jedoch immer wieder zu Missverständnissen bei der Koordination und Kommunikation. Diese müssen in einem stetigen Prozess immer wieder korrigiert und angepasst werden, um das gemeinschaftliche Handeln der Praxisgemeinschaft am Laufen zu halten. Die Notwendigkeit einer fortwährenden Kommunikation führt auch zur Schaffung neuer Bedeutungen innerhalb des Repertoires der Praxisgemeinschaft. Denn für Wenger bietet Ambiguität folgende Chance:

> „It is useless to try to excise all ambiguity; it is more productive to look for social arrangements that put history and ambiguity to work. The real problem of communication and design then is to situate ambiguity in the context of a history of mutual engagement that is rich enough to yield an opportunity for negotiation.“ (Wenger 1998, S. 84)

Kommunikationsprobleme müssen nur dann angesprochen und gelöst werden, wenn sie das gegenseitige Engagement behindern. Ansonsten sind sie eher als Chance zu begreifen, um neue Bedeutungsaushandlungen zu initiieren (vgl. ebd., S. 83f).

Damit arbeitet Wenger die herausragende Stellung der Koordination und Kommunikation für Praxisgemeinschaften heraus, die für das langfristige Bestehen relevant sind. Aus diesem Grund muss bei der Analyse einer FLOSS-Entwicklungsgemeinschaft der Projektkommunikation zwischen den Mitgliedern ein besonderer Stellenwert beigemessen werden. Dementsprechend wird die Kommunikation in einer Entwicklungsgemeinschaft im Rahmen des empirischen Teils dieser Arbeit besonders berücksichtigt.

Lernen als Praxis

Es ist davon auszugehen, dass die meisten Praxisgemeinschaften Lernen nicht als explizites Ziel ihrer kollektiven Tätigkeit definieren. Dennoch können Praxisgemeinschaften als geteilte Lerngeschichten ihrer Mitglieder betrachtet werden. Lernen ist nicht Selbstzweck, sondern was gelernt wird, ist das kompetente Engagement in der gemeinsamen

Praxis (vgl. Wenger 1998, S. 95). Durch die schon beschriebenen Prozesse der Verdinglichung und Partizipation entwickelt sich die Praxisgemeinschaft weiter (vgl. ebd., S. 86).

Wenger unterscheidet dabei drei Dimensionen des Lernens. Es komme zur Weiterentwicklung in den Bereichen des gegenseitigen Engagements, der gemeinsamen Unternehmung und des geteilten Repertoires (vgl. ebd., S. 95):

1. Weiterentwicklung des gegenseitigen Engagements:

   Lernen bedeutet herauszufinden wie kompetent gehandelt werden kann. Dies geht einher mit dem Aufbau von Beziehungen zu anderen Mitgliedern der Praxisgemeinschaft. Diese Austauschprozesse führen dazu, dass Identitäten definiert werden müssen. Das Engagement in der Praxisgemeinschaft erfordert eine Kenntnis darüber, welche Personen darin interagieren, wer was weiß und welche Rollen die Personen innehaben. Dabei zeigt sich auch, mit welchen Personen das gegenseitige Engagement besser funktioniert.

2. Verständnis und Abstimmung über die gemeinsame Unternehmung:

   Dies erfordert, das eigene Engagement auf die gemeinsame Unternehmung auszurichten und ist mit der Übernahme von Verantwortung verbunden. Dementsprechend bedeutet dies aber auch, andere Personen zur Verantwortung zu ziehen. Dadurch wird Deutungsmacht über die gemeinsame Unternehmung aufgebaut, was auch mit Abstimmungsprozesse über widersprüchliche Interpretationen bezüglich der Unternehmung einher geht.

3. Entwicklung des geteilten Repertoires, Stile und Diskurse:

   Die dritte Dimension des Lernens beinhaltet die Neuaushandlung der Bedeutung und die Neuschaffung verschiedener Elemente. Dies betrifft beispielsweise die Umdeutung oder Aufgabe alter Begriffe und die Entwicklung neuer Begriffe. Ebenso werden neue Routinen aufgebaut beziehungsweise alte Routinen angepasst und neue Werkzeuge, Artefakte und Repräsentationen werden sich angeeignet. Das fortgesetzte Engagement in der Praxisgemeinschaft schafft neue Ereignisse, die in Form von Geschichten erzählt und nacherzählt werden.

Diese Dimensionen des Lernens sind alle miteinander verwoben. Ein Ereignis, wie beispielsweise die Integration einer neuen Person in die Praxisgemeinschaft, kann in allen drei Dimensionen zu Veränderungen führen. Für die erweiterte Gruppe können sich neue Möglichkeiten für gegenseitiges Engagement ergeben und neue Beziehungen führen wiederum zu neuen Interessen, die zur Neuverhandlung über die gemeinsame Unternehmung führen können. Ebenso kann dieser Prozess eine neue Generation von Elementen des gemeinsamen Repertoires produzieren. Dies zeigt, dass Lernen kein isolierter Prozess eines Individuums ist und die Quelle für eine soziale Struktur darstellen kann (vgl. ebd., S. 96f).

Das Lernen, das erforderlich ist, um in eine Praxisgemeinschaft hineinzuwachsen, erfolgt nicht entlang eins vorgegebenen Curriculums. Vielmehr ist es eine Modifikation der

Beteiligungmöglichkeiten, die die gemeinsame Praxis für Nichtmitglieder öffnet. Neue Mitglieder nähern sich aus der Peripherie langsam an die volle Partizipation an. Sie ist wie ein Schutzraum, der mehr Fehlertoleranz und Begleitung durch andere Mitglieder bietet, als die volle Intensität der Praxis im Zentrum der Gemeinschaft. Gleichzeitig müssen Neuankömmlinge von den übrigen Mitgliedern genug Legitimität erhalten, um als potentielle Mitglieder behandelt zu werden. Nur so können sie am Lernen in der Praxisgemeinschaft teilhaben (vgl. ebd., S. 100f).

Theorie des Konnektivismus

Viele Aspekte, die bei der Theorie der Communities of Practice betrachtet werden, werden auch vom Konnektivismus abgedeckt. Der Begriff wurde von George Siemens eingeführt und beschreibt Ansätze einer Lerntheorie, die den wachsenden Möglichkeiten der Informations- und Kommunikationstechnik Rechnung tragen soll. Grundlage sei die diagnostizierte Unfähigkeit der drei großen traditionellen Lerntheorien, organisationales Lernen zu erklären. Diese Lerntheorien seien blind für Lernprozesse, die außerhalb von Personen stattfinden (vgl. Siemens 2004, S. 3). Siemens sieht Lernen als Prozess an, der zum Aufbau von handlungorientiertem Wissen führt. Es wird durch die Verbindung von spezialisierten Datensätzen erworben. Dieses Wissen kann auch außerhalb von Personen existieren, wie zum Beispiel in Form von Organisationen oder Datenbanken (vgl. ebd., S. 5):

> „Learning [...] is focused on connecting specialized infomation sets, and the connections that enable us to learn more are more important than our current state of knowing. Connectivism is driven by the understanding that decisions are based on rapidly altering foundations. New information is continually being acquired. The ability to draw distinctions between important and unimportant information is vital.“ (ebd., S. 5)

Lernen ist seinem Verständnis nach die Verknüpfung von persönlichem Wissen, dem Wissen anderer Personen und ausgelagerten Datensätzen, welches in einem spezifischen Kontext Handlungsmöglichkeiten eröffnet. Für ein kontinuierliches Lernen ist die Pflege und der Aufbau dieser Verknüpfungen unumgänglich. Wissen speist sich aus einem sozialen Netzwerk, welches das persönliche Wissen erweitert. Siemens gebraucht die Metapher, dass die Rohrleitung wichtiger ist als der Inhalt des Rohres. Dies referenziert auf die Vergänglichkeit von Wissen. In einer Welt, wo die Halbwertzeit von Wissen schneller vergeht als sie sich ein Individuum in Gänze aneignen könnte, sei es essentiell zu wissen, wo die fehlenden Informationen gefunden werden können (vgl. ebd., S. 5 - 7).

An Siemens Ansatz kritisch anzumerken ist, dass Lernen zwar in nicht-menschlichen Strukturen vorkommt, beispielsweise in Form des maschinellen Lernens, jedoch Organisationen an sich keine lernfähigen Strukturen darstellen. Lernen tun die Personen die in einer Organisation miteinander interagieren. Sie passen ihre Praktiken an und erschaffen im

gemeinschaftlichen Austausch und vor dem Hintergrund kulturell geprägter Handlungsweisen ein gemeinsames Repertoire, wie es Wenger in einer Communities of Practice beschreibt. Die von ihm genutzte Rohrleitungsmetapher ist insofern relevant, als dass Wissen sozial verteilt ist und ständigen Aushandlungen unterworfen ist. Dementsprechend ist der Zugang zu Informationsquellen essentiell für das Lernen. Eine ausschließliche Betrachtung der Rohrleitungen negiert jedoch die Bedeutung der Akteur:innen und kulturhistorischen Entwicklungen und kommt einer inhaltsleeren Außenbetrachtungen von Lernprozessen gleich.

Lernnetzwerke

Stephen Downes greift den Konnektivismus auf und wendet ihn in Bezug auf das Distanzlernen mit dem Schwerpunkt E-Learning an. Bei seinem Lernbegriff fällt die Nähe zur Theorie der Communities of Practice auf:

> „Learning [...] occurs in communities, where the practice of learning is the participation in the community. A learning activity is, in essence, a *conversation* undertaken between the learner and other members of the community. This conversation, in the web 2.0 era, consists not only of words but of images, video, multimedia and more.“ (Downes 2010, S. 19f)

Wissen sei kein objektiver Bestand an Informationen, der via Personen oder Lehrmaterialien transferiert wird, wie es seiner Meinung nach in kognitiven Lerntheorien der Fall sei. Lernen ist ein begleitender Prozess bei der Produktion von Inhalten durch die Interaktion in einem Netzwerk. Downes nutzt dafür den Begriff *Connected Knowledge*. Es liegt dezentral im Netzwerk durch die Verbindung der einzelnen Entitäten vor. Deswegen bezeichnet er den Konnektivismus als Emergenztheorie, da Wissen erst durch die Interaktion im Netzwerk entsteht. Den Kognitivismus beschreibt Downes als Kausaltheorie. Bei kognitiven Lerntheorien wird Wissen von Entität A auf Entität B übertragen. Lernen wird dabei als ein klar nachvollziehbarer Ursache- und Wirkungszusammenhang betrachtet, wie es sich auch in vielen Testverfahren über Lernergebnisse niederschlägt (vgl. ebd., S. 3, 21). Downes berechtigte Kritik an der gegenwärtigen Praxis der Lernevaluation überträgt er auf das gesamte Gebiet kognitiver Lerntheorien. Dabei übersieht er, dass sich gerade der Kognitivismus von einfachen Kausalbeziehungen löst und sich für die kognitiven Prozesse der Lernenden öffnet.

Der für diese Arbeit relevante Aspekt von Downes Ausführungen liegt in der kommunikativen Produktion von Wissen. Dabei stellt sich die Frage, wie es unter der Bedingung von sich stetig ändernden Diskursen und Teilnehmenden, zur Produktion von verlässlichem Wissen kommen kann? Dies kann Downes zufolge nur dann gewährleistet werden, wenn der Diskurs folgende Freiheiten gewährleistet (vgl. ebd., S. 18):

- Diversität an Meinungen und unterschiedlichen Blickwinkeln

- Autonomie: Einzelne Entitäten haben die Möglichkeit, sich anhand eigener Entscheidung, Kenntnissen und Werten an der Interaktion zu beteiligen
- Interaktivität: Entitäten müssen sich in ihren Diskursen aufeinander beziehen
- Offenheit: Jede Perspektive muss kommuniziert werden können und sie darf nicht durch andere Entitäten ignoriert werden

Er sieht Gemeinschaften als Netzwerke, die aus vernetzten Entitäten bestehen, welche untereinander Signale senden und empfangen. Die Bedeutung eines Signals liegt aber nicht im Inhalt selbst begründet, sondern in der Interpretation durch den:die jeweilige:n Empfänger:in und ist abhängig vom Kontext der Nachricht, der Relevanz für die Person, der Emergenz im Sinne von Resonanzprozessen eines Signals im Netzwerk und der individuellen Geschichte (vgl. ebd., S. 7f).

Lernnetzwerke im Internet

E-Learning, welches nach den Grundsätzen des Konnektivismus ausgerichtet ist, bezeichnet er in Anlehnung an das Web 2.0 als *E-Learning 2.0*. Dies entspringt seiner Kritik an traditionellen Konzepten von E-Learning, welche um Kursinhalte, Zeitpläne oder Wissenstests herum konzipiert wurden. Statt auf die Bedürfnisse der Lernenden sei das E-Learning-Angebot entlang den Erfordernissen der jeweiligen Bildungsinstitution ausgerichtet. Die Weiterentwicklung hin zur Version 2.0 entspricht dem Schritt weg von der Übertragung von Lerninhalten hin zur gemeinschaftlichen Produktion von Wissen durch die Lernenden selbst (vgl. Downes 2010, S. 12). Eine Lernumgebung, die diesen Anforderungen entspricht, sollte folgende Rahmenbedingungen beachten (vgl. ebd., S. 15f):

- Lernnetzwerke sind dezentral: Die Informationsflüsse sind über die verschiedenen Lernenden verteilt. Es gibt kein Zentrum, welches die Kommunikation in bestimmte Richtungen lenkt.
- Das Wissen liegt verteilt vor: Es ist an unterschiedlichen Orten verteilt oder wird durch die Lernenden selbst eingebracht.
- Der Austausch unter den Lernenden ist nicht moderiert: Alle haben direkten Zugang zu Informationsquellen und zu anderen Lernenden.
- Die Inhalte sind nicht aggregiert: Das Wissen liegt nicht in Form von vorgefertigten Modulen oder Kurseinheiten vor. Dies soll die Integration neuer Informationen oder Handlungsroutinen erleichtern.
- Offenheit muss gewährleistet sein. Alle müssen Zugang zu den Informationen haben beziehungsweise die Möglichkeit haben, sich an Handlungsabläufen beteiligen zu können.

- Beteiligung: Alle Teilnehmenden können autonom agieren und haben die gleichen Freiheiten.
- Dynamische Struktur: Lernnetzwerke müssen dynamisch sein, um sich an variierende Umstände wie neue Mitglieder und neues Wissen anpassen zu können.
- Lernnetzwerke sind nicht segregiert: Das bedeutet, sie sind nicht ausschließlich für das Lernen da. Das Lernen ist kein Selbstzweck, es ist eingebunden in die Lebenswelt der Lernenden, in alltägliche Aktivitäten. In ihnen wird gelernt, gespielt, sich themenfremd unterhalten oder es dient als Infrastruktur, um Probleme zu lösen die nicht Teil der primären Funktion des Netzwerkes sind.

Die Lernumgebung soll nicht um einen Lerninhalt herum konzipiert sein, sondern sich an einem konkreten Problem oder Aufgaben in der Realität orientieren. Daraus folgt, dass diese interdisziplinär ausgerichtet sein muss. Das Lernen erfolgt in hohem Maße problemorientiert, informell und ist in einem anwendungsbezogenen Kontext eingebettet (vgl. ebd., S. 20f). E-Learning 2.0 basiert auf Interaktion, Partizipation, Teilen und der gemeinschaftlichen Erschaffung neuen Wissens. Es ist kein geplantes Lernen anhand eines Curriculums, sondern ist eingebettet in als sinnvoll erachtete Tätigkeiten. Als konkrete Ausformung einer solchen konnektivistisch ausgerichteten Lernumgebung schlägt er lose verbundene, komplementäre Werkzeuge wie Wikis oder Blogs vor. Diese lassen sich unter der Kategorie *Social Software* zusammenfassen. Dies ist Software, die dazu dient, ad-hoc Lerngemeinschaften aufzubauen und zu pflegen (vgl. ebd., S. 1, 12).

3.2.6 Zwischenfazit: Gemeinschaftliches Lernen im Internet

Die Betrachtung dieser Lerntheorien zeigt, dass der soziale Kontext von Lernprozessen sich äußerst unterschiedlich in den jeweiligen Theorien niederschlägt. Die größte Sensibilität für derartige Einflussfaktoren ist beim Sozialkonstruktivismus zu finden. Durch die These, dass Wissen durch kollektive Aushandlungsprozesse sozial konstruiert ist, ist die soziale Dimension untrennbar mit dem Lernen verknüpft. Aus diesem Grund kommt eine Analyse des Lernens in internetgestützten, informellen Lernnetzwerken nicht ohne eine Betrachtung der Interaktionen zwischen den einzelnen Mitgliedern solcher Netzwerke aus.

Beide in diesem Abschnitt aufgeführten netzwerkorientierten Lerntheorien berufen sich in ihrem Kern auf den Sozialkonstruktivismus und lassen sich auf FLOSS-Gemeinschaften übertragen. Die Gemeinschaft von Entwickler:innen von FLOSS-Projekten agieren als eine Praxisgemeinschaft nach der Theorie der Communities of Practice von Lave und Wenger. Um diese These zu belegen, werden im empirischen Teil dieser Arbeit die Kriterien einer Praxisgemeinschaft anhand der untersuchten FLOSS-Gemeinschaften geprüft. Dies ist das wechselseitiges Engagement, die gemeinschaftliche Unternehmung und das gemeinsame Repertoire. Das Netzwerklernen nach Downes eignet sich hervorragend für

die Beschreibung der Prozesse innerhalb einer solchen Entwicklungsgemeinschaft. Downes hat Rahmenbedingungen formuliert, die für eine solche Lernumgebung notwendig sind. Damit greift er den Ansatz der Praxisgemeinschaften auf und entwickelt ihn mit Schwerpunkt auf Lernnetzwerke weiter. Auch dieser Punkt wird im empirischen Teil aufgegriffen und seine Eignung für die Betrachtung von FLOSS-Projekten untersucht.

3.3 Internetgestütztes Kooperatives Lernen

Zur begrifflichen Einordnung des gemeinschaftlichen Lernens im Kontext von Entwicklungsgemeinschaften der FLOSS-Bewegung werde ich anhand verschiedener Typologien des E-Learnings eine Einteilung vornehmen. Dabei wird der Fokus auf das Lernen durch Kooperation und Kollaboration gelegt und weiter auf die Form von computerunterstütztem kooperativem und kollaborativem Lernen eingegrenzt. Zudem soll der Begriff der Entwicklungemeinschaft durch den Vergleich mit virtuellen Lerngruppen theoretisch untermauert werden.

3.3.1 Eingrenzung von E-Learning

E-Learning ist kein fest definierter Begriff, vielmehr dient es als Sammelbegriff für die unterschiedlichsten Formen von Lernprozessen unter Verwendung von digitalen Medien. Dies kann isoliert als individuelles Lernen mit lokal installierter Software oder vernetzt mit anderen Informationsquellen über das Internet erfolgen (vgl. Reinmann 2007, S. 182).

Gabi Reinmann-Rothmeier (vgl. 2003, S. 31 - 34) unterscheidet drei Formen. E-Learning erfolgt durch:

1. Zugang zu Informationen in Datenbanken oder zu digitalisierten Medien.

2. Interaktion: Lernende interagieren mit didaktisch aufbereiteten Informationen aus einem E-Learning-System. Sie erhalten Rückmeldungen und können sich damit schrittweise Lerninhalte erschließen.

3. Kollaboration: Lernende vernetzen sich untereinander, kommunizieren, bringen ihre individuellen Kompetenzen ein und arbeiten zusammen an der Problemlösung.

Eine nicht ganz unähnliche Typologie haben Egon Bloh und Burkhard Lehmann erstellt. Sie unterscheiden zwischen Telelernen, computergestützten Lehr-Lern-systemen und Online-Lernnetzwerken. Telelernen meint die Übertragung von Präsenzsituationen, zum Beispiel durch eine gefilmte Lehrveranstaltung. Unter Lehr-Lern-systemen verstehen die Autoren das Web-Based Training.[33] Der letzte Typus der Lernnetzwerke bezieht sich auf Umgebungen, wo durch Kommunikation, Kollaboration und Kooperationen lernen ermöglicht wird (vgl. Bloh & Lehmann 2002, S. 19).

[33] Das Web-Based-Training ist eine Form des Computer Based Training bei dem Lerninhalte online über eine Netzwerk abgerufen werden.

Beide Typologien verfolgen eine ähnliche Differenzierung nach der Art der transportierten Inhalte beziehungsweise den Kommunikationsverläufen und der Aktivität der Lernenden in der jeweiligen Lernsituation. Im Folgendem wird sich mit E-Learning durch Kooperation und Kollaboration befasst werden, da dieser Typus für die Analyse der Strukturen und Prozesse in Entwicklungsgemeinschaften der FLOSS-Bewegung maßgebend ist.

3.3.2 Lernen in Gemeinschaft

Wie auch schon die Typologien zeigen, wird in der Forschungsliteratur teilweise von Kooperation und teilweise von Kollaboration geredet. Oft wird der Begriff nur unscharf definiert oder es wird bewusst oder unbewusst nur einer von beiden Begriffen genutzt. Teilweise werden die Begriffe auch synonym genutzt. Dies gilt für die englischsprachige Literatur (vgl. Dillenbourg 1999, S. 8) ebenso wie für die deutschsprachige Verwendung (vgl. Dewe & Weber 2007, S. 76 oder auch Haake et al. 2012, S. 1). Um die beiden Begriffe für diese Arbeit sauber voneinander abzugrenzen, bedarf es zunächst einmal der Präzisierung.

Kooperation versus Kollaboration

Autor:innen, die bewusst Kollaboration und Kooperation unterscheiden, weisen oft auf die unterschiedliche Art der Aufgabenteilung hin. Bei einer Kooperation erfolgt die Aufgabenbearbeitung von Teilproblemen getrennt. Erfolgt die Bearbeitung von Gesamt- oder Teilproblemen dagegen gemeinsam, ist dies eine Kollaboration (vgl. Konrad 2014, S. 80; Hinze 2008, S. 242). Exemplarisch hierfür die Begriffsdefinition von Dillenbourg:

> „In cooperation, partners split the work, solve sub-tasks individually and then assemble the partial results into the final output. In collaboration, partners do the work 'together'." (Dillenbourg 1999, S. 8)

Dillenbourg differenziert die beiden Prozesse also nach der Art der Aufgabenteilung. Er merkt allerdings an, dass es auch bei enger Kollaboration zu arbeitsteiligem Vorgehen in unterschiedlichen Bereichen kommen kann. Aus diesem Grund unterscheidet er zwischen horizontaler und vertikaler Aufgabenteilung. Vertikale Aufgabenteilung liegt vor, wenn Aufgaben in unabhängige Unteraufgaben geteilt werden. Diese können getrennt voneinander bearbeitet werden. Diese Art der Vorgehensweise ist symptomatisch für eine Kooperation. Bei der horizontalen Aufgabenteilung gibt es eine Trennung der Aufgaben zum Beispiel auf operativer Ebene und auf der übergeordneten Metaebene. Bei einer Kollaboration sind die Aufgaben der unterschiedlichen Ebenen eng miteinander verwoben und es gibt keine klare beziehungsweise eine ständig wechselnde Zuordnung zwischen Personen und Aufgaben über die unterschiedlichen Ebenen hinweg (vgl. ebd., S. 8).

Eine andere Art der Unterscheidung sieht Dillenbourg in spezifischen Kriterien, die für eine Kollaboration typisch sind. Dies sind Interaktivität, Synchronität und Verhandelbar-

keit. Fehlen diese Merkmale, wird eher von einer Kooperation ausgegangen (vgl. ebd., S. 8 - 10):

- Interaktivität: Die Mitglieder beziehen sich in ihren Handlungen aufeinander. Ausschlaggebend ist nicht die Häufigkeit der Interaktion, sondern der Grad, also wie stark sie sich gegenseitig beeinflussen.

- Synchronität der Kommunikation: Diese wird vorausgesetzt, um die Möglichkeit zu haben, die Meinung einer anderen Person verändern zu können. Dabei ist es unerheblich, welches Medium zur Kommunikation genutzt wird. Im Bereich von internetgestützter Kommunikation gilt E-Mail-Kommunikation als asynchron, Chat dagegen als synchrones Kommunikationswerkzeug, da die Nachrichtenfrequenz zwischen den Kommunikator:innen sehr viel höher ist. Untersuchungen von Dillenbourg und anderen zeigten aber, dass nicht die technischen Spezifikationen eines Mediums definieren, ob Synchronität möglich ist. Vielmehr ist es eine metakommunikative Übereinkunft, dass eine Antwort erwartet wird, deren Inhalt sich auf die folgenden Handlungen auswirkt. Nicht die Technik entscheidet, sondern die sozialen Regeln der Gemeinschaft.

- Verhandelbarkeit: Im Gegensatz zu hierarchischen Situationen werden bei einer Kollaboration die Bedeutungen nicht aufgrund von Autorität festgelegt. Entscheidungen sind in einem bestimmten Rahmen durch alle verhandelbar. Standpunkte werden erklärt, begründet und es wird versucht, andere davon zu überzeugen. Dies setzt jedoch voraus, dass alle Kollaborierenden frei darin sind, wie sie miteinander interagieren und dass Gestaltungsraum bei der Aufgabenlösung existiert. Dillenbourg verweist in diesem Kontext auch auf die Bedeutung von Missverständnissen. Missverständliche Kommunikation zwingt die Kommunikator:innen dazu, ihre Standpunkte neu zu formulieren beziehungsweise sie anders zu erklären. All diese kommunikativen Akte ermöglichen Lernprozesse.

Kooperatives und kollaboratives Lernen

Kooperatives wie auch kollaboratives Lernen sind also Lernformen, die in einen sozialen Kontext eingebettet sind. Über die konkrete Definition herrscht ebenso Uneinigkeit wie über schon beschriebene Differenzierung der Begriffe.[34] Die Autor:innen heben jeweils unterschiedliche Aspekte in unterschiedlichen Gewichtungen hervor.

Für Reinmann-Rothmeier und Mandl ist kooperatives Lernen solches, das durch Interaktion mit anderen Lernenden einen Beitrag zu einer Problemlösung leistet. Dies geht einher mit Austausch von Erfahrungen, Ansichten und dem Kennenlernen neuer Perspektiven (vgl. Reinmann-Rothmeier & Mandl 1997, S. 110). Klaus Konrad hingegen definiert das Lernen im sozialen Kontext als Gruppenlernen, bei dem die Gruppenmitglieder nur im

[34] Nicht nur bei der Definition, auch bei der Bezeichnung herrscht Vielfalt. Gebräuchliche Begriffe in der Literatur für dieses Lernen sind auch: *peer collaboration*, *peer assisted learning*, *peer learning* oder Gruppenarbeit beziehungsweise Gruppenunterricht (vgl. Konrad 2014, S. 80).

Austausch und durch Diskussion mit anderen lernen. Dabei bauen sie ein gemeinsames Verständnis auf und erwerben dadurch neue Fähigkeiten und Fertigkeiten. Er verweist dabei auf die gemeinsame Schnittmenge zwischen kooperativem und kollaborativem Lernen. An beide Prozesse ist die Erwartung geknüpft, dass am Ende ein gemeinsames Produkt entsteht, welches mit Blick auf das individuellen Lernen der Gruppenmitglieder eine höhere Qualität erreicht (vgl. Konrad 2014, S. 81).

Wie dies genau erreicht werden kann, darüber gibt es eine Vielzahl von Studien, die unterschiedliche Gruppenkonstellation bei der Problemlösung miteinander verglichen haben. Eine Übersicht dazu liefern Johnson und Johnson (vgl. 2008, S. 19 - 26), die wesentliche Kriterien für kooperatives und kollaboratives Lernen zusammenfassen:

1. Positive Interdependenz: Diese resultiert aus gemeinsamen Zielen, komplementären Rollen oder unterschiedlichen Wissensbeständen.

2. Persönliche und kollektive Verantwortung: Verantwortlichkeit basiert auf der positiven Interdependenz. Sie entsteht, wenn sich die individuelle Leistung positiv auf die Arbeit anderer auswirkt. Dazu bedürfe es jedoch einer Messung und Anerkennung der individuellen Leistung, die der jeweiligen Person zurückgemeldet wird. Ähnlich verhält es sich mit der Gruppenverantwortung. Diese entstünde, wenn die kollektive Leistung mit der Leistung anderer Gruppen verglichen wird.

3. Gegenseitige Unterstützung: Die Lernenden wertschätzen die Leistung anderer zur Erreichung der Gruppenziele und unterstützen sie bei ihren Bemühungen. Dies kann durch direkte Hilfe, durch notwendige Informationen oder andere Ressourcen erfolgen. Auch konstruktives Feedback oder gegenseitige Infragestellung von Schlussfolgerungen und Überlegungen hilft der Gruppe, bessere Problemlösungen hervorzubringen.

4. Soziale Kompetenzen: Sie bilden die grundlegende Basis für die Beziehungen innerhalb der Gruppe. Ohne sie können die Individuen nicht produktiv zusammenarbeiten. Sie sind der Garant dafür, dass die Gruppe trotz Belastung handlungsfähig bleibt und die individuellen Beiträge zur Erreichung der Gruppenziele effizient koordinieren kann. Um dies zu vollbringen, braucht es folgende Bedingungen:

   a) sich gegenseitig kennen zu lernen und Vertrauen aufbauen zu können
   b) präzise, unmissverständliche Kommunikation
   c) gegenseitige Akzeptanz und Unterstützung
   d) Übereinkunft, dass Konflikte konstruktiv gelöst werden

5. Gruppenreflexion: Zur Verbesserung der gegenseitigen Unterstützung hat sich ein regelmäßiger Austauschprozess darüber bewährt, wie gut die Gruppe funktioniert und wie die gruppeninternen Prozesse verbessert werden können. Dazu zählt auch die Bewertung der Beiträge der Gruppenmitglieder, ob diese hilfreich oder nicht

hilfreich waren. Ziel ist es, die Effektivität der Mitglieder zu ermitteln, inwieweit deren Beiträge dazu geeignet sind, die gemeinsamen Gruppenziele erreichen.

Die hier aufgezeigten Merkmale für kooperatives und kollaboratives Lernen werden Ausgangspunkt für die Betrachtung der gemeinschaftlichen Lernprozesse in FLOSS-Gemeinschaften darstellen.

3.3.3 Computer Supported Collaborative / Cooperative Learning

Die folgenden Abschnitte thematisieren die technischen Werkzeuge, die für die Betrachtung von internetbasierten Lerngemeinschaften mit kooperativen und kollaborativen Lernformen wichtig sind.

Das Akronym CSCL wird meist in der Bezeichnung für *Computer Supported Collaborative/Cooperative Learning* verwendet. Diese Bezeichnung spiegelt jedoch nur einen Teil der in der Literatur gebräuchlichen Definitionen wieder. Manchmal steht das zweite 'C' auch für *conversational, coordinated, collective* oder gar *competitive*. Als kleinster gemeinsamer Nenner der Veröffentlichungen dazu kann CSCL als gemeinsames Lernen einer Gruppe oder Community bezeichnet werden, deren Mitglieder mit Hilfe der Nutzung von Informatiksystemen Wissen erwerben und aufbauen (vgl. Haake et al. 2012, S. 1f; Hinze 2008, S. 242).

Einschränkend verweist Hinze darauf, dass die Frage des Organisationsgrades einer Gruppe ein Kriterium für die Bezeichnung als CSCL ist. E-Learning bei formalen Lerngruppen wird schnell als CSCL betitelt. Hingegen bleibt es offen, ab welchem Zeitpunkt auch locker strukturiertes, informelles Lernen in virtuellen Communities unter CSCL zusammengefasst werden kann (vgl. Hinze 2008, S. 42). Ergänzend fügt Michael Kerres hinzu, dass auch Formen des informellen Lernens in sozialen Netzwerken im Internet oder der Wissensaustausch in thematischen Foren und Micro-Blogs zunehmend ins Blickfeld der CSCL-Forschung kommen (vgl. Kerres & Nattland 2012, S. 259f).

Formen von CSCL

Mittlerweile steht ein enormes Repertoire an technischen Möglichkeiten zur Verfügung, welche die Kooperation und Kollaboration zwischen Lernenden unterstützen können. Das Spektrum der Möglichkeiten reicht von einfachen, isolierten Werkzeugen für den Alltag wie zum Beispiel E-Mail oder Messenger-Dienste für die Gruppenkommunikation, bis zu hoch spezialisierten Softwareprodukten, die eigens für das CSCL mit Blick auf bestimmte Zielgruppen entworfen wurden. Solche speziellen CSCL-Umgebungen sind beispielsweise die Lernmanagementsysteme *Moodle* oder *ILIAS*, die oft in formalen Lernkontexten eingesetzt werden.[35] Diese vereinen verschiedene Werkzeuge der Kommunikation, Kooordination, Inhaltserstellung oder den Austausch von Lernmaterialien in einem

[35] Lernmanagementsystem ist eine Softwareplattform, die kooperations- und kollaborationsfördernde Werkzeuge (z.B. Kommunikationswerkzeuge) mit Lernmaterialien vereint (vgl. Ebner & Lorenz 2012, S. 110).

zusammenhängenden Lernmanagementsystem.

Welche Werkzeuge für ein bestimmtes Lernszenario zum Einsatz kommen, hängt von verschiedenen Kriterien ab. Nach Kerres und Nattland (vgl. 2012, S. 256 - 259) sind es folgende Punkte, die für die Auswahl entscheidend sein sollten:

- Ziel der Gruppenarbeit: Spontane Hilfestellung bei Schwierigkeiten erfordern andere Werkzeuge als die langfristige Zusammenarbeit sich selbst organisierender Gruppen wie zum Beispiel bei *learning communities*.
- Gruppengröße: Großgruppen erforderten andere Werkzeuge als kleine Teams.
- Arbeitsmodus: Arbeitet die Gruppe zeitgleich (synchron) oder zeitversetzt (asynchron)?
- Gruppenstruktur: Wie homogen oder heterogen ist die Gruppe bezüglich Hintergrundwissen, Kompetenzen, Sichtweisen, Gender oder Erfahrungen?
- Rollenstruktur: Wenn es für die Arbeit sinnvoll ist, sollten die Werkzeuge bestimmte Rollen innerhalb der Gruppe berücksichtigen können.
- Gruppenbildung: Wie wird die Gruppe zusammengestellt und wie können Änderungen der Gruppe durch die Werkzeuge abgebildet werden?
- Betreuungsmodus: Kann oder soll eine Betreuung der Gruppe möglich sein?
- Lernaufgabe: Ist die Art der Lernaufgabe dazu geeignet, die Zusammenarbeit der Gruppe fördern?

Die Klärung dieser Fragen liefern die ersten Anhaltspunkte, welche technischen Werkzeuge für eine geplante CSCL-Umgebung sinnvoll sind. Eine Übersicht über das Spektrum möglicher Werkzeuge würde allerdings den Rahmen dieser Arbeit übersteigen. Es kommen immer wieder neue Lernmanagementsysteme auf den Markt oder werden mit neuen Funktionen ausgestattet. Die Bandbreite möglicher Werkzeuge wächst stetig, auch wenn sich einige Werkzeuge als Standardwerkzeuge etablieren konnten, wie beispielsweise Kommunikationwerkzeuge wie persönliche Nachrichten, Chats, Foren oder gemeinsam genutzte Speicherorte für den Austausch von Dateien. Exemplarisch soll hier nur der Teilbereich der Kommunikation in seiner Komplexität kurz angerissen werden.

Dafür liefern beispielsweise Till Schümmer und Jörg Haake einen Überblick über Vor- und Nachteile der gebräuchlichsten Werkzeuge.[36] Sie vergleichen diverse Kommunikationswerkzeuge hinsichtlich der zeitlichen Dimension (synchron oder asynchron), Anzahl der Kommuniaktionsteilnehmenden, Medienarten, Persistenz der Nachrichten (flüchtig

[36] Folgende Kommunikationswerkzeuge werden miteinander verglichen: E-Mail, Mailinglisten, Newsgroup/Foren, Blog, Micro-Blog (z.B. Twitter), Instant Messenger, Konferenzsysteme und Soziale Netzwerke (vgl. Schümmer & Haake 2012, S. 88).

oder dauerhaft abrufbar), Informationsfluss (Push- oder Pull-Nachrichten[37]), Symmetrie (gleiche Kommunikationsrechte für alle oder moderierte Diskussion) und Offenheit (offene oder geschlossene Nutzer:innengruppen) (vgl. Schümmer & Haake 2012, S. 86 - 95).

So leistungsfähig diverse Werkzeuge und Lernmanagementsysteme mittlerweile auch sind: Sie sind sie nur eine von mehreren Voraussetzungen für eine gelingende Kooperation oder Kollaboration dar. Ein zentrales Problem bei vielen Umsetzungen von CSCL bleibt allerdings:

> „[...] vielen Plattformen gelingt es nicht, eine lebendige Community zu etablieren. [...] Zu bedenken ist, dass der Einsatz von CSCL-Werkzeugen beim formellen wie beim informellen Lernen als solches keineswegs sicherstellt, dass tatsächlich „Kooperation“ stattfindet. Ein systematisches und planvolles Vorgehen bei Auswahl und Einsatz von Werkzeugen für die computergestützte Gruppenarbeit ist unerlässlich, wenn man Kooperation beim Lernen anstrebt.“ (vgl. Kerres & Nattland 2012 S. 260)

Dieses Zitat verweist auf die vielen ambitioniert geplanten Lernmanagementumgebungen, deren Potential nicht aufgrund von fehlenden technischen Funktionen nicht ausgeschöpft wurden, sondern die wegen falscher Prioritätensetzung ihr Ziel verfehlten. Ursache ist oft eine nicht ausreichende Berücksichtigung der Rahmenbedingungen des Lernens in Bezug auf Sinn, Interesse und Situiertheit. Dies lässt sich auch nicht durch einen erhöhten technischen Aufwand kompensieren.

Virtuelle Communities und Lerngruppen

Der letzte Abschnitt zu internetgestütztem kooperativem Lernen widmet sich den virtuellen Lerngruppen beziehungsweise Gemeinschaften. Beide Begriffe liefern den analytischen Rahmen für die Betrachtung von Personen, die sich mit der Entwicklung einer konkreten FLOSS beschäftigen.

Eine Lerngruppe ist eine Gruppe, deren Mitglieder ein gemeinsames Ziel verfolgen: zu lernen, also Wissen zu erwerben. Ansonsten ist es eine soziale Gruppe, die sich von ihrer Umwelt abgrenzt, ein bestimmtes Zusammengehörigkeitsgefühl entwickelt und Gruppenprozesse wie zum Beispiel Kommunikation und Zusammenarbeit innerhalb und zwischen der Gruppe und ihrer Umgebung pflegt. Das Hauptziel einer Lerngruppe besteht jedoch im Erkenntnisgewinn, auch wenn das Ziel der Gruppenaktivität die Erstellung eines gemeinsamen geschaffenen Produktes ist (vgl. Wessner 2012, S. 200f).

Folgt man der Definition einer Community von Tobias Ley, verschwimmt die Grenze zwischen Lerngruppe und Community:

[37] Bei einem Pull-Nachricht muss die Zielperson selbst handeln, um den Empfang einer Nachricht zu registrieren. Im Gegensatz dazu erreicht eine Push-Nachricht die Zielperson ohne ihr Zutun. Ein Internetforum ist ein beispielsweise ein Pull-Medium, während ein Instant-Messenger oder eine SMS auf das Mobiltelefon ein Push-Medium darstellt.

> „Eine Community (oder 'Gemeinschaft') wird dabei als eine Gruppe von interagierenden Individuen betrachtet, die ein gemeinsames Interesse und gemeinsame Praktiken teilen, und in der es deshalb ein gewisses Maß an Kohäsion und ein Gefühl der Zusammengehörigkeit gibt." (Ley et al. 2012, S. 261)

Dies deckt sich mit der Definition von virtuellen Lerngemeinschaften von Nicola Döring, die oft als Referenz genutzt wird:

> „Eine virtuelle Gemeinschaft ist ein Zusammenschluss von Menschen mit gemeinsamen Interessen, die untereinander mit gewisser Regelmäßigkeit und Verbindlichkeit auf computervermitteltem Wege Informationen austauschen und Kontakte knüpfen. [...] Unter virtuellen Lerngemeinschaften sollen hier virtuelle Gemeinschaften verstanden werden, in denen der Erwerb von Wissen oder Fertigkeiten für die Beteiligten explizit im Vordergrund steht." (Döring 2001)

Als Beispiele für solche Lerngemeinschaften führt sie öffentliche Wissensbörsen, Berufund fachbezogene Foren, unternehmensinternes Wissensmanagement oder kursbezogene Foren an. Der Unterschied zur Lerngruppe besteht darin, dass der Zusammenhalt in der Community nicht so stark ist und die Teilnahme oft freiwillig erfolgt (vgl. Ley et al. 2012, S. 261). Udo Hinze beschreibt virtuelle Communities als informelle, virtuelle Lerngruppen, mit einem geringeren Grad an Verbindlichkeit und Zusammenhalt, als dies bei klassischen Lerngruppen der Fall ist (vgl. Hinze 2008, S. 242).

Als Tenor kann also festgehalten werden, dass der Übergang von einer Lerngruppe zur Lerngemeinschaft beziehungsweise einer *learning community* nicht als trennscharf angesehen werden kann. Während die Zuordnung von formalen Lerngruppen zu CSCL noch relativ leicht fällt, verschwimmt die Grenze zur Community je nach Art der Interessen und Verbindlichkeiten der einzelnen Lernenden. Dementsprechend sind die FLOSS-Communities auch nicht klar den virtuellen Lerngemeinschaften oder den virtuellen Lerngruppen zuzuordnen. In Abhängigkeit von den Interessen ihrer jeweiligen Mitglieder und ihrer individuell unterschiedlich stark empfundenen Verbindlichkeiten kann sogar innerhalb der Mitglieder eines Projektes die Zuordnung zu beiden Formen möglich sein.

3.3.4 FLOSS-Entwicklungsgemeinschaften als virtuelle Lerngemeinschaften

Anhand der Typologie des E-Learnings von Rheinmann-Rothmeier (vgl. Kapitel 3.3.1) können Entwicklungsgemeinschaften der FLOSS-Bewegung als E-Learning mit Schwerpunkt auf kollaborative Arbeitsweisen betrachtet werden. In der jüngeren Forschungsliteratur werden solchen Formen des Lernens als computergestütztes kooperatives beziehungsweise kollaboratives Lernen bezeichnet. Beide Formen der Arbeitsweisen können in Entwicklungsgemeinschaften zeitgleich vorkommen oder in deren Verlauf umschla-

gen, was bei der Beschreibung von einer konkreten FLOSS-Entwicklungsgemeinschaft in Kapitel 6.5 dieser Arbeit gezeigt wird.

Ebenso die Frage, ob es sich bei internetgestützten Entwicklungsgemeinschaften um explizite Lerngruppen oder Lerngemeinschaften handelt, muss anhand der Interessen der konkret beteiligten Entwickler:innen und der Ausprägung ihrer Gruppenidentität entschieden werden. Auch die Unterscheidung zwischen Lerngruppen und Lerngemeinschaften ist nicht trennscharf. Als wesentlicher Unterschied zwischen beiden Formen gilt eine geringere Gruppenkohäsion bei Lerngemeinschaften im Vergleich zu Lerngruppen. Wird von der Prämisse ausgegangen, dass Lernen zum großen Teil informell stattfindet, so kann jede auf Gegenseitigkeit beruhende Praxis einer Gruppe oder Gemeinschaft demzufolge als implizite Lerngruppe beziehungsweise als eine implizite Lerngemeinschaft bezeichnet werden. Wenn im weiteren Verlauf der Arbeit von Entwicklungsgemeinschaften gesprochen wird, werden darunter virtuelle Lerngemeinschaften verstanden, die dem Ansatz der Communities of Practice beziehungsweise Praxisgemeinschaften folgen.

3.4 Zusammenfassung

In diesem Kapitel wurde ein Überblick über wichtige Lerntheorien gegeben. Die Beschreibung zielte darauf ab, den Bezug herauszuarbeiten, den die einzelnen Theorien zum Lernen im sozialen Kontext haben. Lerntheoretische Ansätze, die sich auf den Sozialkonstruktivismus beziehen, bilden für die Forschung in dieser Arbeit den zentralen Bezugsrahmen. Dies sind die *Communities of Practice* und Aspekte, die dem *Konnektivismus* zugeordnet werden. Lernen in internetgestützten FLOSS-Entwicklungsgemeinschaften wurde in die Typologie des E-Learnings eingeordnet und es wurde gezeigt, wie diese Form des Lernens als virtuelle Lernnetzwerke beschrieben werden können. In solchen Lernumgebungen findet das Lernen informell, problemorientiert und kooperativ, wie auch kollaborativ statt.

Um einen umfassenden Lernbegriff für diese Arbeit abzuleiten, fehlt noch die subjektorientierte Perspektive des Lernens. Zu diesem Zweck wurde die Bedeutung der beiden Dimensionen Interesse und Situiertheit von Lernprozessen erörtert. Lernen hat die Möglichkeitserweiterung zum Ziel. Dies schlägt sich auch in dem Fähigkeitenansatz von Nussbaum nieder. Sie ist der Frage nachgegangen, zu was Lernen befähigen muss, um ein *gutes Leben* zu führen. Der Fähigkeitenansatz dient in Kapitel 11.5 bei der Beschreibung von Lernen in FLOSS-Projekten als Richtschnur, um die Potenziale für den Fähigkeitenerwerb beim Engagement in FLOSS-Entwicklungsgemeinschaften auszuloten.

Diese theoretische Betrachtung von Lernen ist aber nicht nur für die Bewertung des Lernens in solchen Lernumgebungen notwendig, sondern auch für die Konstruktion der Erhebungswerkzeuge ist sie von erheblicher Bedeutung. Die Praxis einer Entwicklungsgemeinschaft soll einmal aus individueller Perspektive, also aus Sicht von Entwickler:innen, und aus der reinen Außenbetrachtung der projektweiten Kommunikation beschrieben werden. Interesse am Lerngegenstand und Situiertheit des Lernens sind für die Schwer-

punktsetzung bei der Konstruktion des Leitfadens für die Befragung der Entwickler:innen entscheidend. Ebenso die individuelle Wahrnehmung der gemeinschaftlichen Handlungen im Sinne einer Community of Practice. Durch die Analyse der Kommunikation in einem FLOSS-Projekt kann gezeigt werden, wie Kooperation und Kollaboration kommunikativ koordiniert wird und wie die Bedeutungsaushandlung in einer Praxisgemeinschaft vonstatten gehen.

4 Forschungsstand zu FLOSS-Projekten

Die wissenschaftliche Auseinandersetzung mit FLOSS-Projekten setzt erst mit Beginn der Jahrtausendwende ein. Dies ist insofern beachtlich, als dass die kollaborative Entwicklung von GNU/Linux schon 1983 begann, als Stallman die Idee eines freien Betriebssystems in einer Usenet-Newsgroup[38] veröffentlichte und zur Mitarbeit aufrief. Eine Kultur des Teilens und Unterstützens bezüglich Software gab es jedoch schon weit vorher, bevor es überhaupt die Begriffe *Freie Software* und *Open-Source* gab.

Stallmann berichtet von einer *software-sharing community*, die er 1971 bei seiner Arbeit im Artificial Intelligence Lab am Massachusetts Institute of Technology kennenlernte. Diese tauschte mit Angestellten anderer Universitäten und Firmen den Quellcode für Betriebssysteme und Programme der damaligen Großrechner untereinander aus und verbesserten sie in Eigenregie. Das war nicht eine Ausnahme, sondern die Regel zur damaligen Zeit. Wer Interesse an einem Programm hatte, bekam auf Nachfrage dessen Quellcode, ohne dass andere Gegenleistungen dafür verlangt wurden. Erst in den 1980er Jahren wurden von der Computer- und Softwareindustrie proprietäre Software ausgeliefert, deren unerlaubte Weitergabe vertraglich verboten wurde (vgl. Stallmann 2015, S. 9f). Der Austausch innerhalb der Community erfolgte damals schon elektronisch über E-Mail, Newsletter, Usenet und Konferenzen (vgl. Khartitoniouk & Stewin 2004, S. 5).

Doch erst das Aufkeimen freier Softwarelizenzen, die Verbreitung erschwinglicher IT-Hardware und die zunehmende Bedeutung des Internets rückten diese Tausch- und Lerngemeinschaften in den Fokus der Wissenschaft. Einen Überblick über den damaligen Stand der Forschung lieferte beispielsweise Krogh und Hippel. Sie fassten Studien zusammen und ordneten sie nach drei Kategorien: (1) Motivation der Entwickler:innen für ihr Engagement, (2) Struktur und Koordination von FLOSS-Projekten sowie (3) kompetitive Dynamik, durch die Wechselwirkung von FLOSS im Umfeld von proprietärer und kommerzieller Software herbeiführt (vgl. Krogh & Hippel 2006, S. 977). Eine andere Zusammenfassung des Forschungsstandes stammt von Holtgrewe (2004), die allerdings nur die Entwickler:innen im Fokus hat und ebenfalls mit dem aus dem Behaviorismus stammenden Begriff der Motivation arbeitet. Mit Blick auf die vielen Arbeiten auf diesem Gebiet sollen im Folgenden nur die relevanten Studien kurz wiedergegeben werden, die sich mit dem Forschungsschwerpunkten dieser Arbeit decken.

Zu diesem Zweck werden zuerst die Besonderheiten bei der FLOSS-Entwicklung im Vergleich mit der Entwicklung von proprietärer Software dargestellt. Es werden Erkenntnisse über die Strukturen und Prozesse von FLOSS-Entwicklungsgemeinschaften vorgestellt und die wichtigsten Werkzeuge für die gemeinschaftliche Entwicklungsarbeit beschrieben.

Im darauffolgenden Unterkapitel wird der Kenntnisstand über die Kommunikation in-

[38] Das Usenet ist ein forenbasiertes Diskussionsnetzwerk, welches als eines der ältesten weltweiten Internetdienste schon vor dem heutigen World Wide Web existierte und auch heute noch betrieben wird. Das Netzwerk ist in viele thematische Foren strukturiert, wo die Teilnehmenden öffentlich diskutieren.

nerhalb von Entwicklungsgemeinschaften vorgestellt. Hier gibt es noch erheblichen Forschungsbedarf, weshalb hier nur die wichtigsten Studien erwähnt werden. Aus diesem Grund wird der Kommunikation im empirischen Teil dieser Forschungsarbeit viel Raum gegeben, um den Stand der Forschung an dieser Stelle zu bereichern.

Das letzte Unterkapitel gibt die Kenntnisse über die Entwickler:innen wieder. In diesem Bereich sind bereits viele quantitative Studien durchgeführt worden. Sie hatten zum Ziel, die Interessen der Entwickler:innen in Erfahrung zu bringen, insbesondere weshalb diese sich in FLOSS-Projekten engagieren und was sie dabei lernen. Weitere Studien befassten sich auch mit den Lernprozessen selbst. Sie ermöglichen erste Erkenntnisse darüber, wie Entwickler:innen während der Entwicklungsarbeit lernen.

4.1 Struktur und Prozesse von FLOSS-Gemeinschaften

FLOSS-Gemeinschaften können sich in ihren Strukturen und Prozessen stark voneinander unterscheiden. Viele FLOSS-Projekte starten als Idee einer Person, die im Laufe der Zeit immer mehr Menschen davon begeistern kann. So wächst ein Projekt, was anfänglich nur von einer Person repräsentiert wurde, zu einem Gemeinschaftsprojekt mit Aufgabenteilung, sozialen Beziehungen und Austauschprozessen und einem differenzierten Instrumentarium an gemeinschaftlichen genutzten Werkzeugen heran.

4.1.1 Struktur der Entwicklungsgemeinschaft

Die Gruppenstruktur in einem FLOSS-Projekt beinhaltet verschiedene Rollen und Aufgabenbereiche. Eine stark vereinfachte Darstellung bietet das folgenden Schalenmodell.

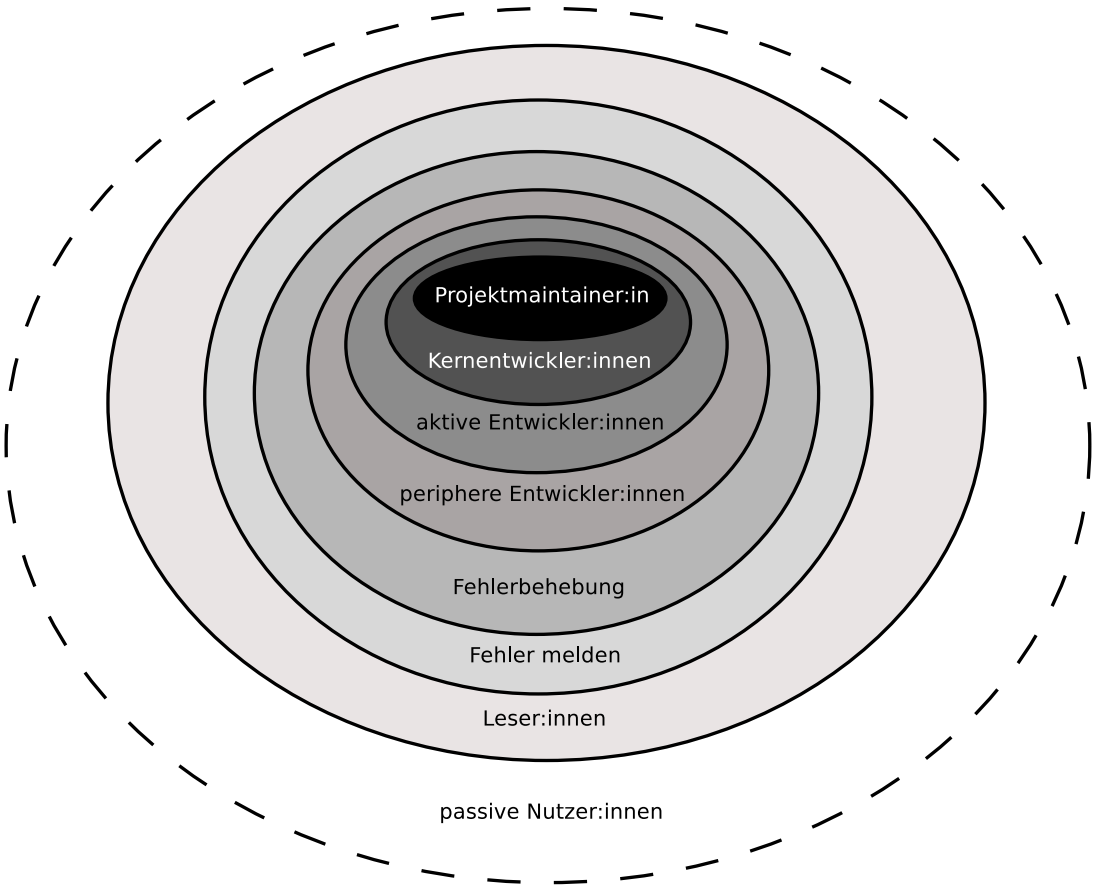

Abbildung 1: Modell der Gruppenstruktur in einem FLOSS-Projekt (Darstellung nach Nakakoji et al. 2003; Anpassung: K.M.)

Es geht von einer zentralisierten Struktur aus. Im Kern des Projektes steht der:die Projektmaintainer:in, welche:r im Regelfall das Projekt initiierte und maßgeblich die Entwicklungsrichtung vorgibt und die Entwicklung koordiniert. Zu dieser Person stehen die Kernentwickler:innen in engem Kontakt. Sie koordinieren einzelne Elemente vom FLOSS-Projekt und haben über einen langen Zeitraum bedeutende Beiträge zu dem Projekt geleistet. Sie sorgen teilweise auch für die Integration der Software in eine bestimmte Softwareumgebung, beispielsweise für eine spezielle GNU/Linux-Distribution oder ein bestimmtes Betriebssystem. Um sie herum gibt es die aktiven Entwickler:innen, die regelmäßig neue Funktionen beitragen und Fehler beheben. Die nächste Ebene wird von den peripheren Entwickler:innen gebildet. Diese beteiligen sich nur gelegentlich mit neuen Funktionen oder Fehlerkorrekturen an der Entwicklungsarbeit. Ihre Beiträge sind unregelmäßig und sie engagieren sich in der Entwicklungsgemeinschaft nur sporadisch und über kurze Perioden. *Bug Fixer* beziehungsweise Community-Mitglieder, die sich mit der Behebung von Fehlern beschäftigen, korrigieren selbst gefundene Fehler oder Fehler, die von anderen gemeldet wurden. Sie müssen nur den Teil vom Quellcode lesen und verstehen können, der die Ursache des Fehlers ist. *Bug Reporter* dagegen beschränken ihr Engagement auf das Melden von Fehlern. Sie überlassen die Korrektur anderen, da sie selbst oft nicht die nötigen Kompetenzen dafür haben. Sie erfüllen die gleiche Funktion wie Softwaretester:innen im traditionellen Modell der Softwareentwicklung. Leser:innen

nutzen die FLOSS selbst und versuchen die internen Strukturen und Funktionsweise zu verstehen beziehungsweise informieren sich über den Entwicklungsfortschritt. Passive Nutzer:innen verwenden lediglich die FLOSS in derselben Art und Weise, wie die meisten Menschen Software verwenden. Sie sind nicht direkt Teil der Entwicklungsgemeinschaft, erfüllen aber dennoch eine wichtige psychologische Funktion. Für viele Entwickler:innen ist eine hohe Zahl an Nutzer:innen eine Form der Anerkennung für ihre Entwicklungsarbeit. Auch sorgen sie für die Verbreitung der Software und ziehen dadurch potenzielle Entwickler:innen an (vgl. Nakakoji et al. 2003, S. 7).

Die Einflussmöglichkeiten auf das Projekt nehmen beim Schalenmodell mit jeder Schicht nach außen hin ab. Passive Nutzer:innen besitzen keine direkten Einflussmöglichkeiten. Die Schichten sind allerdings nicht starr: Je mehr sich eine Person in der Gemeinschaft engagiert und qualitativ hochwertige Beiträge leistet, desto enger kann sie Richtung Zentrum vorrücken. Diese Möglichkeit in solch einer meritokratischen Struktur ist für alle offen. Insofern sind FLOSS-Projekte nicht hierarchiefrei strukturiert.

Das Schalenmodell kann von Projekt zu Projekt variieren: Nicht alle Schalen müssen existieren, Entwickler:innen wechseln ihre Positionen im Modell oder können nicht klar zugeordnet werden. Dies spricht für eine hohe Durchlässigkeit der Schichten. Ebenfalls kann die Gemeinschaft mehrere Zentren aufweisen, deren Kernentwickler:innen in bestimmten Aufgabengebieten die Rolle der Projektleitung ausüben. Besonders in großen FLOSS-Distributionen wie zum Beispiel Debian, haben sogenannte *Paket-Maintainer:innen* einen besonderen Stellenwert in der Projekthierarchie. Ebenso sind auch die Bereiche Übersetzung in mehrere Sprachen sowie die Dokumentation der Software in vielen Projekten von entscheidender Bedeutung. Diese wichtigen Bereiche können auch ohne Programmierkenntnisse von Projektmitgliedern weiterentwickelt werden (vgl. Mikkonen et al. 2007).

4.1.2 Werkzeuge von Entwicklungsgemeinschaften

Die beiden organisatorischen Kernaufgaben für die FLOSS-Entwicklung ist die Zusammenführung der individuellen Beiträge der Entwickler:innen und die Koordination der Aufgaben. Die dezentrale, kollaborative Arbeit erfordert einen Speicherort für den Programmcode und dessen Dokumentation. Weiterhin braucht es ein gemeinschaftliches Kommunikationsmedium, womit alle Entwickler:innen ihre individuellen Beiträge koordinieren können. Darüber hinaus gibt es noch viele andere genutzte Werkzeuge, die jeweils einzelne Prozesse der gemeinschaftlichen Entwicklungsarbeit effizienter gestalten können. Jede FLOSS-Gemeinschaft schafft sich ihre eigenen Strukturen und Prozesse, die den Erfordernissen und Bedürfnissen der Entwickler:innen und Nutzer:innen gerecht werden. Die Wichtigsten werden hier kurz dargestellt.

Quellcode-Management

Die Versionsverwaltung ist der Ort, an dem die Beiträge der Entwickler:innen ge-

speichert werden. Dies ist in erster Linie der Quellcode vom Softwareprojekt, also die eigentliche Essenz eines FLOSS-Projektes. Aber auch die Dokumentation, Gebrauchsanleitung oder die Übersetzungen sind alles Erzeugnisse der kollaborativen Zusammenarbeit, die sich stetig weiterentwickeln. So entsteht schnell ein komplexes Gebilde verschiedener Dateien, die von diversen Entwickler:innen kontinuierlich modifiziert werden. Die Versionsverwaltung ist dabei das Schlüsselelement, um diese Prozesse nachvollziehbar zu halten. Sie archiviert die Beiträge der Entwickler:innen und kann im Bedarfsfall einen früheren Entwicklungsstand wiederherstellen. Dadurch wird der Entwicklungsfortschritt transparent archiviert und durch die Vergabe von Versionsnummern strukturiert. Mittels automatisierten Änderungsprotokollen ist es nachvollziehbar, wann welche Person Änderungen am FLOSS-Projekt durchgeführt hat (vgl. Ettrich 2004, S. 188). Dies ist die Kernfunktionalität einer Versionsverwaltung. Auf dem Markt existieren eine Vielzahl unterschiedlicher Systeme, die sich in Funktionsumfang und Nutzungskonzepten unterscheiden. Oft genutzte Versionsverwaltungssysteme mit einer freien Lizenz sind beispielsweise *Git*, *Subversion* oder *Mercurial*.

Die Versionsverwaltung wird initial durch den:die Projektinitiator:in eingerichtet und mit ersten Inhalten befüllt. Auf diese haben dann interessierte Entwickler:innen Zugriff. Haben sie Verbesserungen vorgenommen, können sie diese entweder direkt an den:die Maintainer:in weiterleiten oder diese über die Versionsverwaltung einreichen und zur Integration ins FLOSS-Projekt vormerken lassen. Der:die Maintainer:in entscheidet dann über die Aufnahme des eingereichten Beitrags in den Hauptentwicklungsstrang des FLOSS-Projektes. Diese Arbeit kann auch durch Zuweisung von weiteren Schreibrechten in der Versionsverwaltung an andere Entwickler:innen übertragen werden. Dies würde dann im Schalenmodell der Transition von peripheren Entwickler:innen hin zu Kernentwickler:innen entsprechen.

Kommunikationskanäle

Die zweite wichtige Infrastruktur ist die der Kommunikation. Sie findet in vielfältigen Formen statt. Zwischen direkter Kommunikation im persönlichen Gespräch oder als Textnachricht bis hin zu Formen der indirekten Kommunikation über die Homepage des FLOSS-Projektes. Hier soll der Fokus auf die technisch vermittelte Kommunikation gesetzt werden. Damit soll nicht die Wichtigkeit des direkten Austauschs zwischen Entwickler:innen in persönlichen Gesprächen negiert werden. Mit zunehmender Größe und geographischer Verbreitung der FLOSS ist anzunehmen, dass technisch vermittelte Kommunikation an Bedeutung gewinnt. Aus diesem Grund sollen hier nur die wichtigsten Kommunikationskanäle kurz erwähnt werden. Bei der Kommunikation ist zwischen synchroner und asynchroner Kommunikation zu unterscheiden. Darüber hinaus ist auch die Persistenz und der Grad an Öffentlichkeit ein wichtiges Kriterium für die Wahl der Kommunikationswerkzeuge (vgl. Schümmer & Haake 2012, S. 86 - 95).

Die wichtigsten asynchronen Werkzeug der Kommunikation zwischen den Entwickler:innen sind nach wie vor Mailinglisten. Die E-Mails können zu verschiedenen Zeiten

geschrieben und empfangen werden. Dies ist besonders für geographisch weit verteilte FLOSS-Gemeinschaften von Vorteil. Das Archiv der verschickten E-Mails von einer Mailingliste dient gleichzeitig als archivierte Wissenssammlung. Mailinglisten als persistentes Medium ermöglichen es so neuen Entwickler:innen, sich über bereits diskutierte Themen und Kommunikationsgewohnheiten der FLOSS-Gemeinschaft zu informieren (vgl. Ettrich 2004, S. 186f)). Neben Mailinglisten gibt es auch Foren als asynchrone Kommunikationskanäle. Hier tritt in besonderer Weise *StackOverflow* hervor, welches als Frage-Antwort-Forum für viele Programmierer:innen als Anlaufstelle bei Problemen dient. Da Foren meist von kommerziellen Unternehmen betrieben werden, verfügen diese über die Infrastruktur und kontrollieren auch die Inhalte. E-Mail und Mailinglisten nutzen dagegen offene Protokolle und setzen auf ein globales Distributionsnetzwerk (vgl. Squire 2017, S. 4f).

Als synchrone Kommunikationskanäle dienen Chat und Instant-Messenger. Diese ermöglichen Kommunikation in Echtzeit und werden normalerweise nicht archiviert.[39] Häufig wird hierbei der offene *Internet Relay Chat (IRC)* genutzt. In jüngerer Zeit werden aber auch proprietäre, also auf nicht freien Protokollen basierende Chat-Dienste eingesetzt, wie beispielsweise Slack. Dies ist insofern interessant, weil sich diese proprietären Kommunikationskanäle der Kontrolle der Nutzer:innen entziehen und damit den Ideen von Freier und offener Software widersprechen (vgl. ebd., S. 5f).

FLOSS-Entwicklungsportale

FLOSS-Entwicklungsportale bündeln verschiedene Entwicklungswerkzeuge auf einer Internetplattform. Dies senkt die Hürden für Projektinitiator:innen, da sie sich nicht mit der Administration und Konfiguration der Arbeits- und Kommunikationsinfrastruktur befassen müssen. Außerdem kann das FLOSS-Projekt potentiellen Nutzer:innen und vor allem Entwickler:innen präsentiert werden. Dies ist besonders in der Startphase vom Projekt wichtig.

Eines der größten Portale mit circa 31 Millionen Entwickler:innen ist mittlerweile *GitHub* (vgl. Warner 2018). Es basiert auf dem Versionsverwaltungssystem *Git* und vereint verschiedene Werkzeuge zur Kommunikation, zum Projektmanagement und zur Präsentation von FLOSS- und proprietäre Projekten. Dies sind projekteigene Wikis, Bug-Tracker, Wunschlisten für neue Funktionen, Aufgabenkoordination und E-Mail-Benachrichtigungen, wenn auf eigenen oder fremden Repositorien Änderungen erfolgt sind. Bei GitHub stehen die Entwickler:innen mit ihren Repositorien im Mittelpunkt und nicht die unterschiedlichen Projekte. Für die Kollaboration zwischen den Entwickler:innen ist GitHub mit typischen Kommunikationselementen versehen, wie sie auch bei anderen sozialen Netzwerken zu finden sind. Dieser Strukturansatz funktioniert dergestalt, dass Ent-

[39] Technisch ist dies aber problemlos möglich und wird von einigen FLOSS-Gemeinschaften auch durchgeführt. Beispielsweise verfügt die FLOSS-Distribution Ubuntu über ein Chat-Archiv mit ca. 300 IRC-Kanälen was bis in das Jahr 2004 zurück reicht und unter http://irclogs.ubuntu.com abgerufen werden kann.

wickler:innen sich Kopien von Repositorien einer FLOSS anfertigen, die von den Projektinitiator:innen erstellt wurden. Darin werden Änderungen vorgenommen und anschließend wird die:der Maintainer:in darüber informiert, dass eine Modifikation erfolgt ist. Der:die Maintainer:in entscheidet dann, ob die Modifikation in das Ursprungsrepositorium übernommen wird.

Bei *SourceForge* stehen dagegen die einzelnen Projekte im Mittelpunkt. Ein Projekt bedeutet ein Repositorium, in dem alle schreibberechtigten Entwickler:innen Änderungen durchführen können. Das Portal bietet den Projekten mehrere Typen von Versionsverwaltungssystemen, Mailinglisten, Foren, Wikis, Micro Blogs und Bug Tracker. Circa 502.000 Projekte sind zurzeit auf SourceForge gehostet (vgl. SourceForge o.D.). Ähnlich wie GitHub bietet auch Sourceforge als Teil eines kommerziellen Unternehmens seine Dienstleistung für proprietäre Softwareprojekte an.

4.1.3 Lebenszyklus eines FLOSS-Projektes

Um die Besonderheiten im Lebenszyklus eines FLOSS-Projektes zu betrachten, soll zuerst der Weg von proprietärer Software dargestellt werden. An den Unterschieden lassen sich die Besonderheiten der Softwareentwicklung von FLOSS gut demonstrieren. Das Wasserfallmodell von proprietärer Softwareentwicklung soll hierfür als Basis dienen. Dieses gibt es in unterschiedlichen Variationen und Detailtiefen. Es besteht aus einer Kette von Schritten, die nacheinander durchlaufen werden und so den Produktlebenszyklus von Software in fünf Phasen einteilen (vgl. Capiluppi et al. 2007, S. 3):

1. Initiale Entwicklung: Analyse der Anforderungen an die Software, Projektentwurf und der erste Programmcode einschließlich seiner Tests werden in dieser ersten Phase durchgeführt. Veröffentlichungen einer Programmversion für Nutzer:innen gibt es noch nicht.

2. Evolution: Die Software wird verbessert und mit neuen Merkmalen und Funktionen ergänzt. Programmversionen werden erstmals veröffentlicht. Das Feedback der Nutzer:innen dient als Grundlage für weitere Änderungen, die wiederum als Programmaktualisierungen veröffentlicht werden. Die Phase dauert an, bis die Kosten der Programmerweiterung den Gewinn durch Verkauf übersteigen.

3. Wartungsphase: Das Programm ist ausgereift. Änderungen werden noch in der Codebasis durchgeführt, aber es gibt keine Erweiterungen mehr. Nur noch einzelne Fehlerkorrekturen werden ausgeliefert.

4. Wartungsabbruch: Es werden keine Änderungen am Programm zur Fehlerkorrektur mehr durchgeführt. Der Wartungsabbruch ist als Ende der Entwicklungstätigkeit zu verstehen. Dies geht meist mit der Präsenz einer neuen Software einher, welche die alte Software zukünftig ersetzen soll.

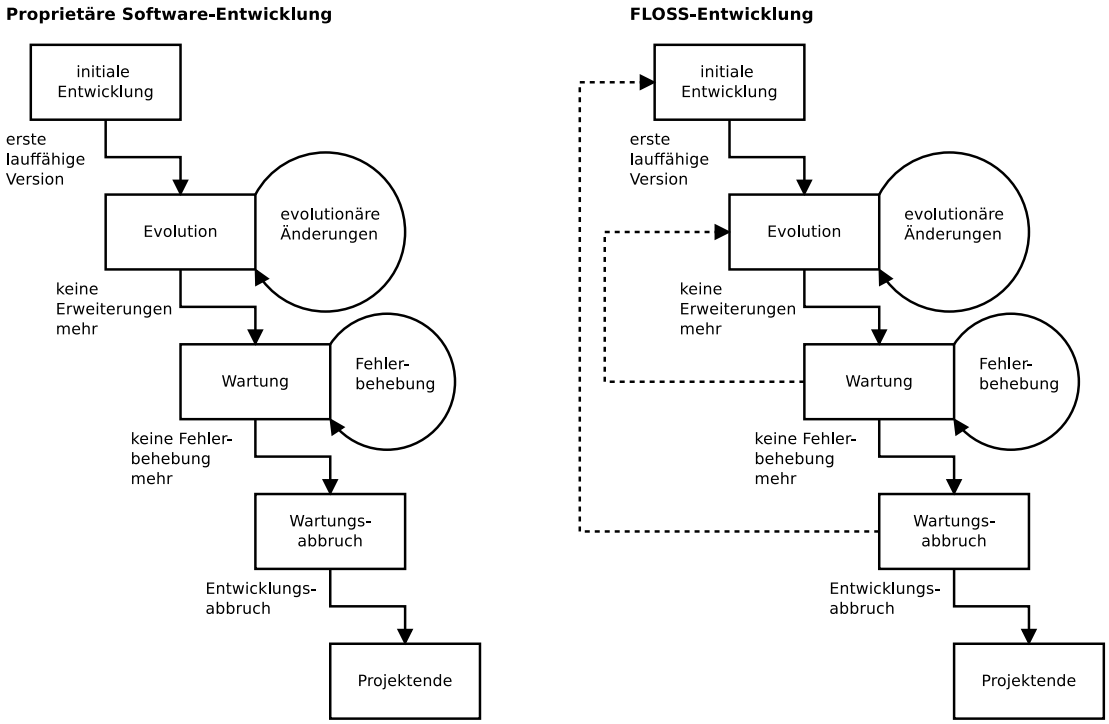

Abbildung 2: Wasserfallmodell der Softwareentwicklung (Darstellung nach Capiluppi et al. 2007, S. 3, 6; Anpassung: K.M.)

5. Projektende: Das alte Programm dient möglicherweise noch als Grundlage für eine neue Software.

Capiluppi et al. haben hierfür acht FLOSS-Projekte im Längsschnitt untersucht und dabei Abweichungen vom Lebenszyklus proprietärer Software entdeckt. Typischerweise wird proprietäre Software erst verkauft, wenn eine erste offizielle lauffähige Version existiert, die den in der Designphase formulierten Anforderungen entspricht. Im Unterschied dazu wird FLOSS in einem Zustand der permanenten Veröffentlichung entwickelt. Dies bedeutet, dass jederzeit unbeschränkter Zugriff auf die Codebasis möglich ist. Mit Blick auf diese Eigenart der FLOSS-Entwicklung konnte festgestellt werden, dass viele FLOSS-Projekte die Phase der initialen Entwicklung nie verlassen haben. Die zweite Abweichung betrifft die Verknüpfung zwischen Evolutions- und Wartungsphase. In manchen Projekten werden in einer begrenzten Zeitspanne vor der Veröffentlichung einer neuen Programmversion keine neuen Funktionen implementiert. In dieser Zeit befindet sich das Projekt in einer Wartungsphase. Nach der Veröffentlichung arbeitet die Entwicklungsgemeinschaft wieder in einem Modus der evolutionären Phase. Die dritte Abweichung vom Entwicklungszyklus zu proprietärer Software zeigt sich in der Möglichkeit, dass das FLOSS-Projekt nach einem Projektende durch neue Entwickler:innen wieder in die Phase der initialen Entwicklung gelangen kann. Solch eine Wiederbelebung ist möglich, da der Quellcode frei zugänglich ist und dadurch für weitere Projekte erneut genutzt werden kann (vgl. Capiluppi et al. 2007, S. 3).

Der Vergleich der beiden Lebenszyklen von proprietärer Software und FLOSS zeigt bei der FLOSS-Entwicklung viele Möglichkeiten auf, den linearen Entwicklungspfad zu

verlassen. Durch neue Entwickler:innen oder eine Änderung der Ressourcen bei den aktuellen Entwickler:innen kann ein zuvor stagnierendes Projekt wieder eine Weiterentwicklung erfahren.

4.2 Studien zur Kommunikation in FLOSS-Gemeinschaften

Wie bereits erwähnt, nutzen FLOSS-Communities viele verschiedene Kommunikationskanäle. Dabei stehen die asynchronen Kommunikationsmittel mit öffentlich einsehbaren Archiven im Fokus der Forschung. Die Kommunikation auf projektspezifischen Mailinglisten ist oftmals von einer kleinen Zahl von Entwickler:innen dominiert. 80% der Beiträge stammen von nur 10 bis 15% der Entwickler:innen (vgl. Hannemann et al. 2012, S. 29). Diese Unterschiede innerhalb der FLOSS-Gemeinschaft wurden auch in vielen anderen Projekten beobachtet. Ein derartiges Ungleichgewicht muss aber nicht zwingend ein Nachteil für die beteiligten Entwickler:innen sein. Es konnte gezeigt werden, dass mit steigender Ungleichheit auch der Wissensaustausch steigt. Eine Gruppe mit homogenen Kenntnissen ist diesbezüglich kontraproduktiv. Ebenso wurde eine Korrelation zwischen Wissensaustausch und der diskursiven Interaktivität gefunden. Je mehr konträre Meinungen und Sichtweisen kommuniziert werden, desto eher fördert dies den Erwerb und die Weitergabe von Wissen (vgl. Kuk 2006, S. 1038). Dies unterstreicht den Stellenwert der Kommunikation in einem FLOSS-Projekt für den individuellen Aufbau von Kompetenzen und wie wichtig eine offene Kommunikationskultur ist. Sie ist eine Voraussetzung dafür, dass neue beziehungsweise unerfahrene potentielle Entwickler:innen sich an der Entwicklungsarbeit beteiligen können.

Eine Studie von Sowe et al. konnte zeigen, dass Entwickler:innen bei der Nutzung der Projekt-Mailingliste kommunikative Rollen einnehmen.[40] Entweder sie antworten auf Fragen anderer oder sie stellen hauptsächlich selbst Fragen und bitten um Antworten. Die Personen die eher Antworten geben, wurden als langfristig Engagierte identifiziert, denen mutmaßlich viel Erfahrung und fundierte Kenntnisse unterstellt wurden.[41] Komplementär dazu sind Fragesteller:innen erst kurzzeitig in dem FLOSS-Projekt aktiv und besitzen weniger Erfahrung (vgl. Sowe et al. 2007, S. 443).

Nils Taubert untersuchte in seiner Forschungsarbeit, wie Zielkonflikte in FLOSS-Projekten gelöst werden. Er führte eine Beitragsanalyse der öffentlich zugänglichen Mailinglisten der Entwickler:innen durch und befragte einige davon per Leitfadeninterview. Dabei analysierte er die Kommunikationstechniken, derer sich die Entwickler:innen in einem FLOSS-Projekt in einem Zeitraum von mehreren Jahren hinweg bedienten.

Er stellte fest, dass aufgrund der fehlenden formalen Hierarchie Konflikte argumentativ

[40] Die Ergebnisse stammen von einer statistischen Auswertung von einer Entwickler:innen-Mailingliste vom Debian-Projekt mit einem Zeitraum von fünf Jahren. Ausgewertet wurden die Metadaten: E-Mail-Sender:in, Thread-Eröffnung (als Frage interpretiert) oder Thread-Antwort (vgl. Sowe et al. 2007, S. 436f).

[41] Genaue Angaben konnten die Autor:innen der Studie nicht machen, da nur die Metadaten der Kommunikation ausgewertet wurden ohne den Inhalt zu berücksichtigen.

ausgehandelt werden. Konfliktfelder sind in der Regel Entwicklungspfade oder Funktionalitäten der FLOSS. Das Fehlen einer klaren Hierarchie führt zum Austausch von Argumenten, die im Normalfall zu einer Kompromisslösung führen, die die Konfliktparteien befrieden. Einen Überblick über die verschiedenen Phasen und Modelle der Problemlösung bei der gemeinschaftlichen Entwicklungsarbeit lieferten Eseryel et al. (2020). In Abhängigkeit von der Komplexität und der Bedeutung des Problems, bedienen sich FLOSS-Entwickler:innen unterschiedlicher Strategien, die auch nicht-konsensuale und intransparente Lösungsstrategien beinhalten können (vgl. ebd., S. 496ff). Interessant ist dabei die Frage, ob alle Entwickler:innen den gleichen Einfluss auf die Entscheidungen in FLOSS-Projekten haben?

Obwohl die offene Struktur einer FLOSS-Gemeinschaft in der Theorie hierarchiearm angelegt ist, bauen Entwickler:innen durch kontinuierliche, gehaltvolle Entwicklungsbeiträge eine Reputation auf, die bei Konflikten von Vorteil ist. Der Einfluss von Reputation führt dazu, dass sie zu relevanten Anderen werden, mit denen ein Kompromiss argumentativ ausgehandelt werden muss, während andere Konfliktparteien leichter ignoriert werden können. Führt weder die Argumentation noch ein Kompromiss zur Konfliktlösung, bleibt eine weitere Lösungsstrategie: Eine Konfliktpartei implementiert ihre favorisierte Lösung und hofft auf Akzeptanz. Dies geht auf die Erfahrung zurück, dass Stagnation im FLOSS-Projekt und die Ermüdung durch den Diskussionsprozess für alle Beteiligten unerwünscht sind und deswegen auch eine nicht gewünschte Entscheidung im Zweifelsfall hingenommen wird (vgl. Taubert 2006, S. 163f).

4.3 Studien zu den Entwickler:innen

Bezüglich der einzelnen Entwickler:innen, die sich in FLOSS-Gemeinschaften engagieren, sieht die Forschungslage viel besser aus, insbesondere im Vergleich zum Kenntnisstand über die kollektiven Aushandlungsprozessen. Durch zahlreiche Befragungen in vielen unterschiedlichen FLOSS-Gemeinschaften konnten Informationen zum sozialen Hintergrund der einzelnen Entwickler:innen ermittelt werden.

Die Studien von Hertel et al. (2003)[42], Lakhani und Wolf (2003)[43] sowie von Hars und Ou (2001)[44] liefern statistische Daten zu den Entwickler:innen, die sich in FLOSS-Projekten engagieren. Mehrere umfangreiche Studien stammen von Gosh et al. (2002, 2007)[45]. Sie erhoben Daten zu einer Vielzahl von FLOSS-Projekten. Diese heterogene

[42] Hertel, Niedner und Herrmann führten eine internetgestütze Befragung in der Linux-Kernel-Community durch. 141 Entwickler:innen nahmen daran teil und berichteten über ihre Aktivitäten und Beweggründe für ihr Engagement am Linux-Kernel und seinen Untersystemen.

[43] Lakhani und Wolf initiierten eine der größten Befragungen mit 684 FLOSS-Entwickler:innen von 287 verschiedenen FLOSS-Projekten. Auch sie schickten E-Mails an die Entwickler:innen über die jeweiligen Projekt-Webseiten und riefen sie zur Teilnahme an einer Online-Befragung auf (vgl. Lakhani & Wolf 2003, S. 8).

[44] Hars und Ou kontaktierten 389 Personen aus verschiedenen FLOSS-Projekten, von denen 79 Personen valide Aussagen in einer Online-Befragung machten (vgl. Hars & Ou 2001, S. 5).

[45] Der Aufruf zur Teilnahme an der mehrsprachigen Umfrage wurden in vielen Foren von FLOSS-Entwickler:innen veröffentlicht, an denen 2774 Befragte teilnahmen (vgl. Ghosh 2007, S. 30).

soziale Bewegung zeigt bei den soziodemographischen Merkmalen der FLOSS-Entwickler:innen folgendes Spektrum:

- 60% der Befragten sind zwischen 16 und 25 Jahre alt (vgl. Ghosh 2007, S. 31); bei Hertel et al. beträgt das mittlere Alter 30 Jahre, mit einer Spanne von 16 bis 54 Jahren (vgl. Hertel et al. 2003, S. 1167f oder auch Lakhani und Wolf (vgl. 2003, S. 9) beziehungsweise 20 - 40 Jahre (vgl. Hars & Ou 2001, S. 5)
- nur 1,2% der Teilnehmenden sind weiblich (vgl. Ghosh 2007, S. 30); 4,3% (vgl. Hertel et al. 2003, S. 1167f); 5% (vgl. Hars & Ou 2001, S. 5) oder 2,5% (vgl. Lakhani & Wolf 2003, S. 9) in den anderen Studien
- knapp 60% leben in einer Partnerschaft, 17% haben mindestens ein Kind (vgl. Ghosh 2007, S. 32)
- circa 70% verfügen über einen College- oder Universitätsabschluss (vgl. Hars & Ou 2001, S. 5), Lakhani und Wolf lieferten detailliertere Ergebnisse und ermittelten bei 51% einen Universitätsabschluss in IT-Studiengängen, während 9% eine formale Programmierausbildung außerhalb der Universität haben und 40% der Teilnehmenden sich das Programmieren selbst informell angeeignet haben (vgl. Lakhani & Wolf 2003, S. 9)
- über die Hälfte der Teilnehmenden arbeiten als professionelle Programmierer:innen (vgl. Hars & Ou 2001, S. 5) beziehungsweise 45% (vgl. Lakhani & Wolf 2003, S. 9), knapp ein Drittel sind Studierende (vgl. Hars & Ou 2001, S. 5) beziehungsweise 20% (vgl. Lakhani & Wolf 2003, S. 9) oder 23% bei der Studie von Hertel et al. (vgl. 2003, S. 1167f)

Zu dem individuellen Engagement der FLOSS-Entwickler:innen lassen sich folgenden Aussagen treffen:

- von denen, die sich als aktive Programmierer:innen oder Maintainer:innen engagierten, werden 20% für ihre Arbeit an dem FLOSS-Projekt direkt bezahlt (vgl. Hertel et al. 2003, S. 1168) beziehungsweise 23% (vgl. Lakhani & Wolf 2003, S. 9) oder 55% können dies während ihrer regulären Arbeitszeit machen (vgl. Lakhani & Wolf 2003, S. 9) beziehungsweise 38% (vgl. Hertel et al. 2003, S. 1168) oder 16% (vgl. Hars & Ou 2001, S. 5)
- 28% der Teilnehmenden betreiben die FLOSS-Entwicklung als reines Hobby (vgl. Hars & Ou 2001, S. 5)
- 50% der Teilnehmenden haben in ein oder zwei Projekte eine herausragenden Position, beispielsweise als Maintainer:in (vgl. Ghosh 2007, S. 36)
- 61% der Teilnehmenden sind in zwei bis zehn Projekten gleichzeitig aktiv (vgl. Hars & Ou 2001, S. 5)

- nur 17% pflegen keine Kontakte zu anderen FLOSS-Entwickler:innen, über die Hälfte aller Teilnehmenden stehen in regelmäßigen Kontakt zu maximal fünf anderen Entwickler:innen (vgl. Ghosh 2007, S. 38)

- der durchschnittliche[46] individuelle Zeitaufwand pro Woche betrug für alle FLOSS-Projekte 14,1 Stunden, mit jeweils 7,5 Stunden am Hauptprojekt (vgl. Lakhani & Wolf 2003, S. 10) beziehungsweise aufgeschlüsselt nach Engagement bei Hertel et al.: durchschnittlich 18,4 Stunden pro Woche für Kernel-Entwickler:innen / Maintainer:innen und 6,6 Stunden für die Gruppe der interessierten Leser:innen der Mailingliste (vgl. Hertel et al. 2003, S. 1168)

Neben der Erhebung dieser soziodemographischen Merkmale wurden die Entwickler:innen auch nach ihren Beweggründen für ihr Engagement in Entwicklungsgemeinschaften befragt. In diesem Zuge wurde auch oft das Lernen durch die Entwicklungsarbeit als ausschlaggebender Faktor benannt. Der Forschungsstand zu beiden Aspekten wird in den kommenden Abschnitten vorgestellt.

4.3.1 Interessen

Die hier präsentierten Studien ähneln sich hinsichtlich ihres Vorgehens und ihrer Frage- und Antwortmuster, sodass sie sich besonders gut für einen Vergleich eignen. Sie folgen einem quantitativen Paradigma. Die Befragungen geben einen ersten Überblick über die Interessen der Befragten, wobei eine Unterscheidung nach extrinsischen und intrinsischen Faktoren vorgenommen wurde.

In allen genannten Studien kristallisierten sich ähnliche Beweggründe heraus, mit denen das Engagement in einem FLOSS-Projekte durch die Entwickler:innen begründet wurde (vgl. Ghosh 2007, S. 34; Hertel et al. 2003, S. 1174; Hars & Ou 2001, S. 5f; Lakhani & Wolf 2003, S. 12):

- Erwerb und Ausbau von Fähigkeiten beziehungsweise diese mit anderen teilen

- sozialpolitische Überzeugung und Identifikation mit der Open-Source oder der Hacker-Community: der Community wieder etwas zurückgeben wollen, Identifikation mit Werten und Handlungsmustern der Community (gegenseitige Unterstützung, Lösung komplexer Probleme, das Teilen von Code, Überzeugung, dass Software ein freies Gut sein sollte)

- Verbesserung oder Implementation von Software, die für berufliche und auch nichtberufliche Zwecke gebraucht wird

[46] Hier zeigten sich große Unterschiede zwischen den Teilnehmenden. Entwickler:innen, die für ihre Arbeit bezahlt wurden, verbringen deutlich mehr Zeit mit der FLOSS-Entwicklung (vgl. Hertel et al. 2003, S. 1168). Dies beläuft sich auf 51% mehr Zeit als bei den Ehrenamtlichen (vgl. Lakhani & Wolf 2003, S. 10).

- Vorteile für die eigene Karriere, Präsentation der eigenen Kompetenzen, Selbstvermarktung
- Freude an der Entwicklungsarbeit

Hars und Ou teilten die Teilnehmenden in drei Gruppen ein und fanden unterschiedliche Interessenlagen vor. Berufliche FLOSS-Programmierer:innen sind hauptsächlich auf Selbstpräsentation bedacht, um ihre Kompetenzen potentieller Kundschaft oder Arbeitgeber:innen zu präsentieren. Auch soll die FLOSS für die eigenen Zwecke genutzt werden. Auch Berufsprogrammierer:innen, die nicht für FLOSS-Projekte bezahlt werden, benötigen die FLOSS für eigene Projekte. Hauptsächlich weil damit der Verkauf von dazugehörigen Produkten oder Dienstleistungen verfolgt wird. Hobbyprogrammierer:innen und Studierende sind stark am Erlernen neuer Fähigkeiten und Fertigkeiten interessiert und zeigen eine hohe Identifikation mit der Community und altruistischen Einstellungen (vgl. Hars & Ou 2001, S. 5f). Gosh et al. unterscheiden dagegen vier Gruppen mit spezifischen Interessenlagen: (1) Die größte Gruppe (53%) ist die Gruppe, die nur soziale Ziele verfolgt (Wissen und Fähigkeiten lernen und weitergeben, an einer neuen Form der Kooperation zu partizipieren, Teil der FLOSS-Community sein). (2) 31% erhoffen sich dadurch noch Vorteile für die Karriere oder verfolgen monetäre Ziele. (3) In Abgrenzung dazu gibt es auch die Gruppe (12,7%), die neben sozialen auch politische Ziele verfolgt (Software darf kein Privateigentum sein, Beschränkung der Macht von Softwarekonzernen. (4) Nur 2,6% der Entwickler:innen wurden allein aus dem Interesse an einer bestimmten Software Teil einer FLOSS-Community (vgl. Ghosh 2007, S. 34f).

In einer qualitativen Studie analysierten Tepe und Hepp das Interesse von Entwickler:innen bei ihrem Engagement in FLOSS-Projekten. Sie stützen sich dabei auf 14 qualitative Interviews einer studentischen Arbeitsgruppe mit internationalen Akteur:innen der FLOSS-Entwicklung, die im Rahmen eines Forschungsseminars 2003 durchgeführt wurde (vgl. Tepe & Hepp 2008b, S. 28). Auch sie bestätigen im Wesentlichen die schon dargestellten Ergebnisse der quantitativen Studien. Sie zeigten, dass viele Entwickler:innen Freude am Lösen von komplexen Problemen haben und dass Spaß am Programmieren ein wichtiger Faktor ist. Dies geht einher mit Lernerfolgen durch den Erwerb neuer Kompetenzen. Als neuen Fakt beschreiben die Autoren die gegenseitige Anerkennung und den Austausch in der Community, der für viele Entwickler:innen wichtig ist (vgl. ebd., S. 32f). Deren Veröffentlichung verbleibt aber bei der reinen Nennung von exemplarisch ausgewählten Zitaten und versucht diese mit Hilfe anderer Autor:innen in einen Kontext einzubetten.

4.3.2 Kompetenzen

Die Vermutung liegt auf der Hand, dass in erster Linie Kompetenzen erworben und ausgebaut werden, die sich auf das technische Wissen rund um die FLOSS-Entwicklung beziehen sowie für die dafür benötigte Werkzeuge und IT-Systeme. Doch der Aufbau von

Kompetenzen ist umfassender. Studien mit dieser Forschungsfokus lieferten einen ersten Einblick zu diesem Thema. Eine Übersicht über den Aufbau von Kompetenzen bei der gemeinschaftlichen FLOSS-Entwicklungen lieferten Glott et al. (2007) und Barcomb et al. (2015). Glott et al. zeigten ein differenziertes Bild an Lernfeldern, bei denen die befragten FLOSS-Entwickler:innen von Lernfortschritten berichteten (vgl. Glott et al. 2007, S. 29).

1. Sachkompetenz beim Programmieren

   Diese erstreckt sich auf die Grundlagen der Programmierung und der Dokumentation von Quellcode. Versierte Entwickler:innen gaben an, dass sich ihr Programmstil hin zu einer besseren Lesbarkeit verbessert hat. So können andere diesen Programmcode leichter wiederbenutzen. Der Umgang mit dem Programmcode von anderen ist wiederum eine weitere Fähigkeit, welche die Fehlersuche und Korrektur in fremdem Programmcode erst ermöglicht. Darüber hinaus fiel es den Entwickler:innen mit zunehmender Erfahrung leichter, den Code modularer zu schreiben. Davon profitiert die Wiederbenutzbarkeit beziehungsweise Austauschbarkeit von einzelnen Codesegmenten und auch das arbeitsteilige Vorgehen zwischen verschiedenen Entwickler:innen wird damit erleichtert. Als weitere Sachkomptenz wurden der Umgang mit unterschiedlichen Programmiersprachen genannt und die Administration und Konfiguration von komplexen Softwaresystemen.

2. Konfliktbewältigung in der Gruppe

   Das Lernfeld in Bezug auf die Gemeinschaftsprozesse beinhaltet zum einen die Koordination der eigenen Arbeit mit der Arbeit anderer und zum anderen die Koordination der Gruppe. Zu diesem Zweck müssen vorher gemeinschaftlich Ziele definiert werden und zusammen verfolgt werden. Die Zielerreichung muss in irgendeiner Art und Weise koordiniert und evaluiert werden.

   Unter den sozialen Fähigkeiten verstehen Glott et al. einen eigenen Standpunkt einzunehmen und ihn gegenüber anderen formulieren zu können. Dazu gehört auch, durch nachvollziehbares Argumentieren andere zu überzeugen beziehungsweise für bestimmte Dinge begeistern zu können. Da dies in Gruppen oft mit Konflikten verbunden ist, zählt der Umgang mit Kritik ebenso als wichtige soziale Fähigkeit. Generell ist eine funktionierende Konfliktbearbeitung eine essentielle Voraussetzung für den Fortbestand der Gruppe.

3. Allgemeine Fähigkeiten

   Darunter wird von den Autor:innen einerseits das Sprachverständnis zusammengefasst, insbesondere die Verwendung von Englisch in technischen Diskussionen. Neben der sprachlichen Kommunikation innerhalb der Entwicklungsgemeinschaft gaben auch viele Entwickler:innen an, eine Sensibilität im gegenseitigen Austausch

im interkulturellen Kontexten entwickelt zu haben. Weiterhin erhielten die Entwickler:innen eine Überblick über die Arbeitsanforderungen im Berufsfeld der Softwareentwicklung.

4. Rechtsverständnis

   Das letzte Lernfeld betrifft die rechtlichen Rahmenbedingungen von FLOSS-Projekten. Dies sind Kenntnisse der Unterschiede zwischen Urheber- und Patentrecht, Haftungsfragen bei Software und ein vertieftes Verständnis von unterschiedlichen Softwarelizenzen.

Die Studie zeigte, dass besonders junge FLOSS-Entwickler:innen in der FLOSS-Community eine informelle Lernumgebung vorfinden, die den Aufbau von Teilkompetenzen in allen genannten Bereichen ermöglicht. Die Einteilung nach Alterskohorten und Vorerfahrungen zeigte, dass es deutliche Unterschiede in den Lernkurven der Untergruppen gibt. Unerfahrene Personen mittleren Alters (27 - 32 Jahre) erwerben grundlegende Programmierfähigkeiten durch die Suche und Ausbesserung von Softwarefehlern. Gleichzeitig versuchen sie andere Entwickler:innen zu motivieren. Nach ein paar Jahren Erfahrung verschiebt sich dies in Richtung der Bereiche Management, Rechtswissen und andere soziale Kompetenzen. Bei der älteren Kohorte (älter als 32 Jahre) profitieren vor allem die erfahrenen Entwickler:innen (mehr als sieben Jahre FLOSS-Entwicklung) durch eine Verbesserung der Fähigkeiten und Fertigkeiten beim Management von FLOSS-Projekten (vgl. Glott et al. 2007, S. 28).

Die quantitative Studie der Arbeitsgruppe um Ann Barcomb hat sich der Frage gewidmet, auf welche Art und Weise FLOSS-Entwickler:innen ihre Kompetenzen erwerben. Im Vergleich von FLOSS-Entwickler:innen mit beruflichen Softwareentwickler:innen zeigte sich, dass dabei Strategien des informellen und non-formalen Lernens zum Kompetenzaufbau eingesetzt werden (vgl. Barcomb et al. 2015, S. 29f)[47]:

- Informelle Strategien

  Diese orientieren sich an den Aktivitäten der anderen Entwickler:innen und anhand der gewonnenen Erfahrungen durch die Partizipation in der Gemeinschaft. Der Aufbau von Kompetenzen wird durch die Evaluation der Arbeit anderer und der Wirkung einer respektvolle Kommunikation in der Projektgemeinschaft realisiert. Neue Entwickler:innen beginnen mit dem Code von anderen zu arbeiten und lernen dadurch selbst, wie wichtig ein guter Programmierstil ist. Auch die Übernahme von Verantwortung, zum Beispiel durch die alleinige Arbeit an einem Softwaremodul, wurde von vielen als wichtige Lernstrategie angesehen.

- Non-formale Strategien

[47] Die Autor:innen der Studie kontaktierte Nutzer:innen eines FLOSS-Produktes und baten sie um deren Teilnahme an einem Erhebung mittels Online-Fragebogen. 5309 Personen nahmen an der Studie teil (vgl. Barcomb et al. 2015, S. 23).

Lernen durch die Teilnahme an Workshops oder Kursen wurde nur wenig genutzt und führte nur in dem Bereich der Grundlagen der Programmierungstechnik zu Verbesserungen. Im Gegensatz dazu wurde häufiger die Nutzung von Fachbücher oder Online-Tutorials thematisiert.

Ansonsten hatte der Katalog der erlernten Fähigkeiten und Fertigkeiten in dieser Studie eine ähnliche Struktur wie in der Studie von Glott et al. 2007. Die aufgeführten Studien geben einen ersten Überblick über die erworbenen Kompetenzen in FLOSS-Projekten. Im Gegensatz dazu bietet die Studie von Bolstad (2006) einen tieferen Einblick.[48] Diese autoethnographische Studie von Bolstad (2006) zielte auf seinen eigenen Kompetenzaufbau: Er partizipierte zwölf Monate aktiv als FLOSS-Entwickler an einem größeren FLOSS-Projekt.[49] In der Arbeit dokumentiert er seinen eigenen Lernprozess. Er beschreibt seine Erfahrungen mit der FLOSS selbst, als Anwender für eigene Zwecke und aus der Sicht eines FLOSS-Entwicklers, der als Teil der Projektgemeinschaft diese Software zusammen mit anderen verbessert. Bei der Beschreibung seines eigenen Lerncurriculums setzt er einen Fokus auf die technischen Fähigkeiten und Fertigkeiten im Umgang mit der FLOSS selbst, den Entwicklungswerkzeugen und der Administration der Serverumgebung, die für den Betrieb der FLOSS notwendig ist. Er listet seine Aufgabengebiete auf und dokumentiert das Kommunikationsnetzwerk und die koordinierenden Handlungen innerhalb der Entwicklungsgemeinschaft. Dies zeigt, dass es neben Sachkompetenz auch Sozialkompetenz bedarf. Für Bolstad ist nicht nur die Entwicklungsgemeinschaft selbst eine Lernumgebung, sondern auch die Anwendung der FLOSS für kommerzielle Nutzer:innen, in deren Auftrag er die FLOSS eingerichtet hat. Dies betrifft beispielsweise auch die Schulung seiner Kundschaft durch ihn selbst (vgl. ebd., S. 51 - 90).

4.4 Zusammenfassung

FLOSS-Projekte unterscheiden sich von proprietären Projekten durch den Modus der permanenten Veröffentlichung aller Softwareversionen samt des zugehörigen Quellcodes. Während proprietäre Software einem linearen Produktlebenszyklus mit festen Verantwortlichkeiten und Entwicklungsschritten unterliegt, gibt es beim Lebenszyklus von FLOSS die Möglichkeit von Schleifen. Ein Ende der Entwicklungsarbeit bei FLOSS bedeutet nicht automatisch das Ende des Projektes. Durch die Verfügbarkeit des Quellcodes können neue Entwickler:innen das Projekt jederzeit wieder reaktivieren und eine neue Entwicklungsgemeinschaft aufbauen.

Das Schalenmodell zeigt die differenzierten Rollen und Aufgabenbereiche die sich in FLOSS-Gemeinschaften herausbilden können. Die Einflussmöglichkeiten auf Entscheidungen über die Entwicklungsrichtung nehmen in Richtung des Zentrums des Modells

[48] Die Masterarbeit von Bolstad findet deshalb hier Erwähnung, weil sie einen differenzierten Einblick in die unterschiedlichen Arbeitsbereiche von Entwickler:innen ermöglicht. Sie erreicht eine Tiefe, wie sie für die üblichen Forschungsarbeiten auf diesen Gebiet ungewöhnlich ist.

[49] Bei dem FLOSS-Projekt handelt es sich um Plone. Es ist einem Content-Management-System, mit dem sich Webseiten mit verschiedenen Funktionalitäten erstellen lassen.

zu. Die Schichten sind prinzipiell durchlässig und daran geknüpft, wie stark das individuelle Engagement ist beziehungsweise wie bedeutsam die Beiträge für die Entwicklungsgemeinschaft sind. Dabei ist zu beachten, dass nicht nur Beiträge relevant sind, die Programmierkenntnisse erfordern. Eine Vielzahl von wichtigen Aufgaben wie zum Beispiel, Softwaretest, Webseitengestaltung, Übersetzung, Unterstützung von Nutzer:innen oder Koordination und Pflege der Entwicklungsgemeinschaft kann auch von Nicht-Programmierer:innen übernommen werden.

Um das arbeitsteilige Vorgehen von regional verteilten Entwicklungsgemeinschaft zu unterstützen, werden verschiedene Werkzeuge eingesetzt. Dabei wird mindestens eine Versionsverwaltung für die Verwaltung der gemeinsamen Codebasis sowie der Dokumentation des FLOSS-Projektes und verschiedene synchrone und asynchrone Kommunikationskanäle verwendet. Diese beiden Grundfunktionen werden in diversen Softwareentwicklungsportalen in der Regel kostenfrei für neue FLOSS-Projekte angeboten. Dies ermöglicht es den Projektinitiator:innen in der Anfangsphase den Fokus auf die Arbeit an der Codebasis zu legen und neue Entwickler:innen für das Projekt zu begeistern. Je nach dem Wachstum der Entwicklungsgemeinschaft und sich ändernden Anforderungen verändert sich auch die Art und Anzahl der genutzten Werkzeuge.

Bezüglich der Kommunikationskanäle wurden in erster Linie asynchrone Kommunikationsmittel erforscht. Die Nutzung von synchronen Kanälen wie zum Beispiel die Chat-Kommunikation in FLOSS-Gemeinschaften wurde bislang kaum wissenschaftlich berücksichtigt. Dabei wurde gezeigt, dass viele FLOSS-Projekte keinen formalen Hierarchien unterliegen, aber auch nicht als hierarchiefrei bezeichnet werden können. Konflikte werden meist argumentativ geklärt. Durch eine informelle Hierarchie die sich auf Reputation der Entwickler:innen stützt, wird bestimmt, welche Personen bei einer Kompromissfindung berücksichtigt werden müssen.

Der Forschungsstand über die FLOSS-Entwickler:innen speist sich hauptsächlich aus quantitativen Befragungen. Diese zeigen, dass die meisten Personen zwischen 20 und 40 Jahre alt sind, einen höhere Bildungsabschluss besitzen und hauptsächlich männlich sind. Ihre Beweggründe für das Engagement sind sehr vielschichtig und reichen von individuellen Notwendigkeiten über soziale und politische Ziele bis hin zu beruflichen Gründen. Wie diese Ziele bei einer Person gewichtet sind und wie sie miteinander in Verbindung stehen ist bislang nur in wenigen Studien Gegenstand der Forschung. Auch die Art und Weise wie und was gelernt wird, ist nur unzureichend erforscht. Klar ist, dass eine Vielzahl von Fähigkeiten und Fertigkeiten erworben werden und dies von Person zu Person sehr unterschiedlich ausfällt. Auch die Lernstrategien variieren stark zwischen den Personen und bestehen aus informellen und non-formalen Strategien und bauen teilweise auf formal erworbenen Qualifikationen auf. Ebenso wurde bislang die Kommunikation innerhalb einer FLOSS-Gemeinschaft mit Bezug auf das Lernen erst ansatzweise beforscht. Diesbezügliche Studien lieferten erste Einsichten, zum Beispiel durch die Beschränkung

auf die Kommunikationsmetadaten.[50] Eine der wenigen Ausnahmen bildet hier die Verknüpfung von Beitragsanalyse und eingereichte Codebeiträge der Entwickler:in von Kuk (2006).

In diese Lücke greift die vorliegende Forschungsarbeit. Durch eine qualitative Befragung von 14 Entwickler:innen und einer Beitragsanalyse der E-Mail-Kommunikation samt quantitativer Betrachtung der Beiträge in der Versionsverwaltung eines FLOSS-Projektes, die über mehrere Jahre hinweg die Entwicklung der FLOSS-Gemeinschaft dokumentiert, wird versucht, die ausgemachten Fehlstellen in der Forschung weiter zu schließen. Damit soll der Erkenntnishorizont mit Bezug auf Interessen, Lerninhalte und den angewendeten Lernstrategien erweitert werden. Wie schon bei den Rahmenbedingungen für das gemeinschaftliche Lernen ausgeführt wurde, wird dabei der Kommunikation zwischen den Entwickler:innen beim Aufbau von Kompetenzen im Forschungsdesign die notwendige Beachtung geschenkt.

[50] Dies sind Empfänger:in bzw. Sender:in der Nachricht, Betreff und Zeitpunkt der Nachricht.

5 Methodische Vorgehensweise

Im Folgendem werden die wissenschaftstheoretischen und die methodischen Grundlagen für diese Arbeit vorgestellt. Der Schwerpunkt liegt bei der qualitativen Analyse von Lernprozessen von Entwickler:innen der FLOSS-Bewegung. Durch unterschiedliche Zugänge zum Forschungsfeld, wurde die individuelle Praxis und die Erfahrungen bei der gemeinschaftlichen FLOSS-Entwicklung erfasst. Um dies leisten zu können, wurden FLOSS-Entwickler:innen mittels Expert:inneninterview befragt und die Kommunikation zwischen einzelnen Entwickler:innen innerhalb eines FLOSS-Projektes ausgewertet.

5.1 Methodologie

Bevor die Forschungsmethoden beschrieben werden, soll der erkenntnistheoretische Hintergrund verdeutlicht werden. Der Forschungsansatz dieser Arbeit ist im *interpretativen Paradigma* zu verorten. Dies bedeutet, dass „die soziale Wirklichkeit [...] als durch Interpretationshandlungen konstituierte Realität" (Lamnek 2010, S. 32) begriffen werden kann. „Gesellschaftliche Zusammenhänge, die einer soziologischen Analyse unterworfen werden können, sind daher nicht objektiv vorgegebene und deduktiv erklärbare soziale Tatbestände, sondern Resultat eines interpretationsgeleiteten Interaktionsprozesses zwischen Gesellschaftsmitgliedern" (ebd., S. 32). Die sich darauf berufenden qualitativen Forschungsverfahren gehen zurück auf den *Symbolischen Interaktionismus*, der *phänomenologischen Soziologie* und der *Wissenssoziologie* (vgl. Kreitz et al. 2016, S. 8). Ihnen gemeinsam ist, dass Personen aufgrund von Sinnkonstruktionen und Deutungsmustern handeln, die individuell verschieden sind.

> „Wirklichkeit wird als eine zu interpretierende verstanden, und zwar nicht nur in der Weise, dass sie in hohem Maße interpretationsbedürftig ist, sondern sie konstituiert sich erst in den Interpretationen der Akteure. Qualitative Forschung weist eine Priorität deskriptiver Verfahren gegenüber explanativen auf. Daraus folgt nicht, dass man auf Erklärungen verzichtet, sondern daraus folgt das Bemühen, das jeweilige Phänomen, um das es geht, so genau wie möglich in seinem Vollzugscharakter zu beschreiben." (Marotzki 2006, S. 112)

Aus diesem Grund ist es für eine theorieentdeckende Forschung notwendig, mit einem offenen Ansatz diese Sinnkonstruktionen und ihre Bedeutung für das individuelle Lernen zu verstehen. Daraus ergeben sich Konsequenzen für die genutzten Forschungsmethoden.

> „Mit der sozialen Konstitution des Gegenstandes ist ein grundlegend anderes Verhältnis der Forschenden zu ihren Informanten verbunden als in der quantitativen Forschung: Die traditionelle Abfrage der Laien durch einen Experten wird in der qualitativen Forschung durch eine Beziehung ersetzt, in der den Informanten die Rolle als 'Experten ihrer Lebenswelt' zugeschrieben wird,

> die über ihren Lebensbereich mehr wissen als die Forschenden und deren Sinnzuschreibungen eben deshalb gefolgt werden muß.“ (Kade 1999, S. 342)

Um die individuellen Sinnbezüge und die Folgen für das Lernen zu rekonstruieren, wurden Selbstzeugnisse von FLOSS-Entwickler:innen ausgewertet. Dies erfolgte anhand von Beitragsanalysen und teilstandardisierten Interviews.

5.2 Beitragsanalyse von Mailinglisten

Auskünfte über die Rahmenbedingungen und Abläufe der individuellen Lernsituationen können nur die Lernenden selbst geben. Aus diesem Grund war der erste Ansatz für theoretische Vorannahmen Selbstzeugnisse der Entwickler:innen auszuwerten. Diese bestanden aus den öffentlich zugänglichen Archiven der Mailinglisten, die für die Kommunikation im Projekt genutzt wurden. In einer ersten Sondierung wurden Projekte ausgewählt, die von der Größe der Entwicklungsgemeinschaft und dem Entwicklungsfortschritt den Eindruck einer funktionierenden Praxisgemeinschaft erweckt haben. Das entscheidende Kriterium dafür war die Veröffentlichung einer ersten stabil laufende Software (Version 1.0) an der mehrere Projektmitglieder mitgewirkt haben.

Vorteile dieser Herangehensweise liegen in der nicht-reaktiven Datenerhebung dieser hermeneutischen Methode. Dies entspräche dem klassischen Ursprung der Methode in der Medienanalyse. Zu Beginn des 20. Jahrhunderts beschrieb Max Weber die Umrisse der quantitativen Inhaltsanalyse als sozialwissenschaftliche Forschungsmethode. Er entwickelte dieses Verfahren bei der Analyse von Zeitungsinhalten (vgl. Kuckartz 2018, S. 14). Verzerrungseffekte zwischen Forschenden und Beforschten bei der Datenerhebung können dadurch verhindert werden, da die Betroffenen sich zum Zeitpunkt des Selbstzeugnisses keiner direkten Erhebungssituation bewusst sind.[51]

Erste Annäherung an die Zielgruppe

Mit dem Ansatz der Beitragsanalyse wurden circa 440 E-Mails eines Projektes, bestehend aus zwei Mailinglisten (Entwickler:innen und Übersetzer:innen) in den Datensatz aufgenommen und qualitativ ausgewertet. Durch das Verfahren des *offenen Kodierens* wurde eine erste Sichtung des Materials in Bezug auf Kommunikationsinhalte vorgenommen. Themen der Kommunikation waren unter anderem:

- Aufgabenverteilung
- Erinnerung an Fristen
- Zusammenfassung des Projektfortschritts

[51] Da alle E-Mails, die über die zentrale Mailingliste des Projekts verschickt werden öffentlich zugänglich sind, ist anzunehmen dass es dennoch eine Verzerrung im Rahmen der sozialen Erwünschtheit existiert. Die Mitglieder der Mailingliste sind sich in der Regel über deren öffentlichen Natur und Speicherung bewusst.

- Information über geleistete Projektbeiträge

Hintergründe über individuelle Lernsituationen konnten nur äußerst sporadisch gefunden werden. Das gesammelte Datenmaterial war dadurch nicht geeignet, umfassende Erkenntnisse zum Kompetenzaufbau zu liefern. Jedoch lassen sich anhand der Beitragsanalyse die Interaktionen zwischen den Entwickler:innen und deren Rollen im Projekt nachvollziehen. Sie liefert wichtige Informationen über die Praxis des Lernens in seinem sozialen Kontext. Sie ist geeignet, um Rahmenbedingungen des Lernens zu erfassen. Deren Betrachtung erfolgt in Kapitel 6.5. Was die Beitragsanalyse hingegen nicht leisten kann, ist eine Beschreibung der Ergebnisse von Lernprozessen. Hierfür bedarf es einer anderen Forschungsmethode.

Als Reaktion darauf wurde versucht, die aktivsten der 33 Autor:innen der kodierten E-Mails für ein Interview zu gewinnen. Die Zielvorstellung war, den Handlungskontext zu erfassen, aus dem die analysierten E-Mails resultierten. Aus diversen Gründen wurden jedoch die Interviewanfragen abgelehnt oder es wurde nur mit kurzen schriftlichen Stellungnahmen reagiert.[52] Aus diesem Grund wurde eine alternative Handlungsstrategie gesucht. Die Interviewanfrage eines für die Entwicklungsgemeinschaft zuvor unbekannten Forschers per E-Mail war in diesem Fall kein geeigneter Ansatz, um Informationen über den Kontext individueller Lernsituationen bei der FLOSS-Entwicklung zu erfassen.

5.3 Leitfadeninterviews mit FLOSS-Expert:innen

Um den individuellen Lernprozess rekonstruieren zu können, wurde das leitfadengestützte Expert:inneninterview als Erhebungsmethode gewählt. Nur die Expert:innen selbst können Aussagen über ihren eigenen Lernprozess treffen. Als nicht-standardisierte Befragung eignet sie sich im besonderem Maße im Rahmen der theorieentdeckenden Forschung.

Ziel war es, dass die Befragten anhand von Leitfragen beginnen, ihren eigenen Lernprozess zu retrospektiv zu beschreiben. Die Leitfragen fungieren als Bindeglied zwischen den theoretischen Vorannahmen und der Erhebungsmethode selbst (vgl. Gläser & Laudel 2009, S. 90). Durch den offenen Charakter obliegt es ihnen selbst, Schwerpunkte zu setzen und dadurch Einflussfaktoren für das eigene Lernen offenzulegen, die zuvor nicht von den Leitfragen abgedeckt waren.

Des Weiteren wurde die Methode des Expert:inneninterviews gewählt, da sie es erlaubt, mehrere Themen abzudecken, die durch das Ziel der Untersuchung bestimmt werden. Diese werden durch die Leitfragen vorgegeben und ermöglichen es, genaue Informationen zu zentralen Aspekten des Lernens zu erfassen (vgl. ebd., S. 111).

[52] Gründe für eine Ablehnung der Interviewanfrage waren beispielsweise das teilweise lange zurückliegende Engagement bei diesem Projekt, selbst keine zentrale Rolle in der Entwicklung gehabt zu haben und andere Leute kompetentere Ansprechpartner:innen wären, Zeitmangel oder generelles Desinteresse an Forschung mitzuwirken.

5.3.1 Konstruktion des Leitfadens

Die Themen der Fragenblöcke leiten sich einerseits aus theoretischen Vorannahmen über den Lernprozess und andererseits aus den grundlegenden Forschungsfragen ab. Demnach soll die Befragung eine Datenbasis ermöglichen, deren Auswertung Antworten über Sinn- und Relevanzstrukturen des Lernens liefert, den Lernprozess rekonstruieren kann und einen Überblick darüber liefert, was im Rahmen der FLOSS-Entwicklung gelernt wird. Die theoriegeleitete Differenzierung der Fragestellung resultierte in den folgenden Strukturierungsdimensionen des Leitfadens:

- Interesse
- Handlungsfelder
- Entwicklungsprozess
- Bildung
- Kompetenzen
- Wissen

Die Fragen sind so angeordnet, dass sie stufenweise von konkreten Handlungen und Erfahrungen hin zu generalisierenden Selbsteinschätzungen überleiten. Der Frageblock *Interesse* geht der Frage nach, warum die befragte Person damals begann, sich für FLOSS zu engagieren. Es wird danach gefragt, ob sich das Interesse im Laufe der Zeit geändert hat und welche Ziele mit dem Engagement verfolgt wurden. Im Anschluss daran wird im Block *Handlungsfelder* gebeten, die konkreten Aufgabenfelder und Handlungen zu beschreiben, denen die Entwickler:in in den jeweiligen Projekten nachgegangen sind. Es soll dargestellt werden, warum die befragte Person sich dafür entschieden hat, ob es im Verlauf der Jahre zu Änderungen kam und was die Ursachen dafür waren.

Der wichtigste Frageblock gilt dem individuellen *Entwicklungsprozess*. Dieser beleuchtet die Routinen bei der Entwicklung von FLOSS. Die Entwickler:innen werden aufgefordert, anhand von konkreten Problemen ihr Vorgehen zu beschreiben. Welche Ressourcen zur Problemlösung werden dabei eingesetzt und was sind hemmende sowie förderliche Faktoren für den Entwicklungsprozess? Ein Schwerpunkt bei den Fragen spielt die Vernetzung mit anderen Entwickler:innen. Welche Bedeutung haben soziale Prozesse bei der Entwicklungsarbeit? Schlüsselereignisse können Anstöße zu Veränderungen im Denken und Handeln geben. Da diese im Vorfeld nicht bekannt sind, zielt diese offene Frage auf die Entdeckung von zuvor unbekannten Einflussfaktoren auf den Lernprozess ab.

Nachdem die befragte Person sich in einer Retrospektive wieder die eigenen Erfahrungen im Verlauf ihres Engagements in der FLOSS-Entwicklung in Erinnerung gerufen hat, folgt die Selbsteinschätzung über Veränderungen im eigenen Denken und Handeln. Die Frage im Bereich *Bildung* versucht die befragte Person zu ermutigen, zuvor implizite Veränderungsprozesse in der Selbstwahrnehmung und ihrer Sicht auf die Gesellschaft

sich bewusst zu machen. Bei den abschließenden Fragen zu *Kompetenzen* und *Wissen* sollen die Entwickler:innen sich selbst in Bezug auf neue Kompetenzen und den Aufbau von Fachwissen einschätzen.

Im Zuge der ersten Interviews zeigte sich, dass der Frageblock *Entwicklungsprozess* den interviewten Personen teilweise zu wenig Anlass gab, die Lösungsstrategien für Probleme zu beschreiben. Dieser Unschärfe wurde durch die Aufnahme weiterer Fragen in dem Block Rechnung getragen. Dabei orientierte sich der Forschungsprozess an dem Modell der Zirkularität, wie ihn Uwe Flick vorschlägt. Im Gegensatz zum linearen Prozessmodell, beinhaltet diese Vorgehensweise eine permanenten Reflexion des Forschungsvorgehens. Durch die zeitliche Nähe von Datenerhebung und Auswertung von empirischen Material lässt sich schon früh prüfen, ob die gewählte Methoden dem Gegenstand der Forschung gerecht wird (vgl. Flick 2016, S. 126f). Der vollständige Leitfaden ist im Anhang aufgeführt.

Feldzugang

Der Feldzugang erfolgte erst durch Interviewanfragen über Mailinglisten verschiedener Projekte. Die Projekte wurden anhand von selbst gewählten Parametern (Teamgröße, Projektlaufzeit, Entwicklungsstand der Software) ausgewählt. Diese Art der Ansprache erwies sich jedoch als nicht geeignet. Die Antwortrate auf diese initiale E-Mail-Ansprache war sehr gering, weshalb dieser Weg nicht weiter verfolgt wurde. Alternativ wurden Wege der direkten Ansprache genutzt.

Um Aussagen über individuelle Sichtweisen auf das eigene Lernen machen zu können, wurden insgesamt 14 Interviews mit FLOSS-Entwickler:innen geführt. Der Feldzugang erfolgte hauptsächlich mittels persönlicher Ansprache auf Konferenzen, über das soziale Netzwerk des Autors und mittels Hinweisen zu potentiellen Interviewpartner:innen durch die interviewten Personen selbst.

Datenkorpus

Alle 14 geführten Interviews fanden Eingang in den Datenkorpus. Ein Interview konnte wegen mangelhafter Aufnahmequalität durch die Telefonverbindung nicht transkribiert werden. Dieses Interview wurde in seinen zentralen Aussagen für die Forschungsfragen stichpunktartig erfasst und in den Datenkorpus aufgenommen. Im Mittel hatte die Interviews eine Dauer von circa 50 Minuten.

Entstehungssituation

Alle Interviews wurden von mir vorbereitet und durchgeführt. Die Teilnahme war freiwillig und erfolgte auf den Konferenzen spontan durch direkte Ansprache. Bei den anderen Interviews fand meist ein kurzer E-Mail-Austausch im Vorfeld statt, bei dem die Modalitäten des Interviews geklärt wurden. Diese Heterogenität spiegelt sich auch in den Orten für die Interviews wieder. Sie erfolgten entweder am Rand von Konferenzen, in Büros, per Telefon, in Privaträumen oder in Cafés .

Formale Charakteristika des Materials

Die Interviews wurden digital aufgezeichnet und anschließend von mir transkribiert. Dabei fand ein vereinfachtes Regelsystem zur Transkription Anwendung:

- es wurde wörtlich, nicht lautsprachlich transkribiert
- Dialektfärbungen wurden in Hochdeutsch überführt
- Wortverschleifungen wurden an das Schriftdeutsch angenähert
- Wort- und Satzabbrüche wurden mit „/" erfasst
- Wiederholungen und Füllworte wurden zugunsten der Lesbarkeit nicht mit transkribiert
- Pausen sind mit „(.)" für eine kurze, „(..)" für eine mittlere und „(...)" für eine lange Pause markiert
- unverständliche Worte und Textpassagen wurden durch „[unverständlich]" ersetzt
- benannte Personen und Projekte wurden pseudonymisiert oder durch eine allgemeine Bezeichnung ersetzt

Richtung der Analyse

In den Interviews beschreiben die befragten Personen ihre Erfahrungen bei der Entwicklung von FLOSS und warum sie sich damals dazu entschlossen, sich in FLOSS-Projekten zu engagieren. Es beinhaltet biographische Elemente, die sich in Veränderungen der individuellen Aktivitäten im Rahmen von FLOSS-Projekten niederschlagen. Das jeweilige Interview soll die befragte Person dazu anregen, über den Kontext von Lernprozessen zu reflektieren. Durch diesen Prozess soll versucht werden, zuvor implizites Wissen über das eigene Lernen zur Beantwortung der Forschungsfrage zu Verbalisieren.

Ablaufmodell der Analyse

Das Material wurde einer strukturierende Inhaltsanalyse unterzogen (vgl. Mayring 2015, S. 97). Für die Genese der Kategorien wurden aus den Strukturierungsdimensionen deduktiv das Kategoriesystem abgeleitet. Diese Kategorien sind theoretisch begründet. Die konkreten Ausprägungen der Kategorien wurden wiederum aus dem Material selbst erstellt. Dabei wurde sich an folgendem Vorgehen orientiert, welches an das Ablaufmodell von Mayring angelehnt ist (vgl. ebd., S. 98):

1. Bestimmung der Analyseeinheiten
2. Festlegung der Strukturierungsdimensionen
3. Ausprägungen der Kategorien bestimmen

4. Kategorien durch Definitionen und Ankerbeispiele abgrenzen
5. Materialdurchlauf und Überarbeitung des Kategoriensystems und der Definitionen
6. finaler Materialdurchlauf
7. Ergebnisaufbereitung

Bei der Kodierung ergaben sich auch Hinweise auf zuvor nicht antizipierte Einflussfaktoren. Diese wurden ebenfalls mit in das Kategoriensystem aufgenommen. Damit enthält die Kategoriebildung deduktive wie auch induktive Elemente, wie es häufig in der Forschungspraxis anzutreffen ist (vgl. Kuckartz 2010, S. 214). Als unterstützenden Maßnahme für die Auswertung, wurden im gesamten Forschungsprozess Memos zu den Interviews, besonderen Textstellen und Analysehinweise für die Auswertung geschrieben. Für die Kodierung wurde die auf qualitative Inhaltsanalyse spezialisierte Software *MaxQDA* eingesetzt.

5.4 Typenbildung

Im Anschluss an die Inhaltsanalyse wird anhand der Interviews eine Typenbildung durchgeführt. Dabei wird vom Verständnis des Einzellfalls abstrahiert und es werden fallübergreifend typische Merkmalsausprägungen herausgearbeitet. Die Bedeutung der Typenbildung resümiert Kuckartz wie folgt:

> „[...] das Ziel der Forschung [ist] das Erkennen von Regelmäßigkeiten, d.h. es wird Verallgemeinerung angestrebt und der einzelne Forschungsteilnehmende interessiert vorrangig als 'Fall von ...' und nicht in seiner einzigartigen Individualität. [...] Typenbildung basiert darauf, Ähnlichkeiten zwischen Individuen (Fällen) zu analysieren und diese Fälle entsprechend zu gruppieren, wodurch der Charakter des Singulären zugunsten des Allgemeinen zurückgedrängt wird.“ (Kuckartz 2016, S. 41)

Die Typenbildung ist damit als ein weiterer Forschungsschritt zu begreifen, um den Fokus von einer Person beziehungsweise ein Merkmal hin zu einer Gruppe von Personen auszuweiten. Diese Gruppe wird als ein Typus konstruiert, der hinsichtlich bestimmter Kategorien eine ähnliche Merkmalsausprägung aufweist. Als zweite wichtige Funktion der Typenbildung führt Kuckartz die Möglichkeit an, typenspezifische Interventionsmaßnahmen vorbereiten zu können. Bei einer anwendungsorientierten Forschung können Maßnahmen für eine Zielgruppe erarbeitet werden, sofern sie sich einem Typus in einer zuvor gebildeten Typologie zuordnen lässt (vgl. ebd., S. 41f).

5.4.1 Verfahren empirischer Typenbildung

Die Geschichte der Typenbildung als wissenschaftliche Methode der empirischen Sozialforschung reicht zurück bis zu den Anfängen des letzten Jahrhunderts. Klassiker wie

Georg Simmel, Max Weber oder Alfred Schütz haben sich in ihren Werken mit der Typenbildung beschäftigt. Einen Überblick über grundlegende Verfahren bietet Kluge (1999). Sie unterscheidet die Typologischen Operationen nach Barton und Lazarsfeld, die Prozeßstrukturanalyse nach Gerhardt und die Typologische Analyse nach Kuckartz. Die Verfahren werden hier in ihren Grundzügen vorgestellt, um im Anschluss eine begründete Auswahl für diese Arbeit zu treffen.

Typologische Operationen nach Barton und Lazarsfeld

Den Begriff des Merkmalsraums prägten Barton und Lazarsfeld. Jeder Typ entspricht einer Merkmalskombination, die einen Merkmalsraum aufspannt. Die Dimensionen des Raumes entsprechen der Anzahl der berücksichtigten Dimensionen. Bei wenigen Dimensionen lassen sich die jeweiligen Merkmalsausprägungen jedes Untersuchungselementes in einem Koordinatensystem oder in Kreuztabellen darstellen. Elemente werden zu einem Typ zusammengefasst, wenn sie eine hohe interne Homogenität aufweisen. Das heißt, dass sie bei vielen Merkmalen eine ähnliche Ausprägung besitzen. Gleichzeitig sollen sich die Elemente eines Typus aufgrund ihrer externen Heterogenität von den Merkmalsausprägungen der Elemente eines anderen Typus unterscheiden (vgl. Kluge 1999, S. 93f).

Um die Typenbildung durchzuführen werden mögliche Kombinationen so lange miteinander verglichen, bis sich aufgrund der Kontrastierung geeignete Typen herausbilden. Da die Zahl der theoretisch möglichen Kombinationen mit weiteren Merkmalen rasant steigt, wird der Merkmalsraum reduziert. Dies kann durch Vereinfachung der Dimensionen[53], funktionale Reduktion[54], numerische Reduktion[55] oder aufgrund forschungspragmatischer Gründe geschehen (vgl. ebd., S. 101ff). Alternativ zur Reduktion kann auch durch deren Umkehrung eine Typologie abgeleitet werden. Dieser Vorgang wird als Substruktion bezeichnet. Dabei wird ausgehend von einem vermuteten Typus die entsprechenden Merkmalsausprägungen abgeleitet. Dieses Vorgehen bietet sich beispielsweise bei großen Datenmengen an, um das Material intuitiv in Untergruppen zu gliedern (vgl. ebd., S. 104). Die Substruktion hat den Nachteil, dass zu einer Typologie mehrere Merkmalsräume ermittelt werden können. Durch das Weglassen von Merkmalen oder durch die Einbeziehung weiterer Merkmale kann dem Problem begegnet werden. Eine solche Operation wird als Transformation bezeichnet (vgl. ebd., S. 104 - 107).

Prozeßstrukturanalyse nach Gerhardt

Anhand biographischer Daten von Patient:innen stellte Gerhardt ein Verfahren vor, dass sich weder in der historisch-biographischen Beschreibung von Einzelfällen verfängt oder sich auf die Ableitung allgemeiner Gesetzmäßigkeiten beschränkt. Sie zielt auf die

[53] Ordinalskalierte Merkmale können beispielsweise auf dichotome Merkmalsausprägung reduziert werden.

[54] Manche Merkmalskombinationen sind so selten, dass sie sich nicht für die Typenbildung eignen.

[55] Einzelne Merkmalsausprägungen werden Zahlenwerte zugewiesen, die wiederum mit den Werten anderer Merkmale addiert werden.

intersubjektive Erklärung von biographischen Verläufen vor dem Hintergrund gesamtgesellschaftlicher Strukturen. Sie orientiert sich dabei an der Bildung von Idealtypen in der Tradition von Weber (vgl. Kluge 1999, S. 116). Ihre Typenbildung erfolgt anhand von vier Schritten: Im ersten Schritt wird jeder Fall einzeln für sich rekonstruiert und anschließend mit den anderen Fällen kontrastiert. Ähnliche Fälle werden in Gruppen zusammengefasst. Je nachdem, welche Merkmale für die Gruppenbildung herangezogen werden, können sich unterschiedliche Gruppen bilden. Im darauffolgenden Schritt wird in jeder Gruppe ein Idealtypus bestimmt. Dieser repräsentiert eine optimale Merkmalsausprägung. In der anschließenden Konfrontierung werden alle Idealtypen mit den anderen Fällen ihrer Gruppe verglichen. Abschließend erfolgt die Struktur- und Prozeßanalyse. Bei diesem Schritt werden alle Idealtypen unter dem Gesichtspunkt soziostruktureller Merkmale zueinander in Beziehung gesetzt. Hierfür werden Mehrfeldertafeln genutzt, auf denen alle Kombinationen von Idealtypen dargestellt werden (vgl. ebd., 127ff).

Typologische Analyse nach Kuckartz

Ziel der Typologischen Analyse nach Kuckartz ist die Bildung von verständlichen Handlungstypen, die durch die Verbindung von hermeneutischen und statistischen Methoden konstruiert werden. Kuckartz zielt auf die Bedeutung des Verstehens sozialer Regelmäßigkeiten ab statt diese nur aufgrund statistischer Korrelationen zu beschreiben (vgl. Kluge 1999, S. 179).

Seine methodisch kontrollierte Typenbildung knüpft an die Idee des Merkmalsraums an. In Anlehnung an Alfred Schütz übernimmt er die Zielvorstellung, dass Typisierung eine grundlegende Basistechnik sozialwissenschaftlicher Analyse ist, „die eben auf das Verstehen des Typischen und nicht psychologisch auf das Verstehen des Einzelnen abzielt“ (Kuckartz 2010, S. 98). Seine Methodik der Typenbildung setzt bei der qualitativen Inhaltsanalyse an.

5.4.2 Auswahl der Methode für die Typenbildung

Die Prozeßstrukturanalyse nach Gerhardt ist stark auf die Analyse biographischer Verläufe ausgerichtet. Im Rahmen der Interviews liegen solche Angaben nur als Sekundärinformationen vor, da die Leitfragen anhand der zentralen Forschungsfragen formuliert wurden. Auch der Abgleich mit soziostrukturellen Bedingungen erscheint nicht sinnvoll, da der Fokus dieser Arbeit stark auf individuelles Handeln und deren Folgen für das eigene Engagement und dem darauf aufbauenden Kompetenzerwerb gerichtet ist.

Die Typologische Analyse und die Typologischen Operationen haben viele Überschneidungsbereiche: Kuckartz Typologische Analyse kann gewissermaßen als Weiterentwicklung der Typologischen Operationen verstanden werden, da sie auf denselben Ursprüngen aufbauen und Kuckartz die Typologischen Operationen für seine Methodik adaptiert hat. Für die Typologische Analyse spricht, dass die Typenbildung auf eine strukturierende Inhaltsanalyse nach Mayring aufbaut.

Die von Kuckartz ausgearbeitete typenbildende Inhaltsanalyse ist methodisch von ihm auf die strukturierende Inhaltsanalyse zugeschnitten. Grund dafür ist die häufige Anwendung dieser Form der Inhaltsanalyse in der qualitativen Forschung (vgl. Kuckartz 2016, S. 45). Dementsprechend ist es sinnvoll, die Typologische Analyse auch für die Typenbildung in dieser Forschungsarbeit einzusetzen. Die typenbildende Inhaltsanalyse besteht aus acht Phasen (vgl. ebd., S. 46 - 49):

1. Bestimmung vom Ziel und Fokus der Typenbildung
2. Festlegen des Merkmalsraums
3. Kodieren und Rekodieren, falls noch nicht fallbezogen kodiert wurde
4. Konstruktion der Typologie
5. Zuordnung der Fälle zu den Typen
6. Beschreibung der einzelnen Typen
7. Zusammenhänge von Typen und sekundären Informationen analysieren (zum Beispiel themenrelevante Einstellungen oder Bewertungen)
8. Fallinterpretation anhand eines repräsentativen Einzelfalls oder eines Modellfalls anhand der Typologie

Anhand dieser Phasen wurde die Typenbildung im Anschluss an die Inhaltsanalyse in dieser Arbeit durchgeführt.

6 Interaktion in einem FLOSS-Projekt

Dieses Kapitel beschreibt die gemeinsame Praxis in einem ausgewählten FLOSS-Projekt. Es wird dargestellt werden, ob diese Entwicklungsgemeinschaft den Kriterien der Theorie der Communiies of Practice gerecht wird. Diese beinhaltet das wechselseitige Engagement, die gemeinschaftliche Unternehmung und das gemeinsame Repertoire. Die Betrachtung der Interaktionen zwischen den einzelnen Mitgliedern der Entwicklungsgemeinschaft soll Rückschlüsse auf den Aufbau von Kompetenzen und ihren jeweiligen Kontext ermöglichen. Zu diesem Zweck werden durch eine Außenbetrachtung der durch das wechselseitige Engagement erzeugten Kulturartefakte die internen Strukturen und Prozesse beschrieben. Ausgangspunkt für die Betrachtung sind die Interaktionen per E-Mail und der Versionsverwaltung. Im Anschluss daran soll das gemeinsame Repertoire der FLOSS-Gemeinschaft auf Zeugnisse für den Kompetenzaufbau hin untersucht werden.

Zu Beginn erfolgt eine Beschreibung des untersuchten Datenmaterials. Dabei handelt es sich um zwei Mailinglisten, welche die FLOSS-Gemeinschaft zur Kommunikation nutzt. Weiterhin gibt es eine Versionsverwaltung auf der die Beiträge zur Softwareentwicklung zentral gesammelt werden. Anhand der Metadaten dieser Interaktionen kann die grobe Arbeitsteilung unter den einzelnen Mitgliedern dargestellt werden. Dadurch wird ersichtlich, wer sich mit welchen Themen beschäftigt. Abgeschlossen wird diese Betrachtung mit einer Längsschnittanalyse der Interaktionen über den gesamten Erhebungszeitraum. Dies erfolgt anhand der quantitativen Darstellung der einzelnen Interaktionen entlang der Teilbereiche und der Anzahl der Projektmitglieder im Projektverlauf. Relevante Ereignisse werden benannt und ihre Folgen für die Projektdynamik abgeschätzt.

Die folgenden Unterkapitel dienen der qualitativen Analyse der Interaktionen auf den Mailinglisten. Auf Grundlage einer Beitragsanalyse werden Kommunikationsinhalte und die Diskussionskultur beschrieben. Anhand von Beispielen wird die Nutzung der Mailinglisten zum problemorientierten Lernen vorgestellt. Diese thematische Betrachtung soll mit einer Analyse aus individueller Perspektive vervollständigt werden. Unterschiedliche Interaktionspraxen und die Differenzierung unterschiedlicher Rollen in der Projektgemeinschaft werden am Beispiel von zwei Mitgliedern der Projektgemeinschaft veranschaulicht.

Abschließend erfolgt eine Bewertung der Interaktionen in Bezug auf den Kompetenzaufbau. Diese Analyse wird ergänzt mit Selbstaussagen von einer Person der Entwicklungsgemeinschaft, die per E-Mail im Rahmen dieser Forschungsarbeit befragt werden konnte.

Analyse von qBittorrent

Das Programm *qBittorrent* ermöglicht den Zugang zu dem Filesharing-Netzwerk *Bittorrent*.[56] Nach Installation auf dem Computer können die Nutzer:innen dieser Softwa-

[56] Bittorrent ist ein Protokoll für ein Filesharing-Netzwerk. Es ermöglicht die Verteilung großer Datenmen-

re beliebige Dateien mit anderen Mitgliedern dieses Netzwerkes direkt austauschen. Direkt bedeutet, dass die Dateien nicht vorher auf einen Server hochgeladen werden müssen, damit anschließend andere diese dann auf ihre eigenen Computer herunterladen können. Der Austausch erfolgt stattdessen *Peer-to-Peer* zwischen den Computern der anbietenden und der nachfragenden Person. Ausgangspunkt für die Entwicklung dieses Programms war die Schaffung einer freien Alternative zu dem proprietären Programm *µTorrent*. Dieses hat eine vergleichbare Funktionalität, ist aber in der kostenfreien Version mit Werbeeinblendungen versehen.

6.1 Beschreibung des Datenmaterials

Für die vorliegende Arbeit relevant waren die Kommunikationsinhalte der beiden zentralen Mailinglisten und die Aktivität bei der Versionsverwaltung. Beide Mailinglisten sind Teil des FLOSS-Portals Sourceforge und die Versionsverwaltung wird über das Portal GitHub betrieben. Der Erhebungszeitraum umfasst neun Jahre und vier Monate, beginnend vom Start der kollaborativen Entwicklung bis zur Erreichung der Programmversion 3.2. Insofern ist das Programm weit über die erste stabil laufende, funktionsfähige Version 1.0 hinausgekommen und kann damit als erfolgreich angesehen werden. Dies ist für die Forschung in dieser Arbeit relevant, da als Kriterium für die Auswahl dieses FLOSS-Projektes eine Entwicklungsgemeinschaft untersucht werden sollte, die möglichst viele Schritte im Produktlebenszyklus der Softwareentwicklung durchlaufen hat.

Insgesamt waren im Erhebungszeitraum 114 Personen an der Entwicklung beteiligt. An Diskussionen auf der Mailingliste der Entwickler:innen, im Folgenden Dev-ML (Developer Mailing List) genannt, beteiligten sich im Zeitraum Oktober 2007 bis August 2015 21 Personen. Im öffentlich zugänglichen Archiv der Mailingliste sind 292 E-Mails dokumentiert, von denen 291 analysiert wurden. Die zweite untersuchte Mailingliste des Projekts war die der Übersetzer:innen: Trans-ML (Translator Mailing List). Im Erhebungszeitraum Mai 2006 bis September 2015 verfassten 18 Personen insgesamt 169 E-Mails, von diesen wurden 168 E-Mails in den Datenkorpus aufgenommen. E-Mails ohne inhaltlichen Bezug zur Mailingliste oder Mehrfachsendung der gleichen E-Mail wurden als Fehlsendung nicht in den Datenkorpus einbezogen.

Das Engagement auf der projekteigenen Versionsverwaltung aller Projektmitglieder im Erhebungszeitraum September 2006 bis September 2015 belief sich auf 6190 Beiträge, die von insgesamt 89 Personen eingereicht wurden. Erfasst wurde, wann welche Person einen Beitrag zum Projektfortschritt eingereicht hat. Eine qualitative Auswertung wurde nicht durchgeführt, da diese im Rahmen dieser Dissertation nicht leistbar gewesen wäre.

gen in einem Netzwerk. Beim Herunterladen von einer Datei von einem Server mit dem HTTP oder FTP-Protokoll wird die Datei als Ganzes vom Server auf den Zielcomputer (sogenannte Clients) übertragen. Die gesamte Upload-Last liegt beim Server, was bei vielen simultanen Downloads den Server überlasten kann. Das Bittorrent-Protokoll teilt die Datenmenge in kleine Pakete auf und überträgt diese sequentiell an die Klient:innen. Die Klient:innen wiederum fungieren zusätzlich als Server für andere Klient:innen, sobald sie ein komplettes Datenpaket vom Server heruntergeladen haben. So wird die Upload-Last von dem einen Server auf alle anfragenden Klient:innen verteilt.

Eine derartige Prüfung wäre beispielsweise anhand der Codequalität und des Umfangs möglich, wenn diese mit früheren Beiträgen der einreichenden Person verglichen werden würde. Bei den durch die Versionsverwaltung archivierten Beiträge ist zu beachten, dass jede Änderung am Repositorium als ein separater Beitrag gewertet wurde.

6.2 Engagement in den Teilbereichen

Die folgende Abbildung zeigt das Engagement aller Projektbeteiligten bezogen auf die Teilbereiche. Diese wurde aus dem Vergleich der bekannten Teile der E-Mail-Adressen generiert.[57] Jeder Kreis steht für einen der Teilbereiche. Die Zahlen geben die Anzahl der Personen wieder, die in dem jeweiligen Bereich aktiv gewesen sind. Zahlen in den Schnittbereichen der Kreise spiegeln die Personenanzahl wider, die in mehreren Bereichen aktiv waren.

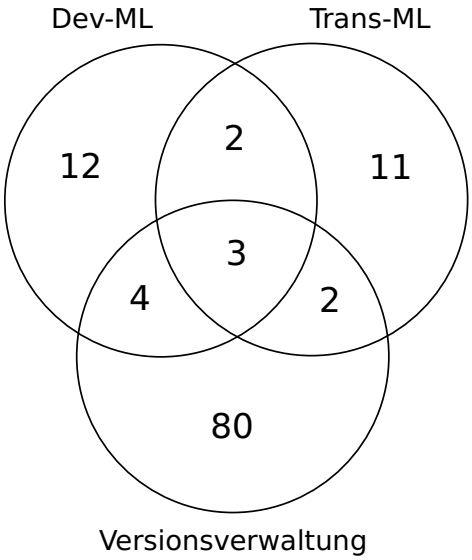

Abbildung 3: Engagement in den Mailinglisten und der Versionsverwaltung

Die Zahlen zeigen, dass die deutliche Mehrheit aller Personen in nur einem Teilbereich aktiv ist. Die überwiegende Mehrheit nutzt nicht die Möglichkeiten, auf beiden öffentlichen Mailinglisten zu kommunizieren. Daraus kann gefolgert werden, dass die Bereiche Entwicklung und Übersetzung sich personell stark abgrenzen. Das ist plausibel, da Sprachkenntnisse oft durch Herkunft und Enkulturation bestimmt sind. Zwei der drei Personen, die in allen drei Bereichen aktiv sind, sind die beiden Projektmaintainer:innen (im Folgenden mit *PM* abgekürzt) des Projekts. Durch die Praxis, dass die Übersetzer:innen ihre Projektbeiträge per privater E-Mail an die PM übermittelten, ist die Nutzung der Versionsverwaltung für sie nicht obligatorisch. Die später eingerichtete Trans-ML Mailingliste diente dann später als Werkzeug für die Bereitstellung aktualisierter Übersetzungen.

[57] Die Daten sind nur begrenzt belastbar. Eine Person, die mit einer E-Mail-Adresse auf einer Mailingliste kommuniziert und mit einer anderen E-Mail-Adresse bei der Versionsverwaltung GIT registriert ist, würde demnach als zwei unterschiedliche Personen erfasst werden.

6.3 Engagement im Projektverlauf

Die Abbildung 4 als gestapeltes Säulendiagramm veranschaulicht die Entwicklungsaktivitäten aufgeteilt nach Teilbereichen im Erhebungszeitraum. Über den Erhebungszeitraum hinweg sind die Aktivitäten in den Teilbereichen diskontinuierlich. Die Verteilung aller Beiträge zeigt einen starken Überhang an Aktivitäten auf der Versionsverwaltung. Dies umfasst Einsendungen auf den beiden Mailinglisten und Projektbeiträge über die Versionsverwaltung.

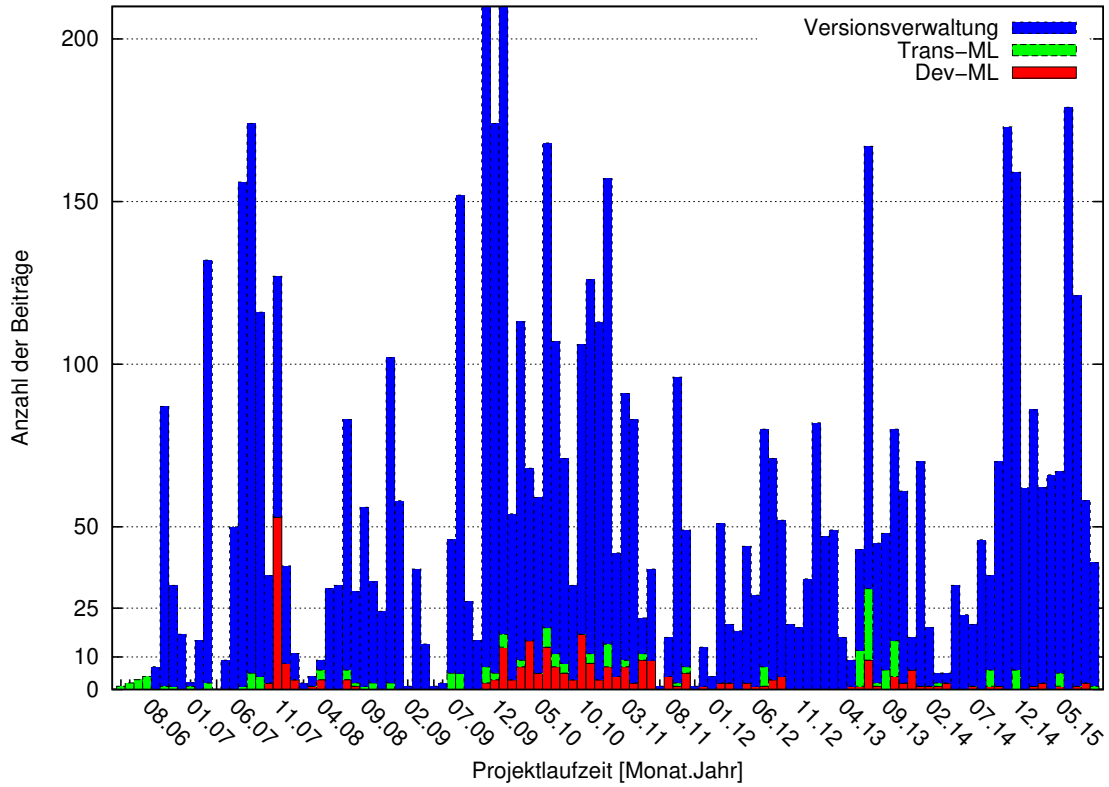

Abbildung 4: Engagement der Entwicklungsgemeinschaft im Erhebungszeitraum

Wichtige kommunikatorische Ereignisse, erkennbar an dem hohen E-Mail-Aufkommen, sind die Eröffnung der Dev-ML im Oktober 2007, die Resonanz nach Veröffentlichung der Programmversion 2.0 im Oktober 2009 und der PM-Wechsel im Juni 2013. Kurz vor Veröffentlichungen neuer Programmversionen gibt es auf beiden Mailinglisten eine erhöhte Aktivität, um die letzten Fehler zu beheben. Auf solch ein Ereignis folgt eine Zeit der Kommunikationsarmut, wie sie zum Beispiel nach der Veröffentlichung von Version 3.0 im Juli 2013 eintrat. In dieser Zeit werden die Fehlerberichte der Nutzer:innen auf einer eigenen Plattform (Bug-Tracker) gesammelt, welche dann für die kommenden Programmversionen abgearbeitet werden. Um die einzelnen Ausschläge in den Teilbereichen zu interpretieren, ist die Berücksichtigung der jeweiligen Mitgliederzahl hilfreich.

6.4 Aufbau der Entwicklungsgemeinschaft

Die Vermutung liegt nahe, dass personelles Wachstum der Entwicklungsgemeinschaft mit einem erhöhten Kommunikationsaufkommen einher geht. Abbildung 5 verdeutlicht den Zuwachs an Mitgliedern im Rahmen der Entwicklungsgemeinschaft. Jeder Graph zeigt die Entwicklung eines Teilbereiches. Gezählt wurde die Zeitpunkte, an denen sich Personen das erste Mal in einem der Teilbereiche engagierten. Personen, die in mehreren Teilbereichen aktiv waren, wurden in jedem einzelnen Bereich separat erfasst. Das Schrumpfen der Entwicklungsgemeinschaft kann nicht direkt gemessen werden, da es keine offiziellen Ausstiegsprozeduren gibt.

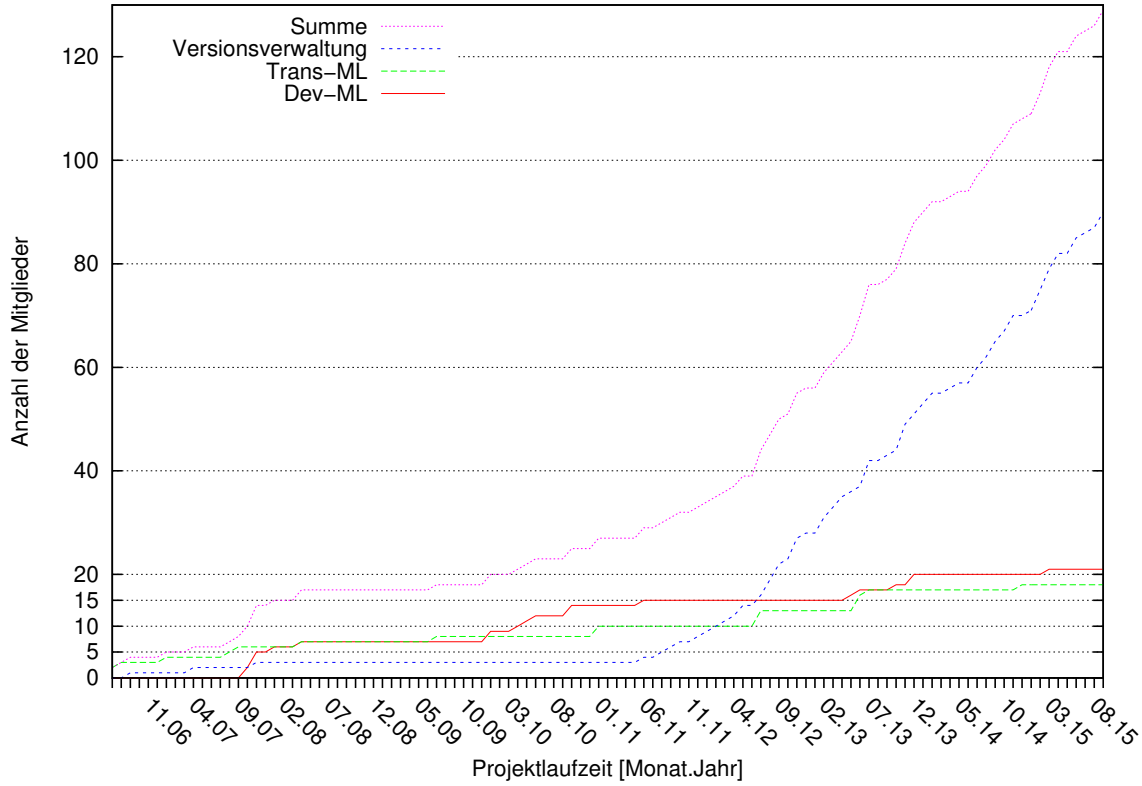

Abbildung 5: Gemeinschaftsentwicklung im Erhebungszeitraum

Im Folgenden werden die drei Projektteilbereiche genauer betrachtet. Die Dev-ML zeigt ihren ersten großen Zuwachs an neuen Mitgliedern direkt nach ihrer Gründung im September 2007. Der Inhalt der ersten E-Mail auf der Mailingliste legt die Vermutung nahe, dass einzelne Entwickler:innen schon vorher untereinander in Kontakt standen. Diese nicht öffentliche Kommunikation im Vorfeld wurde durch die Einrichtung der Mailingliste neu strukturiert.[58] Nach einer langen Phase der Stagnation verdoppelte sich die Zahl der Mitglieder auf 14 innerhalb von zehn Monaten. Dieser Prozess erfolgte direkt nach Veröffentlichung der Programmversion 2.0. Der dritte größere Mitgliederzuwachs erfolgte in der Phase vom Mai bis Dezember 2013. Ein Indiz für diese Veränderung könnte der

[58] Einige E-Mails der Mailingliste enthalten Zitatblöcke aus vorangegangenen E-Mails, die nicht über die Mailingliste geschickt wurden und sind damit der Beleg für direkte, nicht-öffentliche Kommunikation zwischen einzelnen Entwickler:innen.

Wechsel des:der PM sein, welcher im Mai 2013 von dem:der neuen PM der Entwicklungsgemeinschaft über beide Mailinglisten mitgeteilt wurde.

Der Zuwachs auf der Trans-ML erfolgte kontinuierlicher. Zwei kleinere Sprünge im Graphen sind auffällig: zum einen im Monat der Veröffentlichung von Version 3.0 (Juni 2012) und zum anderen im Juni 2013 (Wechsel des:der PM).

Auf der Seite der Versionsverwaltung lassen sich drei Phasen unterscheiden:

1. Langsames Wachstum (September 2006 - November 2007) mit anschließender Stagnation für 42 Monate mit einer Gruppengröße von drei Mitgliedern. Die Zuteilung von Zugriffsrechten für neue Mitglieder erfolgte nach Einzelfallprüfung seitens des:der PM.

2. Schnelles Wachstum ab Juni 2011 mit durchschnittlich einem neuen Mitglied pro Monat.

3. Sehr schnelles Wachstum ab Mai 2012 mit durchschnittlich zwei neuen Mitgliedern pro Monat.

Der Übergang von der ersten Phase zur zweiten Phase könnte mit der Umstellung des Systems der Versionsverwaltung erklärt werden: Ursprünglich kam das Versionsverwaltungssystem *Subversion* zum Einsatz, welches ab Juni 2011 durch *Git* abgelöst wurde. Die dritte Phase mit ihrem schnellem Wachstum begann ab Mai 2012, einem Monat vor der Veröffentlichung von Version 3.0.

Zusammenfassung wichtiger Ereignisse während des Erhebungszeitraums

Beim Vergleich der beiden Abbildungen 4 und 5 im Erhebungszeitraum lassen sich folgende Schlussfolgerungen ableiten:

- Start: Beginn der Entwicklung von qBittorrent erfolgte wahrscheinlich durch eine einzige Person (1. PM), die auch im späteren Verlauf des Projektes eine zentrale Position in der Projektentwicklung einnahm. Die ersten sechs Monate (September 2006 bis Februar 2007) in der Geschichte der Versionsverwaltung war sie die einzige Person, die Projektbeiträge einreichte. Unterstützt wurde sie durch einige Übersetzer:innen, die per E-Mail ihre Beiträge an den:die PM schickten. Die Kommunikation mit dem Übersetzungsteam wurde ab Mai 2006 durch die Einrichtung einer öffentlichen Mailingliste nachvollziehbar.

- Erste Strukturierung: Die Dev-ML wurde im Oktober 2007 eingerichtet, die Kommunikation ist nun wie zuvor auch bei der Trans-ML für Interessierte öffentlich einsehbar und kann zum Austausch mit der Entwicklungsgemeinschaft genutzt werden. Die Gemeinschaft bleibt ein kleiner Kern. Die Veröffentlichung der Version 1.0 im April 2008 hat keinen messbaren Effekt für die Einbindung weiterer Mitglieder.

- Kommunikatives Wachstum: Die Veröffentlichung der Version 2.0 (Dezember 2009) weckt das Interesse neuer Programmierer:innen, die nun die Möglichkeiten der Dev-ML nutzen. Es findet eine verstärkte Aktivität auf der Versionsverwaltung statt. Diese geht allerdings nicht auf das Engagement neuer Mitglieder zurück, sondern ist durch die verstärkte Aktivität des:der PM verursacht.

- Zweite Strukturierung: Im Juni 2011 kommt es zur Umstellung der Versionsverwaltung auf das Git-System. Es erfolgt ein stetiger Anstieg der monatlichen Registrierungen auf dem neuen System, welcher bis zum Ende des Erhebungszeitraums anhält. Die meisten der neuen Entwickler:innen reichen nur wenige Projektbeiträge ein und beteiligen sich nicht an den Diskussionen auf den Mailinglisten. Die Kommunikation kommt ab Oktober 2012, kurz nach Veröffentlichung von Version 3.0, für sieben Monate komplett zum Erliegen. Die Gründe dafür lassen sich nicht aus den Inhalten der Mailinglisten ableiten.

- Erneuerung: Ab Mai 2013 kommt es zu einem Zuwachs an neuen Mitgliedern auf beiden Mailinglisten trotz der anhaltenden Kommunikationsstille. Einen Monat später wird der Wechsel des:der PM durch eine neue Person in dieser Rolle bekanntgegeben. Die Zuwachsrate an Registrierungen bei der Versionsverwaltung bleibt anhaltend hoch. Die eingereichten Projektbeiträge fallen allerdings nach einigen Monaten nach einem kurzen Ausschlag auf einen Tiefpunkt im März 2014.

- Kommunikative Stagnation: Im weiteren Verlauf werden beide Mailinglisten nur noch sporadisch genutzt, die Mitgliederzahl stagniert. Nur die Zahl der eingereichten Projektbeiträge erholt sich wieder und hält sich auf hohem Niveau für knapp ein Jahr bis zum Ende des Erhebungszeitraumes.

Die Phasen zeigen die unterschiedlichen Entwicklungen in der Historie des FLOSS-Projektes. Interessant ist die Tatsache, dass sich ab dem Zeitpunkt der Umstellung auf die neue Versionsverwaltung die Nutzungsverhalten ändert: wenig bis keine Kommunikation auf den Mailinglisten trotz Zunahme der Aktivität bei der Versionsverwaltung. Zwei Thesen könnten zur Erklärung dieser Beobachtungen herangezogen werden: Erstens, die neue Versionsverwaltung öffnet den Zugang für Entwickler:innen die sich nicht langfristig in die Entwicklungsgemeinschaft engagieren wollen und zweitens, statt einer Stagnation der Kommunikation findet nur eine Verlagerung auf andere Kommunikationskanäle statt. Anhand der Betrachtung der Kommunikationsinhalte auf den Mailinglisten wird versucht, Belege für die beiden Thesen zu finden.

6.5 Auswertung der E-Mail-Kommunikation

Das folgende Unterkapitel analysiert die Interaktion zwischen den Mitgliedern der Entwicklungsgemeinschaft auf den beiden Mailinglisten. Die quantitative Betrachtung ermöglicht Aussagen zu Kommunikationshäufigkeit und typischen Kommunikationsmus-

tern. Eine qualitative Betrachtung findet mittels Beitragsanalyse statt und liefert Informationen zu Kommunikationsinhalten. Dadurch können Aussagen zu wichtigen kommunikative Ereignissen in der Historie der Projektgemeinschaft getroffen werden und wie die Diskussionskultur in diesem FLOSS-Projekt bewertet werden kann. Ein besonderer Fokus soll dabei auf die Kultur der Anerkennung gelegt werden. Weiterhin soll anhand ausgewählter Beispiele auf beiden Mailinglisten gezeigt werden, wie sie für die Entscheidungsfindung und das problembasierte Lernen genutzt werden.

6.5.1 Kommunikationsanteile

Wie schon in anderen Studien belegt wurde, geht von den Projektmaintainenden der Hauptteil der Kommunikation aus. In der untersuchten Stichprobe beider Mailinglisten entstammen knapp 72% aller E-Mails von den beiden PM. Beide Mailinglisten unterscheiden sich nur marginal in diesem Kommunikationsüberhang. Auf der Trans-ML ist der Anteil mit 68% aller E-Mails von den PM etwas geringer als auf der Dev-ML. Auf dieser wurden 74% aller E-Mails von den PM gesendet. Dementsprechend ist der Anteil von anderen Entwickler:innen und Übersetzer:innen an der Projektkommunikation gering. Die folgende Tabelle zeigt die Anzahl von Personen und die Menge der von diesen verfassten E-Mails auf der jeweiligen Mailingliste. Ausgenommen sind hier die beiden PM.

Mailingliste	Personen	E-Mails pro Person
Dev-ML	8	1
	8	2 -5
	1	6 - 10
	2	11 und mehr
Trans-ML	8	1
	5	2 - 5
	2	6 - 10
	1	11 und mehr

Tabelle 1: Anzahl der Personen und E-Mail-Aufkommen auf den Mailinglisten

Aus der Tabelle sticht der große Anteil der Personen hervor, die sich nur einmalig oder sporadisch am Diskussionsgeschehen auf den Mailinglisten beteiligen. Die Hälfte aller Personen ist auf beiden Mailinglisten mit nur einer E-Mail in Erscheinung getreten und weitere 41% der Beteiligen haben sich nur sporadisch mit unter fünf E-Mails im gesamten Erhebungszeitraum beteiligt. Daraus geht hervor, dass der Hauptteil der Kommunikation auf den Mailinglisten nur von einem kleinen Kern von Entwickler:innen und Übersetzer:innen stammt. Diese Beobachtung bestätigt die Erkenntnisse, die zuvor auch in anderen Studien über die Kommunikation in FLOSS-Projekten gewonnen wurden. Die Entwicklung des Schalenmodells wie bei Nakakoji et al. 2003, bei dem die Hauptentwicklungsarbeit von einem kleinen Kern an Entwickler:innen erfolgt, die von einer großen

Masse an peripheren Entwickler:innen sporadisch unterstützt werden, konnte damit auch in diesem FLOSS-Projekt bestätigt werden.

6.5.2 Weitere Kommunikationskanäle

Einschränkend muss festgehalten werden, dass die Erhebung der Kommunikation über die Mailinglisten keine Vollerhebung sämtlicher Kommunikation innerhalb des FLOSS-Projektes darstellt. Bei der Analyse der E-Mails ergaben sich verschiedene Hinweise über weitere Kommunikationskanäle, die von den Entwickler:innen genutzt wurden. Diese konnten aus nachfolgend dargestellten Gründen im Rahmen dieser Arbeit nicht in die Datenbasis mit aufgenommen werden: Dies sind zum einen verschiedene E-Mails auf den Mailinglisten, die Zitate aus vorherigen Mails enthielten, die nicht über die Mailinglisten geschickt wurden. Dies deutet darauf hin, dass nichtöffentlicher, direkter E-Mail-Kontakt zwischen den Entwickler:innen bestand. Zum anderen gibt es auch noch weitere Kommunikationskanäle abseits der E-Mail-Kommunikation, deren Existenz aus einigen E-Mails abgeleitet werden kann.

Dies ist beispielsweise der *Bug-Tracker*[59]. Dieser dient der Erfassung und Verwaltung von Fehlermeldungen. Innerhalb der Fehlerverwaltung auf der Bug-Tracker-Plattform können Nachrichten verschickt werden. Diese werden oftmals dafür genutzt, Nachfragen an die Person zu richten, die den Fehler gemeldet hat. Dadurch können die Entwickler:innen Informationen über den Kontext gewinnen, der zu dem Fehler führte. Teilweise werden in den Kommentaren auch Lösungsmöglichkeiten für einen Fehler diskutiert. Parallel dazu gibt es auch synchrone Kommunikationskanäle als Chat, wie es diese E-Mail zeigt:

> „Dear developers,
>
> I would like to remind you all that we have an IRC channel and that it would be nice if people involved in qBittorrent project tried to use it as much as possible.
>
> #qbittorrent on Freenode[60]
> #libtorrent on Freenode is also a good idea
>
> Kind regards, [Name PM].“ (Dokument 130)

In dieser E-Mail weist der:die PM auf die Nutzung der beiden IRC-Kanäle von qBittorrent hin. Auch liegt hier die Vermutung nahe, dass die direkte Kommunikation per Chat präferiert wird. In einer weiteren E-Mail des:der PM berichtet die Person über ein neues

[59] Für das qBittorrent-Projekt nutzte die Entwicklungsgemeinschaft im Erhebungszeitraum den Bug-Tracker auf der FLOSS-Entwicklungsplattform Launchpad https://bugs.launchpad.net/qbittorrent/. Diese stellt neben dem Bug-Tracker auch zahlreiche andere Werkzeuge für die FLOSS-Projekte zur Verfügung.

[60] Freenode ist ein Chat-System, das über den Browser genutzt werden kann. Ziel der Entwickler:innen von Freenode war es, eine Kommunikationsplattform für die Freie-Software-Bewegung anzubieten.

Forum, dass ab diesem Zeitpunkt zusätzlich als Werkzeug für die Projektkommunikation zur Verfügung steht:

> „Hi all,
>
> Thanks to a motivated qBittorrent user, we now have an official qBittorrent forum: http://forum.qbittorrent.org
> If you are interested, please register there. I hope this place will get crowded soon :)
>
> Kind regards, [Name PM].“ (Dokument 139)

Ebenfalls gibt es ein qBittorrent-spezifisches Forum auf Launchpad, welches eigens für die Nutzer:innen der Linux-Distribution Ubuntu gegründet wurde.[61] In diesem Forum werden Fragen von Nutzer:innen der Software gestellt und von erfahrenen Nutzer:innen und Entwickler:innen beantwortet. Diese Kommunikationskanäle haben eine niedrige Einstiegsschwelle. Im Forum melden sich auch Nutzer:innen, die keine eigenen Beiträge zu der FLOSS-Entwicklung leisten wollen, aber an der fehlerfreien Nutzung interessiert sind. Hier erfolgt die Kommunikation also nicht nur innerhalb der Gemeinschaft der Entwickler:innen, sondern über deren Grenzen hinweg. Dies schließt jedoch nicht aus, dass auch Entwickler:innen diese Kommunikationskanäle zur Abstimmung ihrer Entwicklungsarbeit nutzen.

6.5.3 Themen auf den Mailinglisten

Um die Bedeutung der Mailinglisten für das Lernen ihrer Nutzer:innen einschätzen zu können, bedarf es zunächst einer Beschreibung dessen, wozu die jeweilige Mailingliste genutzt wird. Bei der Analyse der Kommunikationsinhalte zeigten sich vier relevante Themenbereiche, die wiederholt Gegenstand von E-Mail-Einsendungen waren. Die Themen sind anhand ihrer Relevanz mit abnehmender Kodierhäufigkeit geordnet.

Release-Ankündigung

Wird eine neue Version der Software, ein sogenanntes Release, durch die:den PM veröffentlicht, wird dies über die Mailingliste bekanntgegeben. Jedes neue Release beinhaltet eine Verbesserung der vorhergehenden Version. Diese wurde beispielsweise mit neuen Funktionen ausgestattet oder es wurden Fehler korrigiert. Eine Release-Ankündigung erfolgt nur durch den:die PM und impliziert eine Handlungsaufforderung an die anderen Entwickler:innen. Sie sollen ihre Arbeit mit der aktualisierten Programmversion fortsetzen, damit es später nicht zu Versionskonflikten kommt. Dementsprechend sind die Release-Ankündigungen ein fester Bestandteil der Nachrichten auf der Dev-ML. Auf der Trans-ML sollen sie die Übersetzer:innen im Fall von neuen Begriffen oder Wortgruppen

[61] Das Unterforum für qBittorrent für die Ubuntu-Nutzer:innen befindet sich auf https://answers.launchpad.net/ubuntu/+source/qbittorrent/ .

daran erinnern, deren Übersetzungen einzureichen. Nur dann kann eine Lokalisation der Software für die kommende Softwareversion berücksichtigt werden:[62]

> „Hi,
>
> I have just uploaded qBittorrent v2.1.1 bugfix release. I had to release so shortly after v2.1.0 due to the number of bugs reported and fixed. One of the bugs is quite important: Torrents added from magnet links were not reloaded on restart. This issue was introduced in v2.1.0. A lot of other issues were fixed, it is strongly recommended to update.
> Here is the Changelog[63]:
>
> * Wed Jan 20 2010 - [...][PM] - v2.1.1 -
> BUGFIX: Fix compilation with Qt4.4 -
> BUGFIX: Save torrent metadata so that it does not have to be re-downloaded on restart (Magnet links) -
> [...]
> I18N: Updated translations (Hungarian, Chinese, Russian)
>
> Best regards, [Name PM].“ (Dokument 91)

Diese Release-Ankündigung zeigt den Aufgabenbereich der:des PM auf. Diese:r ist nicht nur Entwickler:in, sondern diese Rolle beinhaltet auch die Verwaltung und Strukturierung des Projekts. Dazu gehört auch zu entscheiden, wann eine neue Programmversion veröffentlicht wird. Zu diesem Zweck werden Ziele definiert, nach deren Erreichung eine neue Version veröffentlicht wird. In diesem Beispiel ist es die Behebung einer bestimmten Anzahl von Fehlern im Programm und die Sprachaktualisierung für verschiedene Sprachen. Die kontinuierliche Veröffentlichung neuer Versionen ist auch als Signal an neue Nutzer:innen und Entwickler:innen zu deuten, dass das FLOSS-Projekt eine engagierte Gemeinschaft hat und es sich bereits um eine stabil laufende Software handelt.

Handlungsangebot

Handlungsangebote dienen der Koordinierung der anstehenden Arbeiten. Die Entwickler:innen signalisieren damit ihre Bereitschaft, bestimmte Probleme zu beheben oder neue Funktionen zu implementieren. Für die:den PM kommt zusätzlich noch die Aufgabe hinzu, in regelmäßigen Abständen eine neue Programmversion zu veröffentlichen. Zu diesem Zweck werden Anforderungen an eine zukünftige Programmversion definiert. Sind alle Bedingungen erfüllt, also alle gewünschten Fehlerkorrekturen und gegebenenfalls neue Programmmodule in den neuen Entwicklungszweig aufgenommen, wird dieser mit einer höheren Versionsnummer versehen und offiziell veröffentlicht. Die neue Version gilt dann als aktueller Entwicklungsstand. Um diese Arbeit durchführen zu können,

[62] Als Lokalisation von Software wird die Sprachübersetzung der Bedienoberfläche bezeichnet. Sprache wird bei diesem Konzept an einer bestimmten Region festgemacht.

[63] Changelog ist das Änderungsprotokoll. Es beinhaltet alle grundlegenden Änderungen am Programm, die seit der vorherigen Version durchgeführt wurden.

wird durch den:die PM daran erinnert, dass die Entwickler:innen und Übersetzer:innen die dafür notwendigen Entwicklungsbeiträge einreichen. Danach kann das Release offiziell durchgeführt werden.

> „Hi all,
>
> I am currently interested in developing web-interface[64], headless-running[65] and windows-port[66], but I can't decide which one is needed more. Please prioritize the TODO list, see https://blueprints.launchpad.net/qbittorrent/[67] .
>
> Regards, [Name Entwickler:in]" (Dokument 21)

In dieser E-Mail äußert ein:e Entwickler:in den Wunsch, ein Aufgabengebiet zu übernehmen und macht drei Vorschläge. Diese Vorschläge konkretisiert die Person auf der projekteigenen Sammlung für gewünschte Programmfunktionen. Sie bittet die Projektgemeinschaft darum, die für das Projekt wichtigste Funktion zu benennen und damit die Entwicklungsrichtung in eine bestimmte Bahn zu lenken.

Handlungsaufforderung

Wünsche oder Handlungsaufforderungen werden fast ausschließlich durch den:die PM formuliert. Sie richten sich entweder an die Entwicklungsgemeinschaft als Ganzes oder an einzelne Entwickler:innen. Wünsche, die an alle Entwickler:innen gerichtet werden, fordern zum Testen der aktuellen Programmversion auf, weisen auf neue Programmversionen oder Patches hin, die für die weitere Arbeit der Entwickler:innen von Bedeutung sind oder benennen notwendige Aufgaben, die noch erledigt werden müssen. Bei Handlungsaufforderungen an einzelne Entwickler:innen handelt es sich fast ausschließlich um kritische Rückmeldungen zu deren Arbeit. Dies sind beispielsweise Fehler im Programmcode oder ein unerwünschter Programmierstil. Die Kritik wird sachlich kommuniziert, begründet und ist mit der Bitte versehen, die Kritik zukünftig zu berücksichtigen.

> „[Name Entwickler:in],
>
> you will have svn[68] access soon but I need you not to make qBittorrent crash. Even if this is svn, I want it to be pretty stable. Apptly, you didn't test the patch enough. I downloaded a torrent, and displayed properties and it displayed a white bar (no downloaded pieces, ok). I waited a little, it reached 0.4% so I tried to display the properties again and it crashed:

[64] Ein 'web-interface' ermöglicht die Steuerung durch den Browser.

[65] Als 'headless' werden Programme bezeichnet, die im Hintergrund laufen. Sie können so konfiguriert werden, dass sie ohne direkte Interaktion mit dem:der Nutzer:in ihre Aufgaben erfüllen.

[66] Die Windows-Portierung ermöglicht es der Software, auch auf dem Betriebssystem Microsoft Windows laufen zu können.

[67] Diese Funktionalität von Launchpad ermöglicht es, Vorschläge für neue Programmfunktionen zu verwalten. Ideen für neue Funktionen werden gesammelt, priorisiert und gegebenenfalls einer Person zur Implementierung zugeordnet.

[68] 'SVN' ist das Akronym für die Versionverwaltung Subversion.

ASSERT: "idx >= 0 && idx < s" in file /usr/include/qt4/QtCore/qvarlengtharray.h, line 91

Also, I'll need your Full name [Name Entwickler:in] and your nationality to add you as a developer on the website. I know that developping for a project that is already advanced is difficult and it looks like you code well. I hope you will keep on writing good features[69] for qBittorrent :)

Welcome on board.

Regards, [Name PM]" (Dokument 13)

In dieser E-Mail weist der:die PM eine:n Entwickler:in auf einen Fehler hin, der durch eine Modifikation im Quellcode verursacht wurde. Die Änderung, die einen Fehler beheben sollte, verursachte selbst an anderer Stelle einen Programmabsturz. Der:die PM beschreibt das Absturzszenario, damit der:die Entwickler:in dies reproduzieren und beheben kann. Weiterhin wird der:die Entwickler:in gebeten, Änderungen an der Codebasis erst nach einem ausführlichen Test des neuen Codes vorzunehmen, um die Stabilität der Software zu gewährleisten.

Die Person engagiert sich erst seit Kurzem in dem FLOSS-Projekt und hat selbst noch keine Möglichkeit, selbst Änderungen an der Codebasis vorzunehmen. Neue Beiträge werden an den:die PM direkt geschickt und diese:r pflegt diese dann in die gemeinsame Codebasis in der Versionsverwaltung ein. Die Schreibrechte werden der Person allerdings in Aussicht gestellt. Weiterhin soll die Person auch offiziell auf der Projekt-Homepage als Entwickler:in aufgeführt werden. Diese Form der Anerkennung für die bisher geleistete Arbeit wird damit auch nach außen sichtbar. Die betreffende Person rückt dadurch aus dem Kreis der peripheren Entwickler:innen zum Kernteam vor.

Problemschilderung

In der Softwareentwicklung ist das Auftreten von Problemen ein immer wiederkehrender Prozess. Es kann als fester Bestandteil des Entwicklungszyklus aufgefasst werden. Dieser besteht aus Programmieren, Testen und Fehlersuche. Dementsprechend nimmt auch die Kommunikation über Fehler, insbesondere auf der Dev-ML, einen großen Teil in der Projektkommunikation ein.

„I tested the new proxy code.

There was a little typo '[Preferences' instead of 'Preferences' in downloadThread.cpp. I've just fixed it.

I can test HTTP proxy here. search engine is working but I get 'Invalid Url' errors when trying to download files. Apptly my code in downloadThread.cpp is not good. If someone has a bit of time (maybe [Name Entwickler:in]), he can

[69] Mit 'features' sind weitere Programmfunktionen gemeint - also die Implementation neuer Funktionen, welche die FLOSS zukünftig abdecken soll.

have a look at it, libcommoncpp doc is here: http://www.gnutelephony.org/-doxy/bayonne2/a00242.html

My code is :

subDownloadThread::subDownloadThread(QObject *pt, QString url) : QThread(pt), url(url), abort(false){ url_stream = new ost::URLStream(); // Proxy support QSettings
[...][Fortsetzung Quellcode]

I tried with and without "url_stream->setProxyAuthentication(ost::URLStream::-authBasic, "auth");". Neither works. I don't understand what the problem is (yet).

Regards, [Name PM].“ (Dokument 27)

Diese E-Mail, die von der:dem PM versandt wurde, ist ein typisches Beispiel eines solchen Entwicklungskreislaufs. Der:die PM hat die Software getestet und Fehler behoben. Während die Behebung eines Tippfehlers unproblematisch war, bereitet der Download-Prozess über einen Proxy-Server Schwierigkeiten.[70] Um das Problem zu lösen, hat der:die PM selbst einen Algorithmus programmiert. Dieser ist allerdings nicht in der Lage, das Problem zu lösen. Der:die PM informiert die Entwicklungsgemeinschaft über die bisherigen Versuche und schickt den Quellcode über die Mailingliste. Er:sie erhofft sich Hilfe von einer bestimmten Person, die allerdings laut einer Antwortmail keinen Fehler im Quellcode findet und selbst für die Problemlösung gerade zu beschäftigt ist. Der:die PM findet im Nachgang dieser E-Mail schließlich selbst den Fehler und löst das Problem ohne fremde Hilfe. Die Entwicklungsgemeinschaft wird anschließend über den Lösungsweg in einer separaten E-Mail informiert.

6.5.4 Diskussionskultur

Welche Diskussionskultur sich in einer FLOSS-Gemeinschaft herausbildet, ist höchst unterschiedlich und hängt von vielen Faktoren ab. Eine spezifischen Kultur hat einen historischen Kontext, der die aktuellen Prozesse prägt. Dennoch ist Kultur nicht statisch, weil das Umfeld, die Mitglieder selbst und die Mitgliederstruktur einem ständigen Wandel unterworfen sind. Für die Betrachtung der Diskussionskultur in diesem FLOSS-Projekt werden zwei Merkmale betrachtet: Diskussionstiefe und die gegenseitige Anerkennung.

Die Diskussionstiefe soll Aufschluss darüber geben, wer sich an Diskussionen beteiligt und wie kontrovers diskutiert wird. Diese Betrachtung erfolgt auf Grundlage der Kommunikationsmetadaten. Die Anerkennung hingegen wird qualitativ und quantitativ betrachtet. Damit sollen die folgenden Fragen beantwortet werden: Ist das Ausdrücken von

[70] Ein Proxy-Server ist ein Computer, der stellvertretend für den Computer der:des Nutzer:in Informationen aus einem Netzwerk sendet und empfängt und die Informationen an den:die Nutzer:in weiterleitet. Er ist in der Netzwerkverbindung zwischen den Computer der:des Nutzer:in und den Zielcomputer geschaltet, so dass diese nicht direkt miteinander kommunizieren.

Anerkennung Teil der E-Mail-Kommunikation und falls ja, wie wird Anerkennung geäußert?

Diskussionstiefe

Ein Grad für die Beschreibung der Interaktion ist die Tiefe der Diskussionen. Dabei wird gemessen, wie stark sich die Mitglieder der Entwicklungsgemeinschaft in ihren E-Mails aufeinander beziehen. Die E-Mails wurden nach den Merkmalen Diskussionsstart und Folgebeitrag kategorisiert. Es zeigte sich, dass sich auf der Dev-ML nur 30% aller Beiträge inhaltlich auf einen vorherigen Beitrag beziehen. Auf der Trans-ML ist dies hingegen bei 48% der E-Mails der Fall. Daraus kann abgeleitet werden, dass ein Großteil der E-Mails nicht zu einer Diskussion innerhalb der Entwicklungsgemeinschaft führt.

Doch wer sind die Personen, die Diskussionen starten beziehungsweise sich selbst an Diskussionen beteiligen? Bei den Folgebeiträgen ist das Verhältnis zwischen Entwickler:innen und den PM fast ausgeglichen. 53% der Folgebeiträge wurden durch die PM verfasst, der Rest von den anderen Entwickler:innen. Ähnlich sieht es auf der Trans-ML aus. Auch hier haben die beiden PM einen leichten Überhang bei den Folgebeiträgen.

Bei der Betrachtung der Startbeiträge[71] fällt das Ergebnis deutlich klarer aus. Die beiden PM sind bei der Dev-ML für 83%, bei der Trans-ML für 87% aller Diskussionsstarts verantwortlich. Dies unterstreicht die kommunikative Stärke der Rolle der PM in diesem FLOSS-Projekt. Die beiden PM initiieren drei Viertel aller Diskussionen und antworten genauso oft wie die Gesamtheit aller übrigen Entwickler:innen und Übersetzer:innen. Bei der Betrachtung der beiden PM zeigen sich aber große Unterschiede. Der:die erste PM war aktiver beim Verfassen von Folgebeiträgen, insbesondere auf der Dev-ML als der:die zweite PM im Erhebungszeitraum. Dies kann allerdings auch ein Hinweis auf die sich verändernde Nutzung von Kommunikationskanälen hindeuten, wie zum Beispiel durch den Wechsel der Versionsverwaltung von Subversion auf Git über die FLOSS-Plattform GitHub. Da GitHub eigene Kommunikationsmöglichkeiten bietet, ist zu vermuten, das ein Teil der Kommunikation dorthin ausgelagert wurde. Für eine Prüfung dieser These bräuchte es zusätzlich eine Erhebung der Projektkommunikation auf GitHub die aber nicht Teil der Datenbasis dieser Arbeit ist.

Kultur der Anerkennung

Bei der Beitragsanalyse fiel auf, dass in vielen E-Mails Formen der Anerkennung zum Ausdruck gebracht wurden. Dies geschah entweder als verallgemeinerte Danksagung in Grußformeln oder auch ganz konkret an einzelne Projektmitglieder gerichtet. Für die Art und Weise, wie Anerkennung an einzelne Personen in der E-Mail-Kommunikation geäußert wird, folgen zwei Beispiele:

[71] Jede E-Mail, die sich nicht auf eine vorherige E-Mail bezog, wurde als Startbeitrag kodiert. Dies schließt auch Ankündigungsnachrichten ein, die nicht explizit eine Frage beinhalten oder zu einer Stellungnahmen aufrufen.

„[Zitat E-Mail vom PM]
> Hi,
>
> I have just uploaded qBittorrent v2.4.7. It fixes a few bugs that were
> reported recently.

I was a few days off-line and there was 2 new versions of qbt[72] :)

You rock!“ (Dokument 163)

In dieser E-Mail antwortet ein:e Entwickler:in auf eine Release-Ankündigung des:der PM. Diese:r hat innerhalb weniger Tage zwei neue Softwareversionen veröffentlicht. Jede Veröffentlichung einer Softwareversion stellt, insbesondere für die:den PM, eine hohen Arbeitsaufwand dar. Für die Arbeitsleistung des:der PM drückt diese:r Entwickler:in in einer eigenen E-Mail Anerkennung aus.

Ebenso nutzen auch die PM häufig die Mailinglisten, um sich bei den Entwickler:innen und Übersetzer:innen für deren Engagement zu bedanken. Das folgende Beispiel demonstriert zwei verschiedene Ebenen, auf denen Anerkennung in einem FLOSS-Projekt zum Ausdruck gebracht werden kann:

„Hi,
I proceeded with the release of qBittorrent v2.7.0, as planned.

A big thanks to [Name Entwickler:in 1], [Name Entwickler:in 2] and [Name Entwickler:in 3] for their contribution. qBittorrent project is attracting new developers and this is excellent news.

* Sun Mar 20 2011 - [Name PM] - v2.7.0
[...][Ausschnitt Änderungsprotokoll]
- FEATURE: Added option to skip torrent deletion confirmation ([Name Entwickler:in 2])
- FEATURE: IP address reported to trackers is now customizable
- FEATURE: Inhibit system sleep when torrents are active ([Name Entwickler:in 1])
- FEATURE: Added option to bypass Web UI authentication for localhost
- FEATURE: Added option to disable program exit confirmation
- FEATURE: Added per-torrent ratio limiting ([Name Entwickler:in 3])
- FEATURE: Torrent content list is now sortable
- BUGFIX: Fix compilation with namespaced Qt ([Name Entwickler:in 3])
[...]

Kr[73], [Name PM].“ (Dokument 198)

[72] Die Abkürzung ’qbt’ steht für qBittorrent.
[73] ’Kr’ wird als verkürzte Abschiedsfloskel anstelle von ’Kind regards’ genutzt.

Bei der E-Mail handelt es sich wieder um eine Release-Ankündigung durch den:die PM. Er:sie bedankt sich bei den Entwickler:innen, die zu dieser Softwareversion Beiträge eingereicht haben. Sie werden mit vollem Namen in der E-Mail erwähnt. Damit wird auf der Mailingliste auch das Engagement der Projektmitglieder hervorgehoben, deren Beiträge sonst nur in der Versionsverwaltung erfasst werden.

Weiterhin ist in der E-Mail auch das Änderungsprotokoll eingebettet. Dieses wird von der Versionsverwaltung automatisch erzeugt und beinhaltet alle Kommentare, mit denen jeder Beitrag versehen ist. Das Änderungsprotokoll von einem Versionssprung ist eine Sammlung der Beiträge, die für eine neue Softwareversion zusammengefasst wurden. Also die Summe aller neuen Funktionen, Fehlerkorrekturen oder Änderungen an der Übersetzung. Der:die PM benennt damit die Beiträge auf der Versionsverwaltung und macht sie für alle Leser:innen der Mailinglisten transparent.

Wie oben bereits erwähnt erfolgt die Anerkennung der Leistung der gesamten Projektgemeinschaft oft als allgemeine Ansprache. Die überwiegende Mehrheit dieser Danksagungen (knapp 86%) erfolgten durch die:den PM selbst. Meistens wird der Projektgemeinschaft in der E-Mail einer Release-Ankündigung gedankt. Die folgende E-Mail gibt ein Beispiel einer solchen Anerkennung:

> „Hi all,
>
> I have just received an e-mail notifying me that *qBittorrent has been selected* as one of the three finalists (in Desktop software category) in "*Les etoiles du libre*" *Free Software competition* (http://etoiles-du-libre.org/in French only, sorry). As a consequence, I will have to present qBittorrent in front of a jury/committee on December 12th 2009 so that they can decide what the winning project is.
>
> As you can imagine, since v2.0.0 looks stable enough now, I would like to release qBittorrent v2.0.0 just before this presentation (likely *December 10th*) in order to present latest and greatest version :P This is more interesting and this will increase our chances to winning this competition.
>
> The only thing missing before I can make a release are translations update. Indeed, I'm still missing a lot of translations and I cannot make such an important release without most of the languages (if not all) being up-to-date. If you are part of qBittorrent official translation team, I'm asking you to *please hand me your updates before December 10th*. I hope you can all arrange your schedule to meet this deadline.
>
> This is an important moment in the life of qBittorrent, thank you all for being part of this.
>
> Best regards, [Name PM]." (Dokument 78)

Anerkennung wird in dieser E-Mail des:der PM in mehreren Ebenen transportiert. Erstens, als direkten Dank für das Engagement der Projektmitglieder bei der Entwicklungs-

arbeit. Zweites, als indirekte Anerkennung der Gruppenleistung durch ein Lob der kommenden Version als neueste und beste Version und drittens als Anerkennung der Gruppenleistung von außen.

Die:der PM schreibt, dass das FLOSS-Projekt als Finalist in einem Wettbewerb für freie Software zusammen mit zwei anderen FLOSS-Projekten nominiert wurde. Dieser Fakt allein zeigt, dass die Entwicklung der Software auch außerhalb der Entwicklungsgemeinschaft mit Interesse beobachtet wird. Schließlich gibt dies den Projektmitgliedern die Gewissheit, dass es viele Nutzer:innen gibt, die diese Software einsetzen und deren Entwicklung verfolgen. Die Mühen der Entwicklungsarbeit dienen dann nicht nur dem eigenen Interesse, sondern helfen auch vielen anderen durch die Nutzung der FLOSS.

Wird die Häufigkeit von Danksagungen auf beiden Mailinglisten betrachtet, so enthält etwa jede vierte E-Mail eine Form der Anerkennung. Diese ist entweder an eine bestimmte Person gerichtet oder adressiert die Entwicklungsgemeinschaft als Ganzes. Die meisten Danksagungen stammen von den PM. Formulierungen von Anerkennung, die an eine konkrete Person gerichtet ist, wurden auf der Dev-ML in circa zehn Prozent der E-Mails gefunden. Bei der Kodierung von Anerkennung wurde unterschieden, wer diese verfasst hat und wessen Leistungen gewürdigt wurden. Die hier genannten Fallzahlen entsprechen den E-Mails im Datenkorpus. Bei den Häufigkeiten der Anerkennung können auch mehrere Danksagungen in einer E-Mail enthalten sein. Die Fallzahlen sollen lediglich ein Gefühl für die Verhältnisse ermöglichen. Als Projektmitglieder sind alle Entwickler:innen und Übersetzer:innen mit Ausnahme der PM gemeint.

Ziel	Dev-ML (n=291)	Trans-ML (n=168)
PM	11	8
Projektmitglied	20	4

Tabelle 2: Anerkennung an konkrete Personen

Der größte Anteil der Würdigungen von Leistungen der einzelnen Projektmitglieder auf der Dev-ML stammt von den PM. Sie hoben die Beiträge einzelner Entwickler:innen hervor. Nur ein Drittel der Würdigungen stammt dagegen von den Entwickler:innen, die sich für die Leistungen der PM bedanken. Bei der Trans-ML ist das Verhältnis der Zahlen umgekehrt: Hier sind es mehrheitlich die Übersetzer:innen, die sich für das Engagement der PM bedankten. Bei der Anerkennung, die gegenüber der gesamten Projektgemeinschaft ausgesprochen wurden, sehen die Häufigkeiten wie folgt aus:

Quelle	Dev-ML (n=291)	Trans-ML (n=168)
PM	16	49
Projektmitglied	7	1

Tabelle 3: Anerkennung an die Projektgemeinschaft

Auch bei der Würdigung der Leistungen der Projektgemeinschaft nehmen die PM eine herausragende Stellung ein. Sie drücken auf der Dev-ML mehr als doppelt so oft ihre

Anerkennung für die Projektgemeinschaft aus wie die Entwickler:innen. Auf der Trans-ML sind es fast ausschließlich die PM, die das Engagement der übrigen Projektmitglieder würdigen. Fast jede dritte E-Mail der Trans-ML enthält einen Dank an die Projektgemeinschaft. Meist ist dieser Bestandteil der Abschiedsformel, wenn eine neue Version veröffentlicht werden soll und die dafür erforderlichen Übersetzungen eingereicht werden sollen.

Die Zahlen zeigen, dass die Würdigung von Leistungen einen nicht unerheblichen Teil der E-Mail-Kommunikation ausmacht. Welche Auswirkungen diese Kultur der Anerkennung auf einzelne Mitglieder der Projektgemeinschaft hat, lässt sich anhand der Beitragsanalyse nicht objektiv ermitteln. Es wird allerdings zu vermuten, dass sie eine wichtige Rolle bei der Entwicklungsarbeit spielt. Anerkennung ist eine nicht zu unterschätzende Voraussetzung für die Gruppenkohäsion, die durch wechselseitiges Engagement gefördert wird. Die gegenseitige Anerkennung von Leistung wird als wichtige soziale Praxis interpretiert, die aber erst anhand der Aussagen von den Entwickler:innen selbst im späteren Teil dieser Arbeit in ihren Folgen bewertet werden kann.

6.5.5 Problemlösung bei der Entwicklungsarbeit

Die Kommunikation auf beiden Mailinglisten ist fokussiert auf Probleme und deren Lösung. Doch wie genau sieht ein solcher Problemlösungsprozess aus? Um diese Frage zu beantworten, werden im Folgenden zwei beispielhafte Problemlösungsprozesse vorgestellt. Dies gibt einen Einblick in die Art und Weise des Austauschs zwischen den Mitgliedern der Entwicklungsgemeinschaft auf den Mailinglisten. Die Wiedergabe der E-Mail-Beiträge erfolgt anhand eines Diskussionsthemas. Alle E-Mails, die mit Bezug auf das Diskussionsthema von den Entwickler:innen und Übersetzer:innen verfasst wurden, werden chronologisch wiedergegeben und in einen Kontext eingeordnet. Daran soll demonstriert werden, wie Probleme kommuniziert und Lösungsprozesse koordiniert werden. Generell konnte festgestellt werden, dass ein Problem in diesem FLOSS-Projekt in nur wenigen E-Mails thematisiert wird. Diskussionen, die mehr als fünf Folgebeiträge auslösen, sind eher die Ausnahme. Das erste Beispiel auf der Dev-ML thematisiert die Vorbereitung zur Veröffentlichung der Version 1.0 für das Betriebssystem Windows.

Problemlösung auf der Dev-ML

Ein Problem, welches mit zehn Folgebeiträgen schon zu den größeren Diskussionen gehörte, behandelte das Thema der Portierung der Software für das Betriebssystem Microsoft Windows. Für Nutzer:innen mit diesem Betriebssystem soll die Software als direkt ausführbaren Maschinencode verfügbar gemacht werden. Der aus dem Quellcode kompilierte Maschinencode zeigte bei den Programmtests jedoch nicht die gewünschten Eigenschaften.[74]

[74] Der Prozess des Umwandelns eines in einer Programmiersprache geschriebenen Quellcodes in Maschinencode oder Binärcode wird 'Kompilieren' genannt. Nur Maschinencode kann direkt vom Computer ausgeführt werden.

1. Problembeschreibung durch die:den PM: 27.11.2007 / 22:06 Uhr

„I have just tested qBittorrent on Windows XP (last build from [Name Entwicker:in 1]) and I have to say I'm impressed. It works way better than it used to. It downloads just fine (got a very good speed here). Things that don't work:
- Downloads from URLs[75] (this is not surprised because apptly we have to redefine a callback function in libcurl otherwise it won't work on windows). I think this can be easilly fixed. [Name Entwicker:in 1], you didn't change the libcurl code right? which version of the library are you compiling against?
- Search engine. Once again, not suprising as I don't have python[76] installed on my Windows yet. And even it is were installed, I guess the way we call the python scripts should be improved in order to support windows. This would need some testing I guess but totally feasable.
- HTML code in tray icon tooltip is not interpreted, I don't know the reason because it works everywhere else. I guess there must be something windows doesn't like in it.
- Installed search engine plugins are not detected, there must be an easy way to fix this if I could test.

Things that work fine:
- Systray integration
- [...][Aufzählung fehlerfreier Funktionen]
- Almost everything else :)

Briefly, I have to say that I'm very pleased with this new build. I noticed that [Name Entwicker:in 1] used Qt4.3.1[77], I guess I will try to use it on Windows Vista just to see how it looks :) [Name Entwicker:in 1], I have a VMWare[78] with Windows XP here (and also a real Windows Vista but I don't like to reboot...). If you could help me reproduce your building steps, I guess I could debug a lot :)

Regards, [Name PM].“ (Dokument 42,)

Die E-Mail benennt gleich mehrere Probleme in der getesteten Version der Software für Windows. Das Kompilieren dieser Version hat Entwickler:in 1 übernommen. Auffällig

[75] Die Abkürzung 'URL' steht für *Uniform Resource Locator* und bezeichnet eine konkrete Ressource und Zugriffsart wie zum Beispiel eine Webseitenadresse mit dem Protokoll als Zugriffsart.

[76] Python ist die Programmiersprache in der qBittorrent programmiert wird.

[77] *Qt* ist eine Softwarebibliothek zur plattformunabhängigen Programmierung graphischer Benutzerschnittstellen.

[78] *VMWare* ist eine Software, um einen virtuellen Computer innerhalb eines bestehenden Computers zu simulieren. So kann z.B. eine Software für ein Betriebssystem getestet werden, das nicht auf dem eigentlichen Computer installiert ist.

ist, dass zu den geschilderten Problemen auch Vermutungen zu möglichen Lösungswegen formuliert wurden. Die:der PM bietet auch Mithilfe bei der Problemlösung an, benötigt dazu aber von Entwickler:in 1 weitere Informationen.

2. Antwort vom:von Entwickler:in 1: elf Minuten später

In der Antwort nimmt Entwickler:in 1 Stellung zu den Fragen.

> „[...][Zitat von der PM-Mail: Frage zur Libcurl-Bibliothek]
> I have built libcurl-7.17.1 from source myself, with SSL enabled. I haven't tested downloading from URLs.
>
> [...][Zitat von der PM-Mail: Frage zu Python und Suchmaschinendateien]
> I have no python either and I have not packaged the search engine files.
>
> [...][Zitat von der PM-Mail: Frage zum HTML-Code im Taskleistensymbol]
> This is not the first time this fails, v0.9.3 was the same. The tray message that told me there was a write error, probably because my disk was full was shown with a little icon. So fancy tooltips 't entirely broken I guess.
>
> [...][Zitat von der PM-Mail: Frage zu Suchmaschinen-PlugIns]
> Not packaged, getting python to work first would be a priority.
>
> [...][Zitat von der PM-Mail: Frage nach dem Ablauf des Kompilierens]
> I use a big sh script[79] in msys[80] to build everything and finetune the final build manually. I'm completing automated tuning and then I'll post the build script (called automaton.sh).
>
> By the way, there are 2 screenshots in the=20 http://www.[Adresse Webseite]/-qbittorrent/experimentalbuild/ webdir." (Dokument 43)

Der:die Entwickler:in beschreibt den Ablauf des Kompilierens und antwortet damit auf alle Fragen des:der PM. Die Antwort betrifft die genutzte Software beziehungsweise die Bibliotheken und den Umfang der durchgeführten Tests. Dies ermöglicht es auch den anderen Entwickler:innen, die noch fehlenden Tests nachzuholen und grenzt die Lösungsmöglichkeiten ein. Zugleich kündigt Entwickler:in 1 an, das Skript für das Kompilieren zu veröffentlichen. Damit können andere Entwickler:innen die Ergebnisse reproduzieren.

3. Nachtrag von:vom Entwickler:in 1: circa zwei Tage später

[79] Die Abkürzung 'sh' steht für Shell, eine Schnittstelle die per Kommandozeile die Interaktion mit dem Betriebssystem ermöglicht. Mehrere Befehle können in einen Skript zusammengefasst sein und als Befehlsfolge über die Shell ausgeführt werden.

[80] MSYS ist eine Bibliothek für Windows, um Programme lauffähig zu machen die selbst auf Programme einer GNU/Linux-Umgebung angewiesen sind.

Wie angekündigt, wird das angekündigte Skript veröffentlicht und die Ergebnisse weiterer Tests mitgeteilt.

> „[...][Zitat von der vorheriger E-Mail: Frage zur genutzten Qt-Version]
> > Building with 4.3.2 should be possible too, when I install that in my qemu[81]
> > image.
>
> I am now using Qt 4.3.2, works just as well as 4.3.1.
>
> [...][Zitat von der vorheriger E-Mail: Ankündigung Skript-Veröffentlichung]
> I put the automaton.sh script in http://www.[Adresse Webseite]/qbittorrent/-win32/ and there is also a bin dir with a previous build of v0.9.3, but you shouldn't use that. A build of the svn trunk head was succesful, that's the alpha1 build.
>
> For everyone who wants to try the automaton.sh script:
> - Install mingw http://downloads.sourceforge.net/mingw/MinGW-5.1.3.exe
> - Install msys http://downloads.sourceforge.net/mingw/MSYS-1.0.10.exe
> [...] [Anleitung für die Kompilation mit dem Skript]
> - Make sure you have at least 220MB free diskspace and run the script. - Wait for ... 46 minutes (on a 1.5GHz 1GB RAM machine)
> - Enjoy qBittorrent on windows
>
> Let me know if you run into trouble or if you succeed.
>
> [Name Entwickler:in 1]“ (Dokument 44)

Auffallend hierbei ist, dass diese E-Mail ein Zitat aus einer E-Mail des:der PM enthält, die nicht über die öffentliche Mailingliste geschickt wurde. Dies zeigt, dass bestimmte Entwicklungsprozesse in enger Zusammenarbeit zwischen einzelnen Entwickler:innen erfolgen, die unter Ausschluss der anderen Projektmitglieder erfolgen. Dies ist ein Beleg für den direkten Austausch zwischen einigen Entwickler:innen, der einen eigenen Kommunikationskanal darstellt und auf eine besonders enge Zusammenarbeit zwischen den beiden Entwickler:innen hinweist. In diesem Fall ist es ein Hinweis seitens des:der PM, der hauptsächlich für Entwickler:in 1 bedeutsam ist.

4. Ankündigung des:der PM: wenige Sekunden später

> „Very good. I'll try when I get back home :)
> [Name PM].“ (Dokument 45)

[81] *Qemu* ist eine Virtualisierungssoftware, die die Hardware von Computerarchitekturen emuliert. Damit kann die Lauffähigkeit von Software auf unterschiedlichen Computern getestet werden, ohne dass diese physisch existieren.

Kurze Ankündigung, dass der:des PM das von Entwickler:in 1 veröffentlichte Skript in Kürze testen wird. Die kurze Reaktionszeit für diese Antwort zeigt, dass gerade beide Entwickler:innen zeitgleich am Computer tätig sind.

5. Antwort der:des PM: circa fünf Stunden später

> „I experiencing a problem with svn. copying the exes[82] and dlls[83] in bin/ folder didn't work for me.
>
> I had to manually create a "local" folder in "C:\msys\1.0\". Then I copied the subversion folder in the local folder:
> $ ls /c/msys/1.0/local/ bin iconv licenses share
> [...][weitere Befehle aus dem Skript folgen]
>
> I think that when I will have some time, I will create a wiki page for this, taking all information in Peter's last mail so that everybody can try. Thanks [Entwickler:in 1] BTW[84] ;)
>
> best regards, [Name PM].“ (Dokument 46)

Der:die PM schildert die Ergebnisse aus dem Test des Skripts, welches allerdings nicht funktioniert. Er:sie konnte das Problem jedoch selbst beheben und das Skript erfolgreich ausführen. Um den anderen Entwickler:innen dies ebenfalls zu ermöglichen, will der:die PM die Anleitung für das Skript in das projekteigene Wiki aufnehmen. Damit wird das Skript Teil der Projektdokumentation und stellt damit, neben der FLOSS selbst, eine Weiterentwicklung des gemeinsamen Repertoires der Praxisgemeinschaft dar.

6. Nachtrag des:der PM: wenige Sekunden später

> „I added help to compile qBittorrent on windows on the wiki: http://60gp.ovh.net/~dchris/wiki/index.php?title=Windows_port_for_qBittorrent
> This is based on [Name Entwickler:in 1] last mail. Please try it and to not hesitate to fix bugs on windows now. I would love to get v1.0.0final working on Windows as we first planned.
>
> Regards, [Name PM].“ (Dokument 47)

Die:der PM hat die Dokumentation aktualisiert und fordert die Entwickler:innen auf, die anderen Fehler der Software für Windows zu beheben. In diesem Zuge verweist er:sie auf den Zeitplan zur Veröffentlichung der Version 1.0 der Software für Windows, der eingehalten werden soll. Die Version 1.0 wird in der Regel als die erste, stabil laufende Softwareversion mit allen Basisfunktionen angesehen und stellt damit einen wichtigen

[82] Bei Dateien mit der Endung 'exe' für 'executable' handelt es sich um ausführbare Dateien.
[83] Die Dateiendung DLL steht für 'Dynamic Link Library'. Dies sind Programmbibliotheken, auf die andere Programme zugreifen können.
[84] Das Akronym BTW steht für 'by the way'.

Entwicklungsschritt für das gesamte FLOSS-Projekt dar.

7. Problemmeldung beim Kompilieren durch den:die PM: circa vier Stunden später

> „Using automaton.sh, openssl won't build here because it can't find cc[85] compiler. [Name Entwickler:in], can you make it use gcc[86] instead? At the moment I'm trying with a symlink gcc -> cc
>
> Regards, [Name PM]." (Dokument 48)

Durch den:die PM wird ein weiteres Problem beim Kompilieren berichtet, welches aufgrund der Softwareausstattung bei Entwickler:in 1 nicht auftrat. Über einen Umweg konnte er:sie das Problem zwar beheben, wünscht sich aber einen anderen Lösungsweg. Er:sie bittet Entwickler:in 1, stattdessen diesen anderen Lösungsweg umzusetzen.

8. Zweite Problemmeldung beim Kompilieren durch die:den PM: circa eine Minute später

> „Apptly it requires Perl[87] too...
>
> make[3]: Entering directory '/c/qBT-W32/src/openssl-0.9.8g/crypto/objects' /usr/bin/perl obj_dat.pl obj_mac.h obj_dat.h make[3]: /usr/bin/perl: Command not found make[3]: *** [obj_dat.h] Error 127" (Dokument 49; Ersetzung: K.M.)

Gewissermaßen als Fortsetzung der vorherigen E-Mail berichtet der:die PM von einer weiteren Fehlermeldung beim Kompilieren. Das Fehlen des gcc und des Perl-Compilers verhindert die korrekte Übersetzung des Quellcodes in ausführbaren Maschinencode.

9. Dritte Problemmeldung beim Kompilieren durch den:die PM: circa 14 Minuten später

> „even with Perl installed, and a symlink for cc.exe, it won't build here:
>
> /usr/bin/ranlib ../libcrypto.a || echo Never mind. /bin/sh.exe: /usr/bin/ranlib: No such file or directory
> [...]
> [weitere Fehlermeldungen folgen]
> make: *** [top] Error 2 Building libcrypto.a failed
>
> [Name PM]." (Dokument 50)

[85] Die Abkürzung 'cc' steht für C Compiler, eine Software, die Quellcode der Programmiersprache *C* in Maschinencode umwandelt.
[86] Der 'gcc' ist ein spezifischer C-Compiler von GNU.
[87] Bei Perl handelt es sich um eine Programmiersprache.

Trotz Installation des Perl-Compilers und der unerwünschten Nutzung des C-Compilers gelingt es der:dem PM nicht, die Software zu kompilieren. Er:sie schickt den anderen Entwickler:innen zum besseren Verständnis die Fehlermeldungen.

10. Problemlösung durch die:den PM: circa sieben Minuten später

> „Apptly, installing perl to usr/ instead of local/ fixes the last problem. I updated the wiki.
>
> For now, the only workaround needed is for cc.exe.
>
> [Name des PM].“ (Dokument 51)

Der:die PM hat die Ursachen für den scheinbar fehlenden Perl-Compiler gefunden und erklärt die Ursache. Auch aktualisiert sie:er die öffentliche Dokumentation, damit dieser Fehler auch für andere Nutzer:innen gelöst werden kann, die nicht Teil der Entwicklungsgemeinschaft sind. Das Problem mit dem unerwünschten C-Compiler besteht aber weiterhin.

11. Ankündigung durch den:die PM: circa acht Stunden später

> „Following exactly the steps in the wiki, and using the latest automaton.sh, I got qBittorrent to compile on Windows. I will start bug squashing soon :)
>
> [Name des PM].“ (Dokument 52)

Die Probleme mit dem Skript scheinen gelöst zu sein. Unklar bleibt, durch wen die letzten Fehler korrigiert wurden. Der:die PM kündigt an, dass er:sie mit dem Skript erfolgreich den Maschinencode kompilieren konnte und jetzt mit dem Test der Softwareversion für Windows beginnt.

Zusammenfassung

Es fällt auf, dass die Mehrheit der E-Mails durch die Person des:der PM verfasst wurden und ansonsten nur ein:e weitere:r Entwickler:in in die Diskussion involviert ist. Die Vermutung liegt nahe, dass Entwickler:in 1 die Person in der Entwicklungsgemeinschaft ist, der diesbezüglich die meiste Kompetenz bei der Problemlösung zugeschrieben wird beziehungsweise die Verantwortung für diesen Aufgabenbereich übernommen hat. In der Problemmeldung bezieht sich der:die PM direkt auf vorherige Beiträge zum Projektfortschritt von Entwickler:in 1. Das ist ein Hinweis auf die segmentierte Struktur bei diesem FLOSS-Projekt. Einzelne Entwickler:innen kümmern sich um spezifische Aufgabengebiete, in denen sie im Vergleich zu den anderen Entwickler:innen eine besondere Expertise aufbauen oder sich schon vor der FLOSS-Entwicklung angeeignet haben. Die Struktur von abgetrennten Aufgabengebieten mit jeweiligen Verantwortlichen deutet eher auf Kooperation statt auf Kollaboration hin.

Beim Kommunikationsstil des:der PM ist anzunehmen, dass die Antworten nicht nur an Entwickler:in 1 adressiert sind, sondern auch eine weitere Funktion haben: Die Mailingliste dient auch als Informationsarchiv für das FLOSS-Projekt selbst. So können frühere Entscheidungen und der Umgang mit Problemen auch durch zukünftige Entwickler:innen anhand des Archivs der Mailingliste nachvollzogen werden. Die öffentlichen Mailinglisten sind damit nicht nur Werkzeug zur Problemlösung, sondern dokumentieren auch Regelsysteme, die innerhalb der Gruppe über die Zeit hinweg gewachsen sind. Sie bilden somit einen Orientierungsrahmen für neue Mitglieder der Gemeinschaft, wie die Entwicklungsarbeit in der Praxisgemeinschaft funktioniert. Neben der FLOSS selbst sind auch das projekteigene Wiki und die Inhalte auf der Mailingliste Teil des gemeinsamen Repertoires, welches sich stetig weiterentwickelt.

Problemlösung auf der Trans-ML

Auf der Mailingliste der Übersetzer:innen werden hauptsächlich Fragen zum Hintergrund der zu übersetzenden Wörter diskutiert. Dies ist nicht nur für die Kontextabhängigkeit bei der Übersetzung wichtig, sondern führt auch zu Änderungen in der englischen Leitübersetzung. Diese dient allen Übersetzer:innen als Vorlage, an der sich die Übersetzungen in die jeweiligen Sprachen orientieren. In der Regel stellen die Übersetzer:innen eine Frage zu einem Wort oder einer Wortgruppe und der:die PM erklärt die Aktion, zum Beispiel die Bezeichnung eines Menüpunkts, die durch diese Funktion in der Software ausgelöst wird.

Diskutiert werden aber auch die für die Übersetzung genutzten Werkzeuge und Prozessabläufe. Die folgende Diskussion auf der Mailingliste behandelt einen Vorschlag für eine neue Übersetzungsstruktur, die die Abläufe vereinfachen soll.

1. Vorschlag vom:von Übersetzer:in 1 an die:den PM: 04.07.2013 / 09:29 Uhr

> „Hi
> Can you consider using transifex[88] for translation files?
>
> This way we can make an team an you don´t need to send files. Just a head notice and here we go to transifex, make translations and you can then, before release, export them from tx[89] and merge it with code.
>
> I´m working in there for qupzilla, razor-qt, smplayer and it´s working fine.
> Regards

[88] *Transifex* ist eine proprietäre, web-basierte Übersetzungsplattform. Die Dateien für die Übersetzung werden hochgeladen und die Übersetzer:innen können sich diese wiederum herunterladen, offline übersetzen und wieder per Web-Oberfläche hochladen oder stattdessen einen Online-Editor nutzen. Die:der PM kann wiederum automatisiert die übersetzen Dateien von Transifex herunterladen und sie in die Versionsverwaltung des Softwareprojekts einpflegen. Das manuelle Integrieren von Dateien einzelner Übersetzer:innen entfällt dadurch.

[89] Das Abkürzung 'tx' steht für Transifex.

2013/7/4 [E-Mail-Adresse PM]
> Hello to all the translators,
> I have updated the ts files for qbittorrent and there ~4 new strings for
> translation for each branch. Please translate them and send them to me to
> [E-Mail-Adresse vom PM] Also give priority to v3_0_x branch
> because I want to do a 3.0.10 release as soon as possible.
> Here is master: > http://builds.shiki.hu/temp/translations/qbittorrent_lang_4-July-2013%5bmaster%5d.7z
> Here is v3_0_x: > http://builds.shiki.hu/temp/translations/qbittorrent_lang_4-July-2013%5bv3_0_x%5d.7z > > Thank you. " (Dokument 397)

Die Übersetzung erfolgte bislang in einem manuellen Modus. Durch die:den PM wurde auf einem Server eine Datei mit allen zu übersetzenden Begriffen bereitgestellt. Jede:r Übersetzer:in hat sich diese Datei auf den eigenen Rechner geladen, sie übersetzt und sie per E-Mail wieder an die:den PM geschickt. Diese:r hat sie dann in das Hauptverzeichnis der kommenden Version kopiert, damit sie bei der nächsten Veröffentlichung Teil der neuen Version ist.

Übersetzer:in 1 kennt durch das Engagement in anderen FLOSS-Projekten eine zentrale Verwaltungssoftware zur Übersetzung, die auch für qBittorrent eine Vereinfachung der Arbeitsabläufe darstellen kann und schlägt diese dem:der PM als neues Übersetzungswerkzeug vor.

2. Zustimmung von:vom Übersetzer:in 2: circa drei Stunden später

„I liked this idea.

:D
[...][Zitat von Übersetzer:in 1 folgt]" (Dokument 398)

Der Vorschlag von Übersetzer:in 1 findet auch bei Übersetzer:in 2 Zustimmung.

3. Antwort durch den:die PM: circa sieben Stunden später

„[Zitat vom Vorschlag von Übersetzer:in 1]
> Can you consider using transifex for translation files?

Yes it sounds interesting. I'll to investigate it next week.
If you don't see any progress on this for ~3 weeks,
feel free to remind it to me." (Dokument 399)

Der:die PM nimmt den Vorschlag auf und erwünscht sich Bedenkzeit, um sich selbst ein Bild von der vorgeschlagenen Software für die Verwaltung der Übersetzungen zu machen.

4. Nachfrage vom:von Übersetzer:in 3: 17 Minuten später

„Is there an option to translate without any special program? e.g. a simple text file. This way I can translate on the go on my smartphone, tablet etc.“ (Dokument 400)

Eine weitere Person aus dem Kreis der Übersetzer:innen schaltet sich in die Diskussion ein, jedoch mit einem neuen Wunsch. Sie:er möchte die Übersetzung ohne Nutzung einer speziellen Software durchführen, damit dies plattformunabhängig auf verschiedenen Endgeräten möglich ist. Es wurden jetzt verschiedene Wünsche formuliert, wie die Praxis der Übersetzungsarbeit geändert werden soll.

5. Antwort des:der PM auf die Nachfrage: sieben Minuten später

„The .ts[90] files are basically XML[91] files. You can open them in any text editor. An editor with XML highlighting would be more useful though.“ (Dokument 401)

Der:die PM führt aus, dass die Dateien mit jedem Texteditor bearbeitet werden können. Dies ist allerdings etwas umständlich, weshalb zu speziellen Editoren geraten wird, die bestimmte Inhalte der zu übersetzenden Dateien hervorheben können. Dadurch ist die Datei im Texteditor übersichtlicher dargestellt, was wiederum die Arbeit erleichtert. Obwohl dieser Wunsch von Übersetzer:in 3 in dieser Diskussion formuliert wird, steht er nicht in konkreten Bezug zum Vorschlag von Übersetzer:in 1 zum Umstieg eine andere Übersetzungspraxis.

6. Reaktion von:vom Übersetzer:in 1 auf die Antwort vom:von PM: circa 17 Stunden später

„[Zitat vom Vorschlag Übersetzer:in 1]
>>>Can you consider using transifex for translation files?
[Zitat der Antwort PM]
>> Yes it sounds interesting. I'll to investigate it next week. If you don't
>> see any progress on this for ~3 weeks, feel free to remind it to me.

Ok. I will remind you then.
Regards“ (Dokument 402)

Übersetzer:in 1 kündigt an, den:die PM nach Ablauf von drei Wochen zu erinnern.

7. Vorschlagsumsetzung durch den:die PM: circa drei Wochen später

[90] Die '.ts'-Dateien enthalten die Wörter und Wortgruppen für qBittorrent mit der Übersetzung für die jeweilige Sprache.

[91] Das Akronym XML steht für *Extensible Markup Language*, eine Auszeichnungssprache ähnlich zu HTML, die aber für die Darstellung hierarchisch strukturierter Daten in einer Textdatei optimiert ist.

Die nächste E-Mail zu diesem Thema informiert über den Stand der neuen Software für die Übersetzung. Zwischenzeitlich wurden diverse Diskussionen zu anderen Themen über die Trans-ML geführt.

> [...]
> I also have set up a project in Transifex. Check it out. https://www.transifex.com/-projects/-p/qbittorrent/
> I would prefer the translations to be done in Transifex because I intend to fetch them automatically from there before each release. However, I will still accept translations the old way. I will still offer translations in the mailist list, but Transifex is set to automatically update the ts files when there are new strings available.“ (Dokument 413)

Der:die PM hat, wie dies angekündigt wurde, nach Ablauf der drei Wochen reagiert und den Vorschlag von Übersetzer:in 1 umgesetzt. Die neue Software für die Verwaltung der Übersetzungen wurde eingerichtet. Sie soll der neue Standard für die Übersetzungsarbeit in diesem FLOSS-Projekt sein. Nichtsdestotrotz bietet der:die PM weiterhin die Möglichkeit an, Übersetzungen per E-Mail an ihn:sie zu schicken.

Die Übersetzungspraxis für die Entwicklungsgemeinschaft wurde damit aufgrund von zwei Übersetzer:innen geändert. Die ursprünglich von Übersetzer:in 1 genannten Argumente wurden durch den:die PM übernommen.

8. Antwort von:von Übersetzer:in 1 auf die Vorschlagsumsetzung: circa 14 Stunden später

> „[Zitat der Vorschlagsumsetzung vom PM]
> > Here are the .ts’ with 2 new strings from git master:
> > http://builds.shiki.hu/temp/translations/master-lang_28072013.7z
> >
> > I also have set up a project in Transifex. Check it out.
> > https://www.transifex.com/projects/p/qbittorrent/
>
> This is great news. I´ve already done my join team for Portuguese.
> Regards“ (Dokument 414)

Übersetzer:in 1 freut sich über die Umsetzung des Vorschlags und verkündet, in Transifex ein Team für die portugiesische Übersetzung eingerichtet zu haben.

9. Koordinierung der neuen Übersetzungsverwaltung durch die:den PM: circa eine Stunde später

> „@[Name Übersetzer:in 4] I haven’t received a join request from you in Transifex for the Italian lang. Please check again.

@[Name Übersetzer:in 5] I have made you coordinator so you can accept/deny new translators as you see fit.

If anyone else wants to become coordinator in their respective locale pls msg[92] me. Otherwise I will approve any translator request.

2013/7/29 [E-Mail Adresse Übersetzer:in 5]@...>

> Hi [Name PM],
>
> could you please make me the coordinator for Slovak? I'm the original
> author of this translation and I've been updating it regularly. I
> don't want other people to join and change things in my translation.
> ur team had bad experience with free-for-all translations in the past
> (especially on the Launchpad platform) and there was no way for us to
> guarantee translation quality until we enforced team membership.
> Thanks.
>
> Regards, [Name Übersetzer:in 5]" (Dokument 415)

In dieser E-Mail reagiert der:die PM auf Anfragen von Übersetzer:innen die nicht über die öffentliche Mailingliste geschickt wurden. Die Antwort an Übersetzer:in 4 lässt auf Probleme bei der Koordinierung der Verantwortlichkeiten in der Übersetzungsverwaltung schließen. Da diese Problemschilderung jedoch über einen anderen Kommunikationskanal erfolgte, ist der Kontext nicht nachvollziehbar.

Bei der Antwort auf das Anliegen seitens Übersetzer:in 5 ist jedoch die nichtöffentliche E-Mail als Zitat in der E-Mail des:der PM enthalten. Die Person berichtet von Problemen bei der Zusammenarbeit mit anderen Übersetzer:innen für die slowakische Sprachunterstützung. Ob sich diese auf dieses FLOSS-Projekt beziehen oder von anderen die Rede ist, bleibt unklar. Der aktuelle Modus sieht vor, dass alle interessierten Personen Zugriff auf die Übersetzungsdateien haben und Änderungen vornehmen können. Übersetzer:in 5 sieht dies als Problem, weil in der Vergangenheit Übersetzungen der betreffenden Personen geändert und dies als Qualitätsverlust gesehen wurde. Die Person wünscht sich eine Zugangsbeschränkung für neue Übersetzer:innen, die an eine Art Projektmitgliedschaft gekoppelt wird. Wie diese genau aussieht, wird nicht konkret ausgeführt.

In Transifex können Koordinator:innen für die jeweiligen Sprachen bestimmt werden, die wiederum entscheiden können, wer Übersetzungen beisteuern kann. Übersetzer:in 5 hat sich gewünscht, diese Rolle zugesprochen zu bekommen. Dies wird damit begründet, dass die erste slowakische Übersetzung durch diese Person erfolgt ist und sie dann regelmäßig aktualisiert wurde. Dieser Wunsch wurde durch den:die PM erfüllt. So kann

[92] Dieser Internet-Jargon werden bei Floskeln häufig die Vokale weggelassen. 'pls msg me' bedeutet 'please message me'.

Übersetzer:in 5 selbst entscheiden, welche anderen Übersetzer:innen Veränderungen an der slowakischen Sprachversion vornehmen können.

10. Ankündigung über Änderung der Beteiligungsmöglichkeiten durch den:die PM: 90 Minuten später

> „I have made the translators that I already know from before, coordinators to their respective locales. This of course, applies only to those that have made a join request already. I'll the same for the others, when I receive such a request." (Dokument 416)

Der:die PM kündigt an, dass er alle ihm aus der Vergangenheit bekannten Übersetzer:innen zu Koordinator:innen der jeweiligen Sprachen gemacht hat. Damit hat er:sie die Bitte von Übersetzer:in 5 auf alle anderen Übersetzer:innen für das FLOSS-Projekt übertragen. Es folgen noch weitere E-Mails zu dem Thema. Diese enthalten jedoch nur Ankündigungen für neue Übersetzungsteams und weitere Tipps für den Umgang mit Transifex.

Zusammenfassung

Der Diskussionsverlauf beschreibt eine tiefgreifende Änderung der Prozesse bei der Übersetzung. Sie wurde durch eine:n Übersetzer:in angestoßen und argumentativ begründet. Diese Änderung wurde durch eine:n weitere:n Übersetzer:in bestärkt und führte in der Konsequenz zu einer Entscheidung, die zu einer Änderung der gemeinschaftlichen Praktiken führte. Es wurde aber nicht nur der Arbeitsprozess durch die Einführung einer zentralen Übersetzungsverwaltung verändert, sondern auch eine neue Hierarchie innerhalb der Entwicklungsgemeinschaft eingeführt.

Zuvor gab es keine festen Mitglieder, die sich erst nach einer Art Initiation an der Entwicklungsarbeit beteiligen konnten. Jede Person, die eine Übersetzung einreichte, war Teil der Entwicklungsgemeinschaft. Dieses offene System wurde durch ein hierarchisches System ersetzt. Übersetzer:innen, die in der Vergangenheit Übersetzungen für das FLOSS-Projekt beigetragen haben, wurden durch die Rolle als Koordinator:in in die Lage versetzt, Beiträge neuer Übersetzer:in zu akzeptieren oder abzulehnen. Diese grundlegende Einschränkung in den Beteiligungsmöglichkeiten wurde durch den:die PM aufgrund des Wunsches einer:eines Übersetzer:in eingeführt und stieß zumindest auf der öffentlichen Trans-ML nicht auf Kritik. Damit wurde die Hierarchie weiter differenziert. Wo es vorher nur die PM und Übersetzer:innen gab, gibt es nun eine Zwischenstufe in Form von Koordinator:innen für einzelne Sprachen, die in Bezug auf das Schalenmodell den Kernentwickler:innen entsprechen. Ein neues Werkzeug hat damit nicht nur die gemeinschaftlichen Praktiken verändert, sondern auch die Strukturen und die Aufgabenverteilung der Entwicklungsgemeinschaft weiter ausdifferenziert.

Auffällig an dieser Diskussion ist auch der prägnante Schreibstil des:der PM. Es handelt sich nicht um den:die gleiche:n PM, wie bei bei der zuvor beschriebenen Diskussion auf der Dev-ML. Die Schreibstile der beiden PM unterscheiden sich deutlich voneinander.

In diesem Beispiel auf der Trans-ML sind bei dem:der neuen PM fast keine Begrüßungs- und Abschiedsformeln zu finden und auch sonst sind die Ausführungen der Person eher kurz und prägnant gehalten. Ebenso nutzt diese:r PM weniger Formulierungen der Anerkennung für die Arbeit der Übersetzer:innen, als der:die vorherige PM.

6.5.6 Zusammenfassung: Beitragsanalyse der Mailinglisten

Die qualitative und quantitative Analyse der Beiträge auf den beiden öffentlichen Mailinglisten dieses FLOSS-Projektes bestätigt bekannte Studien und ermöglicht zugleich neue Erkenntnisse. Ähnlich wie bei Hannemann et al. (2012) zeigt sich auch hier, dass der Großteil der Beiträge auf den Mailinglisten nur von wenigen Entwickler:innen stammt. Die kommunikativ stärksten Mitglieder der Entwicklungsgemeinschaft sind die beiden PM. Sie starten viele Diskussionen und beteiligen sich auch am häufigsten bei Diskussionen. Zusammenfassend ist festzustellen, dass es sich um PM-zentrierte Diskussionen handelt. Es wurden nur wenige Diskussionen gefunden, die ausschließlich zwischen Entwickler:innen oder Übersetzer:innen geführt wurden.

Obwohl die beiden öffentlichen Mailinglisten die zentralen Orte für Diskussionsprozesse sind, kamen im Laufe der Projektentwicklung weitere öffentliche Kommunikationskanäle hinzu: Foren, Chats, Bug-Tracker, Wiki und der Wechsel zu einem weiteren FLOSS-Portal. Diese verweisen auf die Veränderungsprozesse, welche die Form des gegenseitigen Austauschs, das geteilte Repertoire und die Strukturen der Praxisgemeinschaft beeinflussten. Neben öffentlichen Kommunikationskanälen gab es wiederholt Belege für direkte E-Mail-Kommunikation zwischen einzelnen Mitgliedern der Entwicklungsgemeinschaft. Dies belegt, dass es neben der öffentlichen Kommunikation auch private Kommunikation zwischen den Mitgliedern gab. Damit wurde ein Teil der Entscheidungsfindung im Projekt der restlichen Projektgemeinschaft entzogen. Dies zeigt die Komplexität der sozialen Beziehungen und informellen Hierarchien in einem FLOSS-Projekt.

Thematisch dienen die Mailinglisten vor allem als ein problemorientiertes Kommunikationswerkzeug. Es wurde hauptsächlich genutzt, um Release-Ankündigungen, Handlungsangebote, Handlungsaufforderungen und Problemschilderungen zu kommunizieren. Durch die Persistenz als öffentlich archiviertes Medium dienen die Mailinglisten gleichzeitig auch der Dokumentation von Praktiken und als Wissensspeicher. Es werden nicht nur Probleme benannt und Zuständigkeiten verteilt, sondern auch die Lösungswege werden dokumentiert. Wichtige Wissensbestände werden an anderen Stellen (zum Beispiel im Wiki) zusätzlich dokumentiert und bilden damit ein Konzentrat des gemeinsamen Wissens, welches erst aufgrund der Austauschprozesse innerhalb der Entwicklungsgemeinschaft entstehen konnte. Dies entspricht der Theorie des Aufbaus von handlungsorientierten Wissens, wie es durch den Konnektivismus von Siemens (2004) postuliert wird. Es liegt nicht nur in Personen verkörpert vor, sondern auch durch deren Vernetzung untereinander und gespeichert in Form von Datenbanken und Organisationen.

Bei der genauen Betrachtung der Diskussionskultur fallen vor allem zwei Aspekte auf:

Zum einen sind die Diskussionen allgemein recht kurz. Auch tiefgreifende Veränderungen innerhalb des Projektes lösen keine langwierigen Diskussionen mit vielen Beteiligten aus. Dies könnte ein Indiz für die von Taubert (2006, S. 163f) erwähnte Aversion von FLOSS-Projekten sein, dass durch lange Aushandlungsprozesse die Entwicklungsarbeit stagnieren könnte. Zum anderen konnte gezeigt werden, dass dem Ausdruck von Anerkennung als soziale Praxis eine große Bedeutung in der Kommunikation beigemessen wird. Krogh et al. (vgl. 2012, S. 669) vermuteten, dass sich das individuelle Interesse von Entwickler:innen nach langjährigem Engagement im zeitlichen Vergleich zum Beginn ihres FLOSS-Engagements ändert. Diesbezügliche Forschungsergebnisse fehlen aber bislang. Die in dieser Arbeit beobachtete Anerkennungskultur bei der Kommunikation stärkt die These, dass die soziale Praxis in einer Gemeinschaft selbst ein Beweggrund für das fortgesetzte Engagement für Entwickler:innen darstellen kann. Aus diesem Grund erfüllt ein respektvoller Umgang untereinander weitere Funktionen, als nur die verkürzte Betrachtung der Entwicklungsgemeinschaft als Wissensressource.

6.6 Individuelles Engagement in der Projektgemeinschaft

Im folgenden Abschnitt erfolgt eine personenzentrierte Darstellung des Engagements innerhalb der Projektgemeinschaft. Die Beschreibung erfolgt entlang der beiden Dimensionen Kommunikationsverhalten und personenspezifische Anteile von Beiträgen zum Projektfortschritt auf der Versionsverwaltung.

Wie schon bereits in Abschnitt 6.5.1 erwähnt, bewegt sich die Zahl der an die Mailinglisten gesendeten E-Mails bei der Mehrheit der Entwickler:innen und Übersetzer:innen im einstelligen Bereich. Nur circa 8% der am Projekt beteiligten Personen haben mindestens fünf Beiträge über die Mailinglisten verschickt. Bei 3,5% der Mitglieder waren es mehr als zehn Beiträge. Der Rest der Entwicklungsgemeinschaft verfasste keine bis vier Nachrichten im Erhebungszeitraum.

Im Folgenden wird das Engagement von zwei Personen untersucht, die mindestens zehn E-Mail-Beiträge verfasst haben. Ziel ist die Beschreibung langfristigen Engagements in der Projektgemeinschaft anhand ausgesuchter Mitglieder der Projektgemeinschaft im zeitlichen Verlauf. Es wird gezeigt, mit welchen Themen sich die Personen beschäftigen und wie sich dies in der Kommunikation auf den beiden Mailinglisten niederschlägt. Bei beiden Personen handelt es sich um Entwickler:innen. Die erste Person ist bereits in der Anfangsphase des Projektes dazugestoßen und war Teil des Kernteams. Die zweite Person folgte erst später und ist ein Beispiel dafür, wie Entwickler:innen von der Peripherie bis hin zum Zentrum der Gemeinschaft vorrücken und zentrale Positionen übernehmen. Mit der Auswahl dieser Personen soll die Heterogenität in der Entwicklungsgemeinschaft veranschaulicht werden.

6.6.1 Dominique

Die Abbildung 6 zeigt das Engagement von Dominique im zeitlichen Verlauf. Die Person war bereits beim Beginn des Erhebungszeitraums Teil des Kernteams. Die ersten Interaktionen begannen im November 2007. Ab Mai 2011 konnte keine weiteren Interaktionen mehr registriert werden.

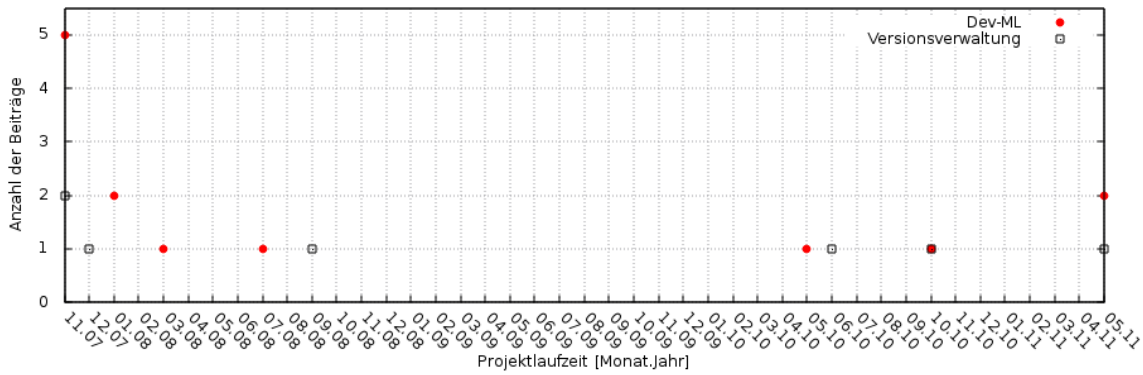

Abbildung 6: Engagement Dominique

Das Engagement von Dominique erstreckte sich auf Aktivitäten der Dev-ML und der Versionsverwaltung. Die erste E-Mail auf der Dev-ML deutet an, dass Dominique schon vorher Kontakte zur Entwicklungsgemeinschaft hatte. Es gab schon einen kleinen Kreis von Entwickler:innen, die vor der Eröffnung der Mailinglisten an dem Projekt gearbeitet hatten. Dominique ist die zweite Person, die durch den:die PM Schreibrechte für die Versionsverwaltung bekommen hat. In den 14 vorangegangenen Monaten führten nur der:die PM und ein:e weitere:r Entwickler:in Änderungen an der Versionsverwaltung durch. Bis zur Erteilung der Schreibrechte hat der:die PM die Beiträge von Dominique zum Projektfortschritt in die Versionsverwaltung eingepflegt.

Von der zeitlichen Verteilung her fokussiert sich das Engagement auf zwei Phasen. Die erste Phase erstreckt sich über elf Monate, beginnend kurz nach Eröffnung der Dev-ML. In dieser Zeit hatte die Dev-ML zwischen fünf und sieben Mitglieder. Nach einer Pause von knapp anderthalb Jahren erfolgte die zweite Aktivitätsphase. Diese dauerte 13 Monate an und war von zwei mehrmonatigen Pausen geprägt. In dieser Zeit hat sich die Zahl der aktiven Mitglieder auf der Dev-ML mutmaßlich verdoppelt. Die Kommunikation im Erhebungszeitraum wird im Folgenden auf ihre Hauptinhalte verkürzt dargestellt. Eine Zeile in der Tabelle kann mitunter mehr als zwei E-Mails repräsentieren. Die Spalte mit dem Verlauf der Kommunikation signalisiert, welche Seite (Dominique oder eine andere Person der Entwicklungsgemeinschaft) den Anfang des jeweiligen Kommunikationsstrangs gesetzt hat (Startbeitrag -> Antwort). Dies soll verdeutlichen, wer Kommunikation initiiert und wer darauf reagiert.

Entwicklungsgemeinschaft	Verlauf	*Dominique*
PM: Kritik am dem von Dominique eingereichten Programmcode	->	Rechtfertigung des gewählten Vorgehens, Bitte um Hilfe
PM: Schreibrecht bewilligt	<-	Wunsch nach Schreibrecht für Versionsverwaltung
Angebot durch PM angenommen, Vorschläge bewertet	<-	Angebot, einen Teilaufgabe zu übernehmen, Formulierung eigener Umsetzungsvorschläge
PM: allgemeine Aufforderung zum Test von Programm-funktionen	->	Mithilfe beim Testen, Hinweis auf fehlende Funktionalität
PM: Bitte, vom:von PM verfassten Code zu bewerten	->	Code-Evaluation
[keine öffentliche Reaktion]	<-	Fehlermeldung mit Lösungsvorschlag, Bitte an den:die PM, diesen zu korrigieren
andere:r Entwickler:in formuliert Anforderungen	<-	Angebot einen Teilbereich zu übernehmen, Bitte um Anforderungen an Teilbereich
[keine öffentliche Reaktion]	<-	Handlungsbericht, Umsetzungsproblem mit Lösungsvorschlag an PM
neues Mitglied: Anfrage zum Mitmachen	->	Begrüßung des neuen Mitglieds, Hinweis auf Partizipationsmöglichkeiten
PM: allgemeiner Handlungsbericht, Informationen zum Projektfortschritt	->	Hinweis zur Aktualisierung der Programmdokumentation
[nicht öffentliche Mail] Bewertung des eingereichten Programmcode	->	Einreichung des aktualisierten Programmcodes, Bitte um Ablösung durch andere Person in diesem Teilbereich
PM: Bewertung des eingereichten Programmcodes, Übernahme vom Teilbereich	->	Einreichung des aktualisierten Programmcodes

Tabelle 4: Kommunikation zwischen Dominique und der Entwicklungsgemeinschaft

Auffällig bleibt bei der Tabelle, dass die Kommunikation zwischen Dominique und der Projektgemeinschaft auf die Person des:der PM fokussiert ist. Ein argumentativer Austausch mit anderen Mitgliedern der Gemeinschaft fand kaum statt. Diese Art der PM-zentrierten Kommunikation war bereits bei den Beispielen von den Problemlösungen auf den Mailinglisten zu finden. Der paraphrasierte Diskussionsverlauf zeigt, dass die Kommunikationsinhalte von Dominique auf folgende Bereiche konzentriert sind:

- qualitative Rückmeldungen zum Programmcode
- Aufgabenverteilung
- Handlungsberichte
- Problemberichte und Lösungen
- Integration von neuen Mitgliedern

Über die Gründe, warum sich Dominique bei dem FLOSS-Projekt engagiert, ist nichts bekannt. Aus den E-Mails geht allerdings hervor, dass er selber Studierender eines IT-Studiengangs ist. Im Rahmen eines Seminars zur Softwareentwicklung war Dominique maßgeblich an der Entwicklung einer webbasierten Benutzer:innenschnittstelle für dieses FLOSS-Projekt beteiligt. Dieses Modul der Software programmierte er zusammen mit anderen Studierenden seines Seminars als selbstgewählte Studienaufgabe.

Die Diskussionsthemen zeigen den problemorientierten Fokus auf der Dev-ML. Diskussionen über private Themen finden nicht statt. Informationen über die Lebenssituation von Dominique fanden sich in nur drei beiläufigen Bemerkungen, die allesamt Begründungen dafür waren, warum Beiträge zum Projektfortschritt verspätet eingereicht wurden.

6.6.2 Marlin

Bei Marlin handelt es sich ebenfalls um eine:n Entwickler:in. Marlin vollzog im Verlauf der Zeit einen Rollenwechsel von der Rolle als periphere:r Entwickler:in hin zum Zentrum der Entwicklungsgemeinschaft. Nach circa zwei Jahren kontinuierlicher Beteiligung am Projektfortschritt wurde die Funktion der Projektleitung an Marlin übergeben.

Bis zur Rollenübernahme als PM ist Marlin auf keiner der beiden Mailinglisten in Erscheinung getreten. Marlin hat als vierte:r Entwickler:in Schreibrechte für die Versionsverwaltung bekommen. Zu diesem Zeitpunkt existierte das FLOSS-Projekt schon vier Jahre.[93] Nach den ersten Projektbeiträgen über drei Monate hinweg folgte eine Aktivitätspause von etwa zehn Monaten. Ab September 2012 erfolgte eine kontinuierliche Mitarbeit mit mehreren Projektbeiträgen pro Monat. In dieser Zeit gab es insgesamt nur drei Personen, die regelmäßig die Projektentwicklung vorantrieben.

Rollenwechsel: von der Peripherie ins Zentrum

In dem Zeitraum, in welchem Marlins Engagement begann, kam es zu einer verstärkten Beteiligung neuer Projektmitglieder. Deren Engagement beschränkte sich aber oftmals auf nur einen einzigen Projektbeitrag. Der Wechsel der:des PM wurde der Projektgemeinschaft in einer E-Mail an beide Mailinglisten durch Marlin selbst im Juni 2013 bekanntgegeben. Eine Ankündigung über den Ausstieg durch den:die erste:n PM erfolgte nicht.

[93] Die Trans-ML war der erste gemeinschaftliche Entwicklungswerkzeug des FLOSS-Projektes. Sie wurde ab Mai 2006 genutzt. Vier Monate später folgte dann die Versionsverwaltung Subversion. Die Dev-ML kam erst im Oktober 2007 dazu.

„Hello to all the commumity,

From some time yesterday [Name PM] transfered the project maintenance over to me. He is really busy with Real Life and cannot review patches and actively support further development in a timely fashion. I am sure you all have noticed the slowdown for the recent months in releases and patch merging. He however didn't quit entirely. He will still review patches when he can, contribute code, offer guidance if needed etc. But not in an active matter. He will also continue to provide OSX builds, unless someone else wants to do it.

As the new maintainer I can do whatever I want, but that doesn't mean that I will. Now patches will be reviewed/merged a lot faster. Releases will happen frequently and maybe every 2 weeks(depending on development traffic). Also I get to decide for features to be implemented and which maybe were discussed here by other forum members. Like new UI icons.

To all [b]developers[/b][94] out there: I will be happy to merge your fixes / patches / features / etc. Just hop over to github and send me pull requests. To all other [b]knowledgeable qbittorrent[/b] people: You are welcome to enrich the wiki, help triage bugs in the bug tracker and help other users in the forums. To the [b]rest of the community[/b]: qBittorrent is alive and kicking. qBittorrent is awesome. Spread the word and come join us in the forum." (Dokument 248)

Diese E-Mail ist in mehrerlei Hinsicht bemerkenswert. Zum einen ist sie ein Zeugnis vom Selbstverständnis der Rolle als Projektmaintainer:in. Zum anderen ist sie auch ein Weckruf an die Projektgemeinschaft. Gegenstand der E-Mail ist die Ankündigung, dass Marlin jetzt die Rolle des:der PM innehat. Auf den öffentlichen Mailinglisten gab es zuvor keine Informationen über den anstehenden Wechsel beziehungsweise über dessen Ursachen. Es liegen keine Informationen darüber vor, ob dies vorher innerhalb der Projektgemeinschaft auf anderen Kommunikationskanälen diskutiert wurde.

Dem Selbstverständnis von Marlin als neue:n PM ist zu entnehmen, dass er:sie für sich umfassende Entscheidungsbefugnisse über die zukünftige Entwicklung beansprucht. Im Gegenzug verspricht er:sie neue Projektbeiträge von der Gemeinschaft schneller zu bearbeiten und in regelmäßigen Veröffentlichungen neuer Versionen zu bündeln. Dieses Führungsprinzip eines *wohlwollenden Diktators*, die sich aus erfahrenen Projektmitgliedern rekrutieren, ist auch in vielen anderen FLOSS-Projekten zu finden. Bekannt ist die Rolle von Linus Torvalds im Linux-Kernel. Diese Art der Entscheidungsstruktur ist häufig bei kleineren Projekten zu finden, bei denen der:die Gründer:in auch die Rolle der:des PM ausfüllt. Die Diktatur ist aber an das Wohlwollen gekoppelt, da es keine Sanktionsmöglichkeiten gibt. Die Entwickler:innen und Übersetzer:innen beteiligen sich freiwillig

[94] Die Zusätze [b] und [/b] sind Teil vom HTML-Code, in dem diese E-Mail verfasst wurde und bedeuten lediglich, dass das Wort 'developers' in fettgeschriebenen Buchstaben dargestellt werden soll.

und können jederzeit ihr Engagement einstellen. Die Person, die die Rolle des:der PM ausübt, sollte sich also in wichtigen Kernfragen mit den Entwickler:innen abstimmen, die sich um Subsysteme des Projektes kümmern (vgl. Raymond 2000, S. 27f.). Bei größeren Softwareprojekten zeigen sich aber auch andere Entscheidungsstrukturen. Manche wählen ihre PM in einem demokratischen Wahlverfahren oder sie entscheiden meritokratisch, wobei jene Entwickler:innen das letzte Wort, welche die größte Reputation im Projekt verfügen (vgl. Brand 2009, S. 90).

Mit dem Rollenwechsel ändert sich auch Marlins Kommunikationsverhalten schlagartig. Zu den neuen kommunikativen Aufgaben gehört es jetzt, regelmäßig über den Projektfortschritt zu berichten und auf E-Mail-Anfragen über die Mailingliste zu reagieren. Beim Vergleich zwischen beiden Personen, die als PM in dem Projekt agierten, stechen die unterschiedlichen Kommunikationsstile hervor. Marlin nutzt die öffentlichen Mailinglisten weitaus weniger als Diskussions- und Organisationsmedium als der:die erste PM. Hier liegt wieder die Vermutung nahe, dass Teile der Kommunikation zwischen PM und Entwickler:innen auf andere Kommunikationskanäle ausgelagert wurden. Speziell die Bitte an andere Entwickler:innen, die Softwareentwicklungsplattform GitHub für die Beiträge zu nutzen, lässt die Möglichkeit zu, dass die Kommunikationskanäle von GitHub gegenüber der Mailingliste bevorzugt wurden. Diese These wird durch den Vergleich mit dem Kommunikationsverhalten auf der Trans-ML untermauert. Dort gehen die Diskussionen in dem Stil weiter, wie sie vor dem Rollenwechsel auch erfolgten. Den Übersetzer:innen stehen keine weiteren Kommunikationskanäle durch neue Werkzeuge zur Verfügung. Hier bleibt das zentrale Kommunikationsmedium die Mailingliste.

6.7 Dokumente des Kompetenzaufbaus

Mit der Analyse der E-Mail-Kommunikation soll auch die Frage beantwortet werden, was genau die Entwickler:innen und Übersetzer:innen durch ihr Engagement in diesem FLOSS-Projekt lernen. Zu diesem Zweck wurde nach Selbstaussagen gesucht, die die Mitglieder in den Mailinglisten verfasst haben.

6.7.1 Zeugnisse des Kompetenzaufbaus innerhalb der Mailinglisten

Auf Grundlage der untersuchten Mailinglisten konnte keine expliziten Aussagen gefunden werden, in denen Personen ihren individuellen Erkenntnisgewinn den anderen Projektmitgliedern mitteilen. Dennoch ist anzunehmen, dass dieser stattfand, da sonst die Einarbeitung in ein neues Projekt eine langwieriges Unterfangen ohne nennenswerten Beitrag darstellen würde.

Wie in dem Abschnitt 6.5.5 über die Problemkommunikation erläutert wurde, werden die Mailinglisten mit der Intention genutzt, Antworten auf Fragen der Entwicklungsarbeit zu bekommen. Beispielsweise durch Nachfragen bei konkreten Problemen oder durch die Bitte um Rückmeldungen durch andere Entwickler:innen bezüglich der Qualität einge-

reichter Beiträge. Fragen auf den Mailinglisten wird ein hoher Stellenwert eingeräumt. Es lassen sich kaum Fragen von Einsender:innen finden, die nicht konstruktiv beantwortet wurden. Derart konstruktive Antworten sind wichtig für die Fortsetzung der gemeinschaftlichen Entwicklungsarbeit. Nur wenn die Entwickler:innen ihre Probleme lösen können, kann die Entwicklungsarbeit am Projekt fortgesetzt werden. Konkrete Selbstzeugnisse vom Kompetenzaufbau anhand der E-Mails konnten innerhalb des vorliegenden Datenmaterials nicht gefunden werden. Es können lediglich Vermutungen angestellt werden, dass eine kontinuierliche Fortsetzung der Entwicklungsarbeit Lernprozesse voraussetzt. Denn ohne Erfolgserlebnisse beim Lösen von Problemen ist die Entwicklungsarbeit wenig sinnstiftend.

6.7.2 Selbstauskünfte über den Kompetenzaufbau

Um dennoch Informationen aus der Versionsverwaltung und der Beitragsanalyse mit dem individuellen Kompetenzaufbau der Projektmitglieder in Verbindung setzen zu können, wurden Interviews als Ergänzung des Datenmaterials gewählt.

Da die Kontaktdaten der Projektmitglieder im Datenmaterial enthalten waren, wurden Interviewanfragen per E-Mail an die Entwickler:innen gesandt. Dafür wurden die Entwickler:innen ausgewählt, die sich an Diskussionen über die Mailinglisten längerfristig beteiligt hatten. Die Interviewbereitschaft war sehr gering. Dies lag zum einen an der geringen Rücklaufquote durch veraltete Kontaktdaten und zum anderen an der geringen Bereitschaft der Entwickler:innen zur Teilnahme an einem Interview. Begründungen für die Interviewablehnung waren beispielsweise die Selbsteinschätzung, selbst nur wenig zum Projektfortschritt beigetragen zu haben oder dass das eigene Engagement schon zu weit in der Vergangenheit liegt, um dazu in einem Interview Aussagen machen zu können. Teilweise wurde die ablehnende Haltung auch mit Zeitmangel begründet.

Kompetenzaufbau bei Marlin

Nur in einem Fall wurde die Ablehnung der Interviewanfrage mit der Bereitschaft versehen, schriftlich auf Fragen zu antworten. Die Antworten stammen von der Person, die im späteren Verlauf auch die Rolle des:der PM übernahm. Mit Bezug auf den Kompetenzaufbau schrieb Marlin:

> „I think using logical and analytical parts of my brain has improved greatly. And that affects real life too. Of course, I have improvement my C++[95] skills with time and my English.“ (Dokument Marlin)

Neue Fähigkeiten und Fertigkeiten mit Bezug auf Programmiersprachen wie *C++*, in der *qBittorrent* entwickelt wird, wurden bereits vermutet. Interessant ist die Aussage, dass sich Verbesserungen des logischen und analytischen Denkens auch auf andere Lebensbereiche auswirken. Leider bleibt unklar, welche anderen Lebensbereiche davon profitieren.

[95] C++ ist eine Erweiterung der Programmiersprache C, um eine höheres Abstraktionsniveau zu erreichen.

Möglicherweise kann Marlin dies auch nicht konkretisieren, da es sich hierbei um implizites Wissen handelt. Implizites oder auch *schweigendes Wissen* zeichnet sich dadurch aus, dass über diese Wissensbestände nicht direkt Auskunft gegeben werden kann. Es ist dadurch charakterisiert, dass:

> „[...] relevante Aspekte des Handlungswissens nicht artikuliert und kognitiv nur in begrenztem Maße zugänglich sind. Dennoch präfigurieren diese Aspekte unsere Präferenzen und Intuitionen, gelingende oder misslingende Lernprozesse, prekäre oder erfolgreiche Lebenskonzepte. Diese Wissensformationen werden als ‚implizites', praktisches' oder ‚unbewusstes' Wissen, als ‚tacit knowledge' (Polanyi) oder als ‚knowing how' (Ryle) bezeichnet." (Kraus et al. 2017, S. 11).

Hier deuten sich die Schwierigkeiten an, die sich bei der Analyse des Datenmaterials zeigten. Aussagen über explizites Wissen wie zum Beispiel Fremdsprachenkenntnisse oder technisches Wissen über Programmiersprachen lassen sich leichter artikulieren als implizite Wissensbestände. Diese erfordern einen Reflekionsprozess der Person, um implizite Wissensbestände zu explizieren. Dies ist allerdings nicht Gegenstand der Kommunikation der Mailinglisten, weshalb solche Selbstaussagen nicht zu finden waren.

Dennoch können durch Marlins Antworten Aussagen zur Art und Weise des Lernens durch die Entwicklungsarbeit getroffen werden. Auf die Frage über den Stellenwert der Kommunikation mit anderen Entwickler:innen antwortete Marlin Folgendes:

> „It is very important especially when you want to discuss new features or new approaches. Also reviews on non-trivial code changes are always very helpful. No matter how good you are, you might forget a corner case. Or your eyes play tricks with you and you can't [see] a very obvious error/bug. So a reviews is always helpful and welcome. My questions are usually answered by a web search. I usually contact contributors via github issues/PRs." (Dokument Marlin)

Marlin führt aus, dass einfache Probleme beim Programmieren durch eine selbständige Internetsuche gelöst werden können. Dagegen bedarf es bei komplexen Problemen der Hilfe anderer Entwickler:innen. Dies zeigt sich in den Bitten um Rückmeldungen zu eigenen Codebeiträgen. Diese Anfragen tauchten regelmäßig auf der Dev-ML auf und wurden von erfahrenen wie auch unerfahrenen Entwickler:innen gestellt. Daraus kann abgeleitet werden, dass der gewählte Lösungsweg beim Programmieren von der Komplexität des Problems abhängt.

Mit Bezug auf die präferierten Kommunikationskanäle wird die Beobachtung bestätigt, dass Marlins maßgeblich genutzter Kommunikationskanal mit anderen Entwickler:innen nicht die Mailingliste der Entwickler:innen ist. Stattdessen kommen hauptsächlich die Kommunikationsmöglichkeiten von GitHub zum Einsatz. *GitHub Issues* ist der Bug-Tracker, der Teil eines jeden Git-Repositoriums ist. Damit lassen sich Fehler erfassen und

verwalten. Zu jeder Fehlermeldung gibt es eine Kommentarfunktion, die für alle schreibberechtigten Entwickler:innen offen steht. Zusätzlich gibt es diverse Möglichkeiten, die gerade für die PM interessant sind: Fehlermeldungen können zukünftigen Versionsnummern zugeordnet werden und sie können mit Schlagwörtern kategorisiert werden. Auch können Personen zu den Fehlermeldungen zugeordnet werden. Beide Funktionen sollen die Verwaltung der Fehlermeldungen vereinfachen.

Daneben existiert auch der Kommunikationskanal mittels *Pull-Requests*. Dieser entstammt der Funktionalität von Git, das sich als verteiltes Versionskontrollsystem von zentralisierten Systemen unterscheidet. In einem verteilten Versionskontrollsystem hat jede:r Entwickler:in eine eigene Kopie des Repositoriums. Änderungen an der lokalen Version werden gegenüber dem:der Projektmaintainer:in als Pull-Request kommuniziert. Diese:r entscheidet dann, ob dieser Projektbeitrag Teil des Hauptentwicklungszweigs des Projekts wird. Auch dieser Pull-Request besitzt eine Kommentarfunktion. Entwickler:innen schicken einen Beitrag als Pull-Request und begründen in dem Kommentar, welche Änderungen oder Erweiterungen durch diesen Beitrag zum Programmcode implementiert werden. Auch über diesen Kommunikationskanal können Rückfragen und Anforderungen an diesen Projektbeitrag diskutiert werden.

Aufgabenbereiche und Anforderungen in den verschiedenen Rollen

Die Fähigkeiten und Fertigkeiten entwickeln sich auch abhängig von den Aufgabenbereichen, der sich die Person widmet. Da Marlin im zeitlichen Verlauf des Projekts die Rolle als Entwickler:in wie auch als PM ausgeübt hat, ist ein Blick auf die jeweiligen Anforderungen dieser Rollen möglich:

> „Since I am the admin[96] now, I don't have to worry about other's approval. But contributors in general have to worry about admin approval. But as an admin I have to think about my users needs first and what they expect. Even though I have the power, I can't just code whatever **I** want.“ (Dokument Marlin)

Hier zeigt Marlin die unterschiedlichen Rollenerwartungen zwischen Entwickler:innen und Projektmaintainer:innen auf. Ob ein eingereichter Beitrag Teil des FLOSS-Projektes wird, obliegt der Entscheidung der:des PM. Dies zeigt die klaren Hierarchieunterschiede zwischen den Rollen. Entwickler:innen sollten sich deshalb bezüglich ihrer geplanten Beiträge im Vorfeld abstimmen, um keine Ablehnung ihres Beitrags zu riskieren. Auf der anderen Seite stellt der Verwaltungsaufwand als PM eine Herausforderung dar. Die Frage nach Hindernissen bei der Entwicklungsarbeit beantwortete Marlin folgendermaßen:

> „Time. Especially when the project is popular and a lot of bug reports come in. Or a lot of pull requests come in. Then you[r] time as admin is spent between triaging issues and reviewing patches. And you get less and less

[96] Das Wort Admin wird als Abkürzung für Administrator verwendet.

> time to code yourself. And this gets exacerbated if you [do] the project as hobby while working a real job (not necessarily in the software field). So in a sense [of] money too.“ (Dokument Marlin)

Die Aufgaben als PM belaufen sich auf die Sichtung und Priorisierung von Fehlerberichten und die Bewertung von Pull-Requests. Damit gewinnt das Verstehen und Bewerten eingereichten Quellcodes gegenüber dem eigenen Programmieren an Bedeutung. Die Fähigkeit, den Quellcode von andere Programmierer:innen in kurzer Zeit verstehen zu können, wird trainiert. Alle diese Tätigkeiten erfolgen im Falle von Marlin in der Freizeit und unbezahlt. Insofern ist die für die FLOSS-Entwicklung aufgebrachte Zeit auch immer in Bezug auf andere Anforderungen im Lebensalltag der Entwickler:innen zu sehen. Dies war wahrscheinlich auch ein Grund für die Übergabe der Rolle als Projektmaintainer:in an Marlin, da die:der vorherige Maintainer:in nicht mehr die erforderliche Zeit für diese Aufgaben erübrigen konnte oder wollte.

6.8 Zusammenfassung

In diesem Kapitel wurden die Ergebnisse der Beitragsanalyse der E-Mail-Kommunikation innerhalb einer Entwicklungsgemeinschaft vorgestellt und mit den Interaktionen bezüglich der Versionsverwaltung in Beziehung gesetzt. Über einen Zeitraum von circa neun Jahren wurde eine Beitragsanalyse aller E-Mails von zwei öffentlichen Mailinglisten durchgeführt, die von den Projektmitgliedern für die tägliche Entwicklungsarbeit genutzt wurden. Es zeigte sich dass es eine funktionale Trennung zwischen Entwickler:innen und Übersetzer:innen gibt, die sich auch anhand der beiden getrennten Mailinglisten manifestiert. Zwischen beiden Mailinglisten gibt es kaum personelle Überschneidungen, mit der Ausnahme der beiden PM. Die Entwicklungsarbeit in dem Projekt ist allerdings für die Projektmitglieder nicht an die Nutzung der Mailinglisten gekoppelt. Die Aktivitäten auf der Versionsverwaltung stehen nicht direkt mit dem Kommunikationsaufkommen auf den Mailinglisten in Verbindung.

Allerdings stellen die Mailinglisten für die PM ein effizientes Werkzeug dar, um die gesamte Projektgemeinschaft anzusprechen. Die übrigen Projektmitglieder können sich wiederum dadurch an kollektiven Entscheidungsfindungsprozessen beteiligen, sofern Bedarf dafür gesehen wird. Ebenso sind die E-Mails der Mailinglisten wichtig für die Gruppenkohäsion. Sie werden zur Anerkennung der Leistungen der anderen Projektmitglieder genutzt und geben Aufschluss über Rollenverständnis und die Stellung der jeweiligen Personen in dem sozialen Netzwerk der Projektgemeinschaft. Gleichzeitig stellen die Mailinglisten ein Archiv der relevanten Entwicklungsschritte dar. Über die Art und Weise, wie Ereignisse kommuniziert wurden, lassen sich Schlüsselereignisse identifizieren und teilweise ihre Bedeutung für die Projektgemeinschaft erahnen. Damit war es auch möglich, anhand der Beitragsanalyse die Praxis in einer Entwicklungsgemeinschaft für unterschiedliche Mitglieder in Bezug auf die arbeitsteilige Vorgehensweise zu demonstrieren.

Mehrere Beiträge in den E-Mails lieferten Beweise für die Vermutung, dass die öffentlichen Mailinglisten der Projektgemeinschaft nur einen Teilbereich der genutzten Kommunikationskanäle darstellten. Je nach Vorlieben der Personen und dem Grund der Kommunikation, werden spezifische Kommunikationskanäle abseits der Mailinglisten genutzt. Dies sind beispielsweise die direkte E-Mail-Kommunikation zwischen zwei Projektmitgliedern, die Nutzung von Chats oder den Kommunikationsmöglichkeiten von FLOSS-Entwicklungsportalen.

Dennoch sind die Mailinglisten ein wichtiger Kommunikationskanal, der hauptsächlich durch die PM zur kollektiven Ansprache an die übrigen Projektmitglieder genutzt wird. Die Mailinglisten dienen zur Vorbereitung für neue Programmversionen, die durch die PM strukturiert werden. Auf ihnen wird aber auch die Aufgabenteilung koordiniert und sie dienen als Diskussionsplattform bei Problemen samt der möglichen Lösungswege. In ihnen spiegelt sich die Entwicklungsgeschichte des FLOSS-Projektes wider. Sie stellen eine kollektiv erzeugte Dokumentation der sozialen Praxis und ihrer Entwicklungsschritte innerhalb der Projektgemeinschaft dar. Zugleich sind sie eine Wissenssammlung, welche neuen Mitgliedern eine Hilfe für die Partizipation an der Entwicklungsgemeinschaft bietet.

Bei der Betrachtung von zwei exemplarischen Diskussionen auf den Mailinglisten war zu beobachten, dass hauptsächlich einzelne Entwickler:innen oder Übersetzer:innen mit den PM kommunizieren. Eine Diskussionskultur zwischen den einzelnen Mitgliedern der Projektgemeinschaft untereinander konnte auch nicht in den übrigen E-Mails des Datenmaterials gefunden werden. Mit Blick auf die geringe Zahl gleichzeitig aktiver Mitglieder anhand der Versionsverwaltung liegt die Vermutung nahe, dass es an einer kritischen Menge an Mitgliedern für Diskussionen mangelte. Ein Indiz dafür könnte sein, dass tiefgreifende Entscheidungen über die Entwicklungsrichtung ohne lange Diskussionsprozesse innerhalb der Entwicklungsgemeinschaft über die Mailinglisten von den PM durchgesetzt wurden.

Die Inhaltsanalyse des Datenmaterials konnte keine befriedigenden Antworten auf die Frage nach dem Kompetenzaufbau der Projektmitglieder liefern. Die auf den Mailinglisten formulierten Fragen und die sich daraus ergebenden Diskussionen ließen keine klaren Rückschlüsse auf den Erkenntnisgewinn einzelner Individuen zu. Lediglich die Fortsetzung der Entwicklungsarbeit nach einem Problem ist ein Anhaltspunkt für einen Kompetenzaufbau. Solche individuellen Erkenntnisprozesse wurde jedoch nicht auf den Mailinglisten beschrieben.

Um dennoch zu Aussagen zum Aufbau von Fähigkeiten und Fertigkeiten treffen zu können, wurden Interviewanfragen an langfristig aktive Mitglieder verschickt. Aufgrund der geringen Interviewbereitschaft scheiterte dieser Versuch jedoch. Nur anhand der schriftlichen Antworten von einer Person konnten Informationen zu individuellen Lernprozessen gewonnen werden. Die Person verbesserte ihrer eigenen Einschätzung zufolge ihr logisches und analytisches Denken und eignete sich umfangreiche Kenntnisse in einer

Programmiersprache an. Auch Kenntnisse der englischen Sprache verbesserten sich durch den Austausch mit anderen Entwickler:innen und das Lesen englischer Dokumentationen bei der Problemlösung. Wie es jedoch genau dazu kam, blieb in den Antworten offen. Als wichtig wurde allerdings die Hilfe durch andere Entwickler:innen eingestuft. Problematisch bei der Beitragsanalyse und den schriftlichen Antworten Marlins ist die Schwierigkeit, implizites Wissen zu externalisieren. Die Nutzungspraxis der Mailinglisten zeigte, dass diese in erster Linie der Koordination der problemorientierten Entwicklungsarbeit dienen. Die individuellen Lernprozesse der Mitglieder sind nicht Bestandteil der vorgefundenen Kommunikationsakte gewesen.

Um überhaupt Aussagen dazu treffen zu können, braucht es einen vorausgehenden Reflektionsprozess, um implizites Handlungswissen ins Bewusstsein zu rücken. Das in diesem Kapitel untersuchte Datenmaterial ist nicht dazu geeignet, die Ergebnisse solcher Reflektionsprozesse zu rekonstruieren. Aus diesem Grund ist es notwendig, weiteres Datenmaterial hinzuzuziehen, um den Kontext von Lernprozessen zu beleuchten.

7 Interessen beim Engagement in FLOSS-Projekten

Das folgende Kapitel löst sich von dem Fokus auf ein spezifisches FLOSS-Projekt und betrachtet die diversen Interessen von Entwickler:innen für ihr Engagement in verschiedenen Projekten. Grundlage hierfür ist hauptsächlich der Fragenkomplex *Interesse* des Leitfadeninterviews.[97] Allerdings fanden sich auch in den Antworten zu Fragen anderer Fragenkomplexe Informationen zu den Sinnstrukturen, die für die Entwickler:innen bei ihrem FLOSS-Engagement relevant sind. Das Kapitel ist in vier Abschnitte unterteilt, die jeweils unterschiedliche Aspekte der Interessenlagen betrachten: Im ersten Abschnitt werden verschiedene Vorbedingungen aufgezeigt, die bei vielen Entwickler:innen vor ihrer ersten Mitarbeit in einer Entwicklungsgemeinschaft eine Rolle gespielt haben. Der darauf folgende Abschnitt beleuchtet Werteorientierungen, die beim FLOSS-Engagement zum Tragen kommen.

Der Hauptteil dieses Kapitels gilt der Darstellung der unterschiedlichen Interessen. Hierbei soll der Fokus auf die Änderungen der Interessen gelegt werden. Das hier genutzte Forschungsdesign ermöglicht Antworten auf die von Krogh formulierte Vermutung, dass sich Interessen im Laufe der Zeit verändern (vgl. Krogh et al. 2012, S. 669). Es ist anzunehmen, dass dies dadurch passiert, weil die Entwickler:innen durch die Praxis in einem FLOSS-Projekt neue Perspektiven kennenlernen und Erfahrungen machen, die ihre Interessen verändern oder diese anders gewichtet werden. Vor dem ersten Engagement in einer solchen Gemeinschaft sind die Praxen entweder unbekannt oder nur indirekt durch Überlieferungen anderer beziehungsweise durch die passive Beobachtung von FLOSS-Projekten bekannt. Aus diesem Grund wird in dieser Arbeit eine Unterteilung der Beweggründe vorgenommen: Zum einen in die Auslöser, die zu einem ersten Engagement in einer Entwicklungsgemeinschaft führen und zum anderen die Beweggründe, wie sie von Entwickler:innen nach einer längerfristigen Mitarbeit in FLOSS-Projekten angeführt werden.

7.1 Vorbedingungen für ein Engagement in einem FLOSS-Projekt

Anhand der Interviews zeigte sich, dass oft schon mehrere Jahre vor dem ersten Engagement in einer Entwicklungsgemeinschaft die Befragten FLOSS für den eigenen Gebrauch nutzten. Dies geschah oftmals durch die Nutzung von FLOSS auf einem proprietären Betriebssystem. Populär auch unter Nicht-Entwickler:innen sind die freien Browser, E-Mail-Programme, Textverarbeitungs- und Tabellenkalkulationsprogramme oder Software für die Bildbearbeitung. Es hat oft also schon im Vorfeld der aktiven Teilnahme an einem FLOSS-Projekte ein Prozess stattgefunden haben, der FLOSS als praktikable Alternative für proprietäre Software erscheinen ließ und dies zum Wechsel der Software führte. Dies

[97] Alle Fragenkomplexe und die jeweiligen Fragen des Leitfadeninterviews sind im Anhang dieser Arbeit aufgeführt

wird pragmatisch begründet, da diese Programme ohne gebührenpflichtige Nutzungslizenz erhältlich sind und insbesondere bei den populären Programmen stetig weiterentwickelt werden, gut dokumentiert sind und es eine hilfsbereite Community im Fall von Problemen gibt.

Diese Form der vorausgehenden Nutzung freier Software ist eine wichtige Grundlage für das Engagement in einem FLOSS-Projekt. Sie bildet die erste Kontaktfläche mit dem durch FLOSS aufgespannten Möglichkeitsraum der Partizipation auf unterschiedlichsten Ebenen. Um jedoch überhaupt partizipieren zu können, bedarf es verschiedener struktureller Bedingungen, die im Folgenden erläutert werden.

7.1.1 Strukturelle Bedingungen

Neben den FLOSS-bezogenen Faktoren braucht es zunächst die strukturellen Ressourcen, um sich überhaupt an einem derartigen Projekt beteiligen zu können. Eine der wichtigsten Faktoren bei ehrenamtlichen Arbeiten ist die Zeit:

> „Was für die Kommunikation wichtig ist, dass man hoffentlich einen Maintainer hat, der irgendwie zuständig ist, der halt zeitnah auch antwortet. Was natürlich auch schwierig ist, weil die Leute dann teilweise auch andere Jobs haben und dann auch nicht viel Zeit haben.“ (Dokument I-8, Abs. 68)

Dieses Zitat verweist auf die Bedeutung der Ressource Zeit. Sie ist nicht nur für das eigene Engagement notwendig. Durch die Austauschprozesse in der Gemeinschaft wirkt sich auch der Zeitmangel anderer Personen auf die Praxis der Entwicklungsgemeinschaft aus. Dies fällt besonders ins Gewicht, wenn es Entwickler:innen mit wichtigen Funktionen betrifft. Besonders für unerfahrene Entwickler:innen ist es wichtig, dass deren Fragen und Hilfsangebote zeitnah und konstruktiv beantwortet werden, damit sie die Art und Weise des wechselseitigen Engagements kennenlernen und sich das Repertoire der Praxisgemeinschaft aneignen können.

> „Dadurch, dass es im Moment eher noch so Fehlerkorrekturen sind, würde ich sagen, das Größte, und das ist auch das, was eigentlich am meisten Zeit braucht, ist immer dieses wie/ (.) Was muss ich jetzt beachten, wenn ich jetzt den Pull-Request mache oder brauche ich jetzt hier noch einen Account? Oder bei dem [Name FLOSS-Projekt] war es z.B. so, dass ich mir dann erst einen Account beantragen musste. Da gab es dann zum Spamschutz noch einen Mechanismus, wo ich dann auch nicht so ganz wusste, ob ich den Account jetzt erst in zwei Tage habe und dann war es dort so, dass ich den Patch im Bug-Tracker z.B. eingereicht habe. Bei Arch-Linux ist es durchaus üblich, dass für den Paketmanager z.B. die Patches auf der Mailingliste eingereicht werden und dann gibt es/ Da gibt es aber irgendwie leider kein Dokument, in welcher Form und wenn man da eben vorher noch keine Erfahrung hat, dann ist man da auch erst einmal ein Stückchen beschäftigt: Wie mache ich das?

> Und man findet dann/ Oh hier bei Git gibt es so einen Befehl, der dann genau schon dafür die Ausgabe erzeugt und so was/ [...] weil diese kleineren Sachen, wenn man es nicht wiederholt, ist es eben weg.“ (Dokument I-1; Abs. 49)

Dieser Entwickler schildert die Hürden auf dem Weg zu einem eigenen Beitrag zur gemeinsamen Codebasis. Es geht nicht nur darum, sich die Kenntnisse einer Programmiersprache anzueignen. Auch die projektspezifischen Werkzeuge für die kollaborative Arbeit müssen beherrscht werden, um an der Praxis partizipieren zu können. Der Entwickler beschreibt die verschiedenen Strukturen und Prozeduren, die er sich bei seinem Engagement in mehreren FLOSS-Projekten aneignen musste. Diese unterscheiden sich zwischen den Projekten und können sich auch im zeitlichen Verlauf der Entwicklungsarbeit ändern, wie es zum Beispiel auch beim qBittorrent-Projekt dokumentiert wurde. Daher erfordert jedes FLOSS-Projekt neben der eigentlichen Entwicklungsarbeit ein kontinuierliches Lernen bezüglich der projektspezifischen Prozeduren und der gemeinsam genutzten Werkzeuge.

7.1.2 Nutzung von freier Software

Wie in der Einleitung schon erwähnt wurde, geht einem FLOSS-Engagement oft eine FLOSS-Nutzung voraus. Wird der Fokus auf die Nutzung von FLOSS konsequent weiter gedacht, wird diese Frage auch auf der Wahl des Betriebssystems ausgedehnt. Eine der wichtigen Schlüsselerlebnisse auf dem Weg zur FLOSS-Partizipation war der Umstieg auf eine GNU/Linux-Distribution. Damit ist eine stärkere Selbstverpflichtung verbunden, da gewohnte Nutzungskonzepte umgelernt werden müssen und bestimmte Programme möglicherweise nicht mehr genutzt werden können oder dies mit erheblichem Aufwand verbunden ist. Auch schränkt sich damit die Zahl derer ein, die bei Computerproblemen helfen können. Eine solche Entscheidung führt zwangsläufig dazu, dass Probleme vermehrt selbst gelöst werden müssen.

Als Übergang wird oftmals auch die parallele Nutzung von Windows und GNU/Linux auf einem Computer praktiziert.[98] Der jeweilige Nutzungswunsch entscheidet dann darüber, welches Betriebssystem gestartet wird. Die Nutzer:innen lernen Schritt für Schritt die Möglichkeiten des neuen Systems kennen und gehen dazu über, immer mehr Tätigkeiten am Computer mit dem neuen Betriebssystem zu verrichten.

> „[...] als ich mit der Schule dann fertig war und mich dann auf berufliche Ausbildung im Studium konzentriert habe, war es so dieses: Ja jetzt, also von Linux hat man jetzt viel gehört, jetzt probierst du das eben mal aus. Und da hatte ich mit Ubuntu angefangen. Und nach und nach hat man dann angefangen, nicht nur immer so zwischen Windows und Linux hin- und herzuwech-

[98] Ein Hybrid-System mit Windows und GNU/Linux als Betriebssystem arbeitet mit unterschiedlichen Festplatten oder Festplattenpartitionen in einem Computer. Beim Starten des Computers wird das gewünschte Betriebssystem ausgewählt und ist dann für die jeweilige Nutzungsdauer das maßgebliche Betriebssystem.

seln, sondern einfach immer mehr unter Linux zu machen und das dann auch einfach im Studium und in der Vorlesung benutzt. Dadurch bin ich eben sehr dann da reingekommen und seit zwei Jahren nutze eigentlich primär Linux und das Windows ist eher so zum Spielen noch verkommen." (Dokument I-1; Abs. 45)

Die Nutzer:innen lernen schrittweise die Möglichkeiten des neuen Systems kennen und können bei Problemen jeder Zeit wieder zu ihrem alten System wechseln. Das Interesse an der Nutzung von FLOSS führt dann dazu, dass immer mehr Tätigkeiten am Computer mit dem neuen Betriebssystem verrichtet werden. Dadurch erfolgt eine Transition der Nutzung dergestalt, dass das alternative Betriebssystem GNU/Linux mit der Zeit zum Hauptsystem avanciert.

„Also man hat irgendwie Leute aus dem Freundeskreis, die ähnliche Software benutzen wollen und dann kann man sich irgendwie austauschen. Aber da ist bei ganz Vielen halt schnell abschalten gewesen im Sinne von: Nee, das ist mir zu aufwändig, keine Lust mich darum zu kümmern und durch viele Sachen habe ich mich dann selber durchgewurschtelt bis sie funktioniert haben. So ein Schritt wie (.) ich stelle jetzt mal auf Linux um, ist schon erst mal ganz schön aufwändig, ich habe es ja selber auch gemacht. Genau, ich erinnere mich, doch doch, wir hatten/ Doch das war ganz spannend: Ich hatte einen Freund, der hat tatsächlich auf Linux gesetzt. Da hatte ich schon aus der Ferne gesehen: Oh, das läuft ja doch alles irgendwie. Und was mich beeindruckt hatte, der hatte es geschafft Warcraft 3[99] unter Linux zum Laufen zu kriegen und wir hatte alle unter Windows gespielt. Und Warcraft 3 war zu dieser Zeit das Spiel, was wir halt am allermeisten gespielt haben und ein entscheidender Faktor, warum ich überhaupt noch Windows auf meinem Rechner habe. Und dann zu sehen: Hey, der hat es geschafft Warcraft 3 zum Laufen zu kriegen und es funktioniert, vernünftig ne', also ohne irgendwelche großen Einschränkungen. Ich: Okay, dann kannst du auch andere Spiele zum Laufen bringen und dann kannst du auch endgültig weg, genau/ Also da war eine dritte Person die mir vorgemacht hat, dass es funktionieren kann." (Dokument I-13; Abs. 202)

Der Entscheidung zu Linux zu wechseln, ging eine fundamentale Kritik an den Lizenzbedingungen von proprietärer Software voraus. Erst diese Kritik löst eine Suche nach Alternativen aus, die in dem Wechsel des Betriebssystems gipfelte und den Grundstein für das spätere Engagement legte. Durch vorangestelltes Zitat wird auch die Rolle von *peers* deutlich: Durch das Vorbild eines Freundes wurde der Wechsel zu GNU/Linux überhaupt erst als umsetzbares Handlungsziel in Erwägung gezogen. Weiterhin fungieren *peers* als Bezugspersonen für den Wunsch nach Austausch und Hilfe bei einem Prozess, dessen Umsetzung als schwierig eingeschätzt wurde.

[99] Warcraft 3 ist ein Echtzeit-Strategiespiel, dass 2002 für Windows und Mac OS veröffentlicht wurde.

Bei beiden Beispielen zeigt sich der Stellenwert von Computerspielen bei der Computernutzung: Während es für die Alltagsanwendungen am Computer viele FLOSS-Alternativen gibt, ist das Angebot bei populären Computerspielen für GNU/Linux beschränkt.

7.1.3 Biographische Erfahrungen mit der Schenkökonomie

Statt der FLOSS selbst können auch die Praktiken, wie sie bei einer FLOSS-Gemeinschaft üblich sind, biographisch verankert sein. Einige Befragte waren schon in ihrer Jugend Teil von einer Community, bei der der gemeinschaftliche Austausch von Wissen und die daraus hervorgehende Software ein fester Bestandteil der gemeinschaftlichen Praxis war. Ohne die Begriffe *open-source* oder *Freie Software* zu kennen, nutzten sie ähnliche Strukturen und Prozesse in einer eigenen Community.

> „Und ich glaube, mein Bekanntenkreis, in dem ich halt ganz früh in meiner Kindheit so rumgebastelt habe, in dem haben wir das ganz natürlich einfach auch schon gelebt. Wir haben da so in den anfänglichen Computerzeiten von uns so den Zweisechsundachtziger[100], die wir dann irgendwie von Papa aus dem Altlager in die Hand gedrückt bekommen haben, haben wir dann selber Computerspiele geschrieben, die wir dann untereinander getauscht haben. Und in dem Eindruck untereinander, neue Ideen, die wir dann dazugefügt haben, auch getauscht haben ohne uns Gedanken darüber zu machen, dass das jetzt open-source oder irgendwie Freie Software oder Sonstiges wäre. Dieser Austausch, der war ganz/ Ja der war halt einfach ganz natürlich da. Im Endeffekt habe ich das einfach immer nur noch weitergelebt.“ (Dokument I-7; Abs. 92)

Ein weiterer Interviewpartner berichtet von einer Community, die sich mit einem kommerziellen Taschenrechner beschäftigt. Als damals Zwölfjähriger begann er zusammen mit anderen Interessierten, die Funktionsweise dieses proprietären Taschenrechners zu rekonstruieren. Diese Tätigkeit wird als *Reverse-Engineering* bezeichnet.[101] Ziel ist die Offenlegung der Funktionsweise von Soft- und Hardware, damit sie für eigene Zwecke angepasst werden kann.

> „Das hatte durchaus zum einen einen Entwickler-Gemeinschaft, die/ Wo man halt in Gruppen einfach aktiv ist, an Dingen halt mitarbeitet. Und man halt einfach in der Gruppe entscheidet: Okay, dass, was wir halt bauen, stellen wir anderen Leuten halt auch mit bereit. Und da war dann halt/ Ich bin ja auch in dem Bereich TI-83 Plus[102], da gab es auch eine sehr aktive Community, die halt Assembler[103]-Programmierung dort gemacht hat. Und für

[100] Damit ist der Mikroprozessor 80286 gemeint, der von Intel ab 1982 auf dem Markt kam und von IBM in seinen PCs verbaut wurde.

[101] Als Reverse-Engineering wir der Prozess bezeichnet, wo durch die Analyse eines bestehenden Systems dessen Konstruktionselemente und Funktionsweisen ermittelt werden.

[102] Grafikfähiger Taschenrechner von Texas Instruments, der 1999 auf dem Markt kam.

[103] Assembler ist eine maschinenorientierte Programmiersprache.

> Assembler-Programmierung braucht man halt irgendwo die Listen, wo welche Offsets[104] für Speicherbereiche liegen und da haben halt irgendwo Leute die Listen bereitgestellt und durch Reverse-Engineering, was man ja nebenher beim Debugging[105] dort macht, wenn man eh auf Assembler-Ebene[106] unterwegs ist, vervollständigt man seine eigenen Listen ja auch noch um Sachen, die man da findet und da war dann so die Sache, dass ich im Forum dann auch gepostet habe: Ja, ich habe die und die Adresse noch gefunden. Und Leute fragen dann halt an: Ja, können wir die entsprechenden Adressen, Offsets auch einfach mit kriegen? Also da [ist] es dann halt so, man hat selber zwar erst einmal investiert mit/ Aber dadurch, dass man meistens aus der Community schon so eine Vorleistung hatte, ist dann die Schwelle relativ niedrig, um halt auch zu sagen: Ihr habt in der Community mir Dinge bereitgestellt, dann gebe ich dann auch entsprechend das, was ich zusätzlich finde auch wieder zurück.“ (Dokument I-6; Abs. 144)

Das Beispiel beschreibt den reziproken Tausch von Speicheradressen in der Community, die sich um einen proprietären Taschenrechner herum gebildet hat. Die Adressen sind die Voraussetzung dafür, eigene Software für den Taschenrechner zu programmieren oder auch die vorhandene proprietäre Software zu modifizieren.

Beide Beispiele zeigen, in welchem Kontext schon in der Kindheit und Jugend das Programmieren erlernt wurde. Beide Personen waren eingebettet in eine soziale Gemeinschaft mit gemeinsamen Zielen und Praktiken. Das Ziel war nicht ein kollektiv erschaffenes Produkt, aber die Aneignung von Hard- und Software zur Verwirklichung eigener Ideen. Die Gemeinschaft gab diesem gemeinsamen Lernen einen sozialen Resonanzboden. Der lebt davon, dass Einzelne ihr Wissen mit der Gemeinschaften teilen und im Gegenzug auf das Wissen anderer zurückgreifen können. Dies entspricht denselben Praktiken, wie sie auch in FLOSS-Projekten zu finden sind.

Als Vorbedingung für ein Engagement in FLOSS-Gemeinschaften kann also festgehalten werden, dass es schon weit vor der eigentlichen Partizipation bei der Mehrheit der befragten Personen einen individuellen Bezug zu FLOSS gibt. Dieser erfolgt zum einen über die eigene Nutzung von FLOSS und zum anderen über die Akzeptanz beziehungsweise bereits die Anwendung der Normen und Werte der FLOSS-Bewegung in anderen Kontexten.

[104] Um auf Daten und Algorithmen im Speicher des Rechners zugreifen zu können, müssen die Speicherorte bekannt sein. Eine Adresse besteht aus dem Speichersegment und dem Offset. Der Offset gibt an, an welcher Stelle im Speichersegment die Daten lokalisiert sind.

[105] Das Beheben von Programmfehlern wird allgemein als Debugging bezeichnet. Ein Debugger als Werkzeug ermöglicht das Anzeigen von Befehlen, die gerade vom Prozessor ausgeführt werden. Daraus lassen sich die Algorithmen und die genutzten Daten rekonstruieren.

[106] Damit können die Befehle im Mikroprozessors des Taschenrechners Schritt für Schritt nachvollzogen werden. Die vom Prozessor ausführten Rechenoperationen werden vom Debugger als Assembler-Befehle angezeigt.

7.2 Werteorientierung bezügliche FLOSS als allgemeines Gut

Die Interessen für ein Engagement bei FLOSS-Projekten lassen sich auch auf grundlegenden Überzeugungen und Orientierungsmuster der Entwickler:innen zurückführen. Durch die Interpretation von verschiedenen Interviewpassagen lassen sich diese Werte deuten. In den Interviews finden sich an vielen Stellen Anhaltspunkte für diese Werteorientierungen. In dieser Arbeit konnten vier Werte identifiziert werden: soziale Teilhabe, reziproke Hilfsbereitschaft, der Schutz von Kulturgütern und die Partizipation. Diese wurde in vielfältigen Formen in allen Interviews gefunden und werden im Folgenden anhand typischer Beispiele erläutert.

7.2.1 Ermöglichung Sozialer Teilhabe

Die Merkmale von FLOSS ermöglichen eine Senkung der Hürden, um Soft- und Hardware nutzen zu können. Diese sind wiederum die Voraussetzung zu anderen Diensten beziehungsweise ermöglichen den Zugang zu Informationen. Dies ist insbesondere für einkommensschwache Personengruppe wichtig, die ansonsten von bestimmten Bereichen der Gesellschaft ausgeschlossen sind:

> „So, und wenn man dann sich halt so ein bisschen mit den Problemen, die wir in der Gesellschaft haben, insbesondere im digitalen Umfeld, beschäftigt, dann ist es halt so ein Projekt, in das man dann auch reinrutschen kann. Also, wir haben eine Gesellschaft, die immer mehr digitalisiert, immer mehr Aspekte, die wir vorher in der analogen Welt erledigt haben, [werden] in die digitale Welt verschoben. Auf der anderen Seite haben wir einen immer größer werdenden Bevölkerungsanteil, der an diesem Leben nicht teilnehmen kann. Weil wir halt im Kern immer, wenn wir halt die digitale Welt betreten wollen, irgendeinen Carrier haben, irgendein Telekommunikationsunternehmen haben, was uns halt diese Möglichkeit an diesem Netz teilzunehmen oder am digitalen Leben teilzunehmen, verkaufen. So, und die Idee, da zu sagen, okay, wir brauchen eigentlich eine Infrastruktur, die wieder in den Händen der Gesellschaft liegt und die für Leute halt in Anführungszeichen erschwinglich ist, egal in was für Umständen sie (.) sich befinden. Und deswegen bin ich halt mehr oder minder immer weiter in das Thema reingerutscht und/ Naja, größtenteils wegen den sozialen Aspekten und in den Diskussionen darum dann bin ich auch viel in die technischen Probleme dann immer weiter eingestiegen und habe da dann/ Pflege ich da halt auch Software [...].“ (Dokument I-7; Abs. 12)

Dieser Entwickler ist Teil eines FLOSS-Projektes, welches sich mit dem Aufbau eines freien Funknetzes beschäftigt. Durch die Vernetzung verschiedener WLAN-Router[107]

[107] Ein WLAN-Router bildet die technische Schnittstelle zwischen zwei Netzwerken. Meist ist es ein lokales Funknetzwerk, welches durch das Gerät mit einem kabelgebunden Netzwerk verbunden wird. Das kabelgebundene Netzwerk kann über ein Internetmodem mit dem Internet verbunden sein.

kann damit der Zugang zum Internet und anderen Netzwerkdiensten ermöglicht werden. Im Abdeckungsbereich dieser Router ist der kostenfreie Zugang zu Internetdiensten möglich. Dies ist besonders für die Menschen wichtig, die über kein ausreichendes Budget für einen Vertrag mit einem Telekommunikationsanbieter verfügen oder ihnen der Vertragsabschluss aus strukturellen Gründen nicht möglich ist.

Neben dieser sehr direkten Form der Unterstützung durch die Bereitstellung von Infrastruktur folgen auch alle andere FLOSS-Projekte dem Anspruch nach sozialer Teilhabe. Da FLOSS nicht an die Zahlung von Lizenzgebühren gekoppelt ist, ermöglicht sämtliche FLOSS in irgendeiner Form die Teilhabe an diversen Diensten, Informationsangeboten oder die Nutzung des eigenen Computers für bestimmte Aufgaben. In diesem Sinne bedeutet das Engagement bei einem FLOSS-Projekt die Förderung von sozialer Teilhabe von Menschen durch die Nutzung von Informations- und Kommunikationstechnologien.

7.2.2 Schutz von Kulturgütern

Bei der Betrachtung von Software als kulturelles Gut fällt der Quelloffenheit eine besondere Bedeutung zu. Der im Folgenden zitierte Entwickler vergleicht den Lebenszyklus von FLOSS mit der von proprietärer Software.

> „Naja, (.) ich finde das vom Ziel her eine ganz feine Sachen, dass die Software die ich habe, irgendwie schon frei erhältlich ist im Sinne, dass ich mir halt auch den Quelltext anschauen kann und wenn es mal nicht funktioniert, dass ich im Zweifel tatsächlich in dem Quelltext irgendwas machen könnte. Ob man das jetzt tatsächlich macht, ist ja mal dahingestellt, aber (.) an sich finde ich den Ansatz von Freier Software halt schon sehr gut. Ich meine, ich benutze jetzt Linux seit inzwischen fast zehn Jahren und die Philosophie dahinter finde ich eigentlich ganz gut, dass man jetzt halt nicht irgendwie so einen Mist hat wie, was weiß ich, was ich bei irgendwelchen Leute mitkriege, die so: Hey, ich habe hier diese wunderschöne alte Windows-Software, die läuft aber unter meinem Windows 10 nicht mehr, was kann ich denn da machen? (.) Mhh, kannst du nichts machen, weil der Hersteller ist zum Beispiel schon seit zehn Jahren pleite, existiert nicht mehr oder es gibt ihn noch, aber die haben natürlich kein Interesse daran. Und da könnte man halt im Zweifel sagen/ Könnte ich halt schauen, ob ich das halt doch irgendwie hinkriege. Ja, das geht halt bei proprietärer Software nicht. “ (Dokument I-12; Abs. 12)

Durch die Verfügbarkeit des Quellcodes der Software kann der Produktlebenszyklus von einer nicht mehr gepflegten FLOSS wieder neu gestartet werden. Die komplette Neuentwicklung entfällt, da nur die inkompatiblen beziehungsweise die fehlerhaften Codestrukturen ausgetauscht werden müssen. Der ökonomische Aufwand für die Wiederbelebung von Software kann sich dadurch unter Umständen erheblich reduzieren.

7.2.3 Reziproke Hilfsbereitschaft

Fast alle Entwickler:innen erwähnten den Wunsch, der Gesellschaft durch ihre Arbeit etwas zurückgeben zu wollen. Dies impliziert oftmals biografische Bezüge: Ein wichtiges Schlüsselerlebnis ist die Entscheidung, von Microsoft Windows als proprietäres Betriebssystem hin zu einem Freien Betriebssystemen zu wechseln, beziehungsweise innerhalb des selbst genutzten Betriebssystems so viel Freie Software wie möglich einzusetzen. Dies zeigt sich in einem längerfristigen Transformationsprozess im Denken und Handeln in Bezug auf die eigenen Softwarenutzung bis hin zur aktiven Mitwirkung an FLOSS-Projekten. Als exemplarisches Beispiel dazu folgender Interviewausschnitt:

> „Aber ich habe selber natürlich immer sehr viel offene Software benutzt. Was vor allem daher kommt, dass ich/ Ich bin vor 15 Jahren oder so auf Ubuntu[108] umgestiegen. Und da ist ja von Haus aus quasi alles frei. Und es war mir ein Anliegen, mich auch immer umzugucken, wo kriege ich freie Alternativen her. Ah, für mein Handy auch, genau. Das ist halt immer ein Aufwand und da habe ich immer das Gefühl, wenn man Sachen macht, dann muss man es auch den Leuten die bereit sind, sich diesen Aufwand zu machen, möglichst leicht machen, an den Kram zu kommen. Ich finde das total gruselig. Ich stelle mir gerade vor, da ist irgendjemand, der sagt, Open-Source ist geil. Ich habe mein Handy freigemacht, meinen Laptop freigemacht und möchte gerne deine Software benutzen und ich sage dir: Nee, geht nicht, weil ist nicht frei verfügbar. Also, ich muss den Leuten, die quasi mit auf der richtigen Seite sind, schon irgendwie unter die Arme greifen." (Dokument I-13; Abs.42)

Die Existenz eines reichhaltigen Spektrums an FLOSS wäre ohne das hauptsächlich ehrenamtlich geleistete Engagement unzähliger Entwickler:innen nicht denkbar. Die kostenfreie Nutzung von FLOSS bedeutet damit auch immer die Aneignung fremder Arbeitskraft. Dessen sind sich viele Entwickler:innen bewusst, weshalb sie selbst das Bedürfnis haben, sich für diese anonyme Hilfe zu revanchieren:

> „Also es ist halt so: Man hat selber zwar erst einmal investiert mit, aber dadurch, dass man meistens aus der Community schon so eine Vorleistung hatte, ist dann die Schwelle relativ niedrig, um halt auch zu sagen: Ihr habt in der Community mir Dinge bereitgestellt, dann gebe ich dann auch entsprechend das, was ich zusätzlich finde auch wieder zurück. Und so ist es halt bei vielen Projekten. Dadurch, dass ich ja meistens wirklich darauf gestoßen bin auf ein Projekt, was halt existierte und es nicht so ganz dem entsprach, was ich halt brauchte/ Kurz Patches geschrieben habe, dann ist es ein Einfaches für mich, die Patches halt auch einfach wieder dort öffentlich bereit zu stellen, so dass andere auf den Patches im Zweifel gleich schon aufsetzen können.

[108] Ubuntu ist GNU/Linux-Distribution, die auf einfache Installation und leichte Bedienbarkeit hin optimiert ist.

> Und denen die Arbeit halt ersparen, weil man kriegt im Zweifel halt auch viel Feedback von den Leuten, die es dann mit nutzen und relativ viel Dank auch mit.“ (Dokument I-6; Abs. 144)

Der Entwickler beschreibt eine generalisierte Reziprozität. Es findet kein direkter Tausch von in FLOSS oder anderen Dienstleistungen geronnene Arbeitszeit statt. Stattdessen wird die Hilfe zeitlich und personell entkoppelt.

7.2.4 Partizipation

Aufgrund der Verfügbarkeit des Quellcodes ist eine Partizipationsmöglichkeit formell gegeben. Dies bedeutet jedoch nicht, dass sich jedes FLOSS-Projekt auch als Community-Projekt eignet. Verschiedene andere Voraussetzungen müssen erfüllt sein, damit sich um ein Projekt auch eine Gemeinschaft aus Entwickler:innen und interessierten Nutzer:innen aufbaut. Zwei Beispiele machen dies deutlich:

> „Also es gibt halt oder gab zu der Zeit nur kommerzielle Angebote. Und dann da irgendwie eine Alternative zu schaffen, an der halt auch jeder partizipieren kann. Also ich glaube, ich finde diesen Gedanken wichtig, dass da so eine Art Infrastruktur ist, an der halt alle teilnehmen können, sie verbessern können, für ihre Zwecke nutzen können. Diesen Gedanken finde ich einfach interessant und der hat mich dann auch/ Also man ist dann einfach motiviert, vielleicht mit den eigenen Fähigkeiten diesem Projekt irgendwie zum Erfolg zu verhelfen. Genau (.) und das zieht sich halt vielleicht auch durch die anderen Projekte oder beim [FLOSS-Name] ist es halt auch so, das ist halt im Grunde eine sehr grundlegenden Infrastruktur. Menschen kommen an Informationen, können darüber kommunizieren und ja (.) wenn man diese Idee dann interessant findet und das konsequent verfolgt, dann guckt man natürlich irgendwie, wo kann man seine Fähigkeiten einbringen. Und bei mir ist es dann natürlich alles, was irgendwie mit Informatik zu tun hat.“ (Dokument I-2; Abs. 8)

Am Anfang steht oft ein Projekt, dass potentielle Entwickler:innen in irgendeiner Art und Weise bewegt und das Interesse weckt. Um sich aber auch einbringen zu können, braucht es zunächst eine Struktur, die Beteiligungsangebote wertschätzend annimmt und in den Entwicklungsprozess produktiv einbindet. Bei kleinen Projekten ist dies meist die Aufgabe von den Projektmaintainer:innen selbst. Bei größeren Projekten kann dies delegiert werden. Zusätzlich können die Einstiegshürden für interessierte Entwickler:innen so gering wie möglich gehalten werden. Dies wird beispielsweise damit erreicht, dass die zu erledigenden Aufgaben offen dokumentiert werden, der Partizipationsprozess transparent dargestellt wird und mit allgemein zugänglichen Entwicklungswerkzeugen gearbeitet wird. Nicht zuletzt ist auch der Umgangston innerhalb der Entwicklungsgemeinschaft ein nicht zu unterschätzender Faktor, wie es dieser Entwickler beschreibt:

> „Und eben dass es relativ leicht war, dort auch selbst was beizutragen, also eigentlich die offene Art auch von [Name PM], dass er gleich gesagt hat: Ja, wenn du da was hast, kannst du da beitragen. Das fand ich eigentlich gut. Also, das ist bei manchen Projekten halt nicht so. Da braucht man erst eine ganze Weile und muss da formell irgendwie kommunizieren, um da (.) ranzukommen. Das war dort überhaupt kein Problem.“ (Dokument I-9; Abs. 67)

Die Möglichkeit zur Partizipation ist nicht nur durch das Konzept von Open-Source gegeben, es bedarf vielmehr eine Reihe zusätzlicher Komponenten, um interessierten Entwickler:innen die gewünschte Partizipation auch zu ermöglichen. Um Partizipieren zu können, braucht es die notwendigen Kompetenzen beziehungsweise den Willen, sich diese anzueignen und die notwendigen Rahmenbedingungen für ein gelingendes Engagement.

7.3 Auslöser für das erste Engagement bei einem FLOSS-Projekt

Als Engagement an der gemeinschaftlichen FLOSS-Entwicklung werden in dieser Arbeit die Tätigkeiten verstanden, die über die passive Nutzung von Software hinausgehen. Dies bedeutet nicht in jedem Fall, dass die Person selbst programmieren muss. Bereits das Testen von Software und die anschließende Meldung von Programmfehlern ist ein wichtiger Teil der Softwareentwicklung. In Anlehnung an das Schalenmodell nach Nakakoji et al. (2003) (vgl. Kapitel 4.1.1) entspricht dies den sechs innersten Schalen einer FLOSS-Gemeinschaft.

In Bezug auf die erste Beteiligung stellt sich die Frage, was der Auslöser für ein Engagement in einem konkreten FLOSS-Projekt ist. Alle Befragten gaben an, einen direkten Bezug zu dem jeweiligen FLOSS-Projekt zu haben. Oft gibt es einen konkreten Auslöser, der dazu führt, dass die Nutzer:innen den qualitativen Sprung zu Entwickler:innen vollzogen. Im Folgenden werden die Auslöser dargestellt, die die einzelnen Personen dazu bewogen, sich in einem FLOSS-Projekt zum ersten Mal zu engagieren.

7.3.1 Software eigenen Ansprüchen anpassen

Einer der am häufigsten genannten Gründe der Interviewten ist die Verbesserung von bestehender Software. Dabei handelt es sich um Software, die selbst genutzt wird. Entweder geht es darum, Fehler schneller zu beheben oder eine vorhandene Funktionalität den eigenen Ansprüchen anzupassen. Die hier aufgeführten Beispiele sollen auch die Themenbereiche und die Einstiegshürden in Bezug auf das Vorwissen wiedergeben.

Eine FLOSS-Entwicklerin berichtet über eine schlechte Sprachübersetzung in der selbst genutzten Software, die sie nicht hinnehmen wollte:

> „Ich mag gerne Sprachen, ich mag gerne Sprachen lernen, ich mag gerne (.) ja Übersetzen. Das macht mir einfach Spaß. Und ursprünglich das Erste, was

> mich quasi zum Projekt gebracht hat, war eine schlechte Übersetzung. [...]. Ja, voll genervt in den Bug-Tracker gegangen und habe gesagt: Ey, das ist ja so scheiße übersetzt, das klingt ja wie Google-Translate[109]. [...] Dann habe ich mich beschwert und dann wurde ich gleich abgefangen: Ja willst du es denn besser machen? Dann habe ich gesagt: Ja klar! Naja und damit fing es dann an.“ (Dokument I-3; Abs. 45 - 53)

Durch ihren Anspruch an eine korrekte Übersetzung wurde die Entwicklerin Teil des Übersetzungsteams. Ihr Engagement bei dem FLOSS-Projekt weitete sich dann auch auf das Schreiben der Dokumentation, Softwaretest und die Unterstützung von Nutzer:innen dieser Software aus.

Ein anderer Nutzer entdeckte einen Fehler bei einem von ihm genutzten FLOSS-Browser. Um diesen Fehler zu melden, suchte er nach dem Bug-Tracker von diesem Projekt und fand ihn auf dem FLOSS-Portal GitHub:

> „Das war wieder einmal etwas, das hat mich genervt, dass es nicht ganz genauso ging wie ich wollte. Dann war sogar schon das Issue[110] offen auf GitHub. War dann ein bisschen längeres Issue und dann habe ich es einfach/ Ja, so vielleicht 200 Zeilen [Quellcode] oder so, waren es dann insgesamt, [die ich] daran geschrieben [habe]. Ich merkte selbst, es funktionierte nicht so wie es sollte, auf GitHub nachgeschaut. Ist es bekannt oder wäre es was Neues? Wenn es was Neues wäre, hätte ich das Issue irgendwie erst einmal reported und dann geschaut, ist es etwas das auch andere nervt? Ist es etwas, was ich nur für mich patchen sollte? Nachdem es schon offen war, war klar: Ja, okay, da gibt es Interesse. Ich habe mich dazu gemeldet. Soll ich mich daran setzen, soll ich daran arbeiten und dann wird es halt gemacht.“ (Dokument I-11; Abs. 32)

Dieser FLOSS-Entwickler beschreibt sein Vorgehen bei der Behebung eines Fehlers über das FLOSS-Portal GitHub. Hier deutet sich schon ein wichtiger Faktor für das individuelle Engagement an: Viele FLOSS-Entwickler:innen orientieren sich auch an der Nachfrage durch andere Nutzer:innen. Wird ein Softwarefehler nur auf dem eigenen Computer behoben, ist der Lösungsaufwand oft geringer. Hintergrund sind beispielsweise die erhöhten Anforderungen an den Programmierstil und der Dokumentation des Quellcodes, falls der Code Teil eines öffentlichen FLOSS-Repositoriums werden soll. In diesem Fall muss der Quellcode so geschrieben sein, dass er auch von anderen Entwickler:innen gut nachvollzogen werden kann.

Ein anderer Entwickler beschreibt den Auslöser für sein FLOSS-Engagement und wie er dadurch dann die Rolle als Maintainer übernahm aus einer anderen Perspektive:

[109] Google Translate ist ein Übersetzungsdienst von Google, der auf Grundlage von maschinellem Lernen auch ganze Textpassagen übersetzen kann.

[110] Ein ’Issue’ ist die Meldung eines Fehlers. Es entspricht einen Bug-Report. Auf dem Entwicklungsportal GitHub werden Bug-Reports ’Issues’ genannt.

> „Ja, ich pflege so ein bisschen Utility-Kram, den ich selber mal gebraucht habe und der die Funktionalität nicht hatte/ Also [FLOSS-Name], das ist so ein Tray-Icon. Man kennt es so, unten rechts an der Leiste die kleinen Icons in den Menüs. Da gibt es halt verschiedene Möglichkeiten, die Funktionalität in die eigene Oberfläche einzubetten. Und da habe ich halt (.) da irgendwann das Projekt [unverständlich] bereitgestellt, quasi übernommen. Weil keiner der dort notierten Maintainer in irgendeiner Art und Weise ansprechbar war. Und ich habe halt selber dafür Patches geschrieben, um mehrere Monitore benutzen zu können, damit die Positionierung denn halt immer ordentlich stimmt. Und wollte diese dann einreichen und dann gab es da niemanden. Ja (.) und dann habe ich das geforked.“ (Dokument I-7; Abs. 20)

Ohne Absprache mit der Projektgemeinschaft behob dieser Entwickler die Programmfehler und wollte diese an die Maintainer:innen des Projektes zur Veröffentlichung zur Verfügung stellen. Weil diese jedoch die Hilfsangebote ignorierten, blieben ihm nur zwei Möglichkeiten: Entweder die Fehlerbehebung wird nicht veröffentlicht oder der Entwickler kopiert die gesamte Codebasis und veröffentlicht das gesamte Projekt unter einem anderem Namen neu. Dieses Vorgehen wird als *Fork* im Sinne von Abspaltung bezeichnet. Dieses Phänomen ist häufig das Ergebnis von Prioritätenänderungen zugunsten anderer Lebensbereiche der involvierten Entwickler:innen. Die Entwicklungsarbeit stagniert und eintreffende Hilfsangebote werden nicht mehr bearbeitet, was einem Projektende gleichkommt. Da es sich allerdings um FLOSS handelt, wurde der Quellcode unter neuem Namen auf GitHub wieder veröffentlicht. Der Entwickler übernahm die Rolle des Maintainers und es baute sich eine neue Entwicklungsgemeinschaft um das Projekt herum auf.

7.3.2 FLOSS-Projekt in eine GNU/Linux-Distribution integrieren

GNU/Linux-Distributionen sind eine Zusammenstellung von aufeinander abgestimmter Software um den Linux-Kernel. Für die Verwaltung der Software auf dem jeweiligen Computer gibt es diverse Paketverwaltungen. Diese ermöglichen, neben der manuellen Paketverwaltung, die einfache Installation und Deinstallation von kompatibler Software. Um Software in eine Distribution zu integrieren braucht es eine Person, die die Software auf die speziellen Anforderungen der jeweiligen Distribution anpasst. Diese werden ebenfalls als Maintainer:innen oder präziser als Paket-Maintainer:innen bezeichnet. Sie kümmern sich vorrangig um das Funktionieren einer FLOSS in einer bestimmten Distribution. Sie sind nicht zwangsläufig an der Entwicklung des Originalpaketes beteiligt.

> „Da habe ich hauptsächlich angefangen Pakete zu maintainen, weil die halt nicht in Fedora[111] mit drinnen waren. Und ich wollte halt Paket XY haben und dann habe ich es halt gebaut.
> [Nachfrage durch den Interviewer] Für dich selbst zur Nutzung?

[111] Fedora ist eine auf Red Hat basierende GNU/Linux-Distribution.

> Ja, ja. Und das gleiche bei OpenSUSE[112]. Da war es so, ja das brauchte ich als Abhängigkeit[113] für mein Paket und dann habe ich es auch gebaut und submitted[114]. Also das war (..) ja purer Eigennutz, um das mal so salopp zu sagen.“ (Dokument I-12, Abs. 8 - 10)

Obwohl dieser Entwickler sein Engagement als Paket-Maintainer als Eigennutz deklariert, so ermöglicht er es vielen anderen Nutzer:innen von Fedora und OpenSUSE die durch ihn betreuten Softwarepakete zu nutzen. Ohne diese Möglichkeit der Installation via Paketverwaltung erfordert es fortgeschrittene Kenntnisse, um die Software manuell zu installieren. Diese Barriere wird durch die Verfügbarkeit der Software in der distributionseigenen Paketverwaltung abgebaut.

Folgendes Zitat eines Paket-Maintainers beschreibt die Anforderungen an diese Rolle bei der Softwareentwicklung:

> „Meistens führt es dazu, dass man auch in dem eigentlichen Programm dann zum einen die Tickets dort noch einmal eröffnet und dann, wenn man den Patch hat, den selber dann auch schreibt und dann im Upstream[115] bereitstellt und hofft, das Upstream den dann einspielt, dann muss man den selbst bei Arch[116]-Linux ja nicht mehr einspielen. Das geht dann fließend ineinander über, wenn man einmal Paket-Manager ist für so ein Paket, dann kommt zum einen auch eine gewisse Erwartung, man hat immer mit dem Paket zu tun, dass man auch das nach Upstream versucht zu kommunizieren und damit zu arbeiten.“ (Dokument I-8, Abs. 54)

Paket-Maintainer:innen haben eine Vermittlungsfunktion. Sie machen eine Software für eine konkrete Distribution verfügbar und dienen dadurch auch als Ansprechpartner:innen für die Nutzer:innen dieser Distribution. Sie leiten Fehlerberichte weiter und pflegen Softwareänderungen für die FLOSS-Version dieser Distribution ein. Teilweise sind sie auch selbst an der Entwicklungsarbeit im Ursprungsprojekt beteiligt, wie es dieser Entwickler beschreibt:

> „Na ursprünglich habe ja ja eher das Debian[117]-Zeug gemacht, also ich bin von dem Debian-Paket zu dem [FLOSS-Projekt] gekommen. So herum ist es eigentlich. Also [FLOSS-Projekt] war vorher schon in Debian drin, allerdings war da kein Maintainer mehr da und da habe ich mich dann (.) um das

[112] OpenSUSE ist eine auf Slackware basierende GNU/Linux-Distribution

[113] Eine Programmabhängigkeit verweist auf andere Programme oder Softwarebibliotheken, die für die Funktion eines Programms notwendig sind. Um eine Software in eine Paketverwaltung aufzunehmen, müssen auch alle Programmabhängigkeiten Teil der Paketverwaltung sein.

[114] Submitted ist eine Wortentlehnung vom englischen *submit* und bezeichnet das Einreichen von Änderungen über eine Versionsverwaltung.

[115] Als 'Upstream' wird die Entwicklungsgemeinschaft beziehungsweise der:die Projektmaintainer:in vom Ursprungsprojekt bezeichnet.

[116] Bei Arch handelt es sich um eine GNU/Linux-Distribution

[117] Debian ist eine GNU/Linux-Distribution.

> Paket gekümmert, weil ich das selbst gebraucht habe. Und habe dann darüber durch die ersten Sachen, was wir aus Debian da mitgebracht haben, bin ich eigentlich dort an das Projekt rangekommen und habe dann halt gefragt [ob Unterstützung erwünscht ist]. Aber (.) naja, jetzt ist es im Grunde so, dass ich beides mache. Also ich kümmere mich sowohl um das Debian-Paket, als auch um [FLOSS-Projekt] selbst.“ (Dokument I-9, Abs. 42)

Die Arbeit als Paket-Maintainer:in kann auch der Ausgangspunkt zu einem Engagement beim Ursprungsprojekt sein beziehungsweise dazu führen, dass die Person sowohl Paket-Maintainer:in als auch Projektmaintainer:in ist. Dieser Prozess läuft oft auch in die gegenläufige Richtung ab: Ein:e Projektmaintainer:in ist gleichzeitig auch in der von ihr:ihm favorisierten GNU/Linux-Distribution als Paket-Maintainer:in aktiv, um die Zugangshürden zur Nutzung dieser FLOSS im Rahmen dieser Distribution für potentielle Nutzer:innen der FLOSS zu reduzieren.

7.3.3 Sicherheitslücken in FLOSS beseitigen

Auch der Wunsch nach einem sicheren und stabil laufenden Computersystem kann ein Auslöser für ein FLOSS-Engagement sein. Dies kann sich auf eine spezielle Software beziehen oder komplett davon losgelöst als genereller Anspruch an das Computersystem bestehen.

> „Ja also bei [FLOSS-Projekt] war das hauptsächlich so: Ich habe über, ich glaube, die OSS-Security-Mailinglist[118], da gab es ein paar eklige E-Mails von wegen: Ja diese Software ist auf all euren Rechnern installiert und die ist fürchterlich unsicher. Und der damalige Maintainer hatte nicht/ Ich weiß nicht, ob er das Problem überhaupt verstanden hat, was ihm da geschildert wurde. Naja und ich dachte mir halt: Irgendwer muss es halt reparieren. Und dann habe ich halt angefangen. [...] Also das [FLOSS-Projekt]-spezifische, weil eigentlich interessieren mich Bildmetadaten nicht die Bohne. Fotografie ist eigentlich überhaupt nicht meins und das verwenden anscheinend hauptsächlich Fotografen oder Leute, die ihre großen Bild-Libraries verteilen. Muss ich sagen, ist eigentlich überhaupt nicht meins. Genau, eigentlich hauptsächlich, weil das Teil auf meinem Rechner installiert ist, möchte ich da nicht so ein fürchterliches Security-Hole haben.“ (Dokument I-12, Abs. 34)

Der Entwickler schildert sein Engagement in dem FLOSS-Projekt, obwohl er an der eigentlichen Nutzung dieser FLOSS kein Interesse hat. Daraus entwickelte sich ein langfristiges Engagement, bei dem er kontinuierlich Fehlerbehebungen einpflegte, die auch von anderen beigesteuert werden. Schon die Auseinandersetzung mit sicherheitsrelevanten Themen über eine diesbezügliche Mailingliste verdeutlicht diesen Anspruch an ein

[118] Die open-source-software-security Mailingliste ist eine offene Mailingliste rund um Sicherheitsprobleme und Lösungsstrategien von FLOSS.

sicheres Computersystem, welches unabhängig von einer bestimmten Software für diese Person relevant ist.

7.3.4 Neues FLOSS-Projekt initiieren

Neben der Partizipation an einem bestehenden Projekt ist auch die Gründung eines neuen FLOSS-Projektes ein Auslöser für ein FLOSS-Engagement. Der Projektinitiator, der im Folgenden zitiert wird, schildert den Verlauf, wie sich aus einer Idee heraus ein FLOSS-Projekt mit einer kleinen Entwickler:innengemeinschaft aufbaute:

> „Ich hatte irgendwie keinen Bildbetrachter, der mich komplett befriedigt. Und dann habe ich mir gedacht: Ja, wie witzig wäre es denn, dass mal so selber zu schreiben. Und dann habe ich es natürlich im ganz kleinen Rahmen selber mal probiert. Dann ging das einigermaßen und dann habe ich es auf GitHub geschoben und irgendwann kamen dann tatsächlich Leute, die das auch verwendet haben. Und dann ist es auch einfach gewachsen. Ja und seit dem/ Die Motivation war einfach: Es gibt nichts, was mich komplett befriedigt und dann ist mir aufgefallen, es gibt tatsächlich mehr Leute, denen es genauso geht und die jetzt irgendwie mein Projekt auch cool finden.“ (Dokument I-11, Abs. 16)

Die interviewte Person berichtet von ihrer Neugier nach kleinen FLOSS-Projekten in der Startphase. Sie schaut sie sich an, installiert sie probeweise und steuert gegebenenfalls eigene Lösungen für Programmfehler bei. Mit ihrem FLOSS-Projekt ist sie ähnlich verfahren. Sie hat nach der Fertigstellung einer ersten lauffähigen Version in diversen Foren darüber berichtet und dadurch das Interesse anderer Entwickler:innen auf das Projekt gelenkt. Die ersten Entwickler:innen, die eigene Beiträge für den Projektfortschritt beisteuerten, waren Paket-Maintainer:innen, die diesen Bildbetrachter für einzelne GNU/Linux-Distributionen anpassten.

7.3.5 Unterstützung von Personen im sozialen Umfeld

Einige Befragte berichteten auch von Auslösern, die dem sozialen Umfeld von Entwickler:innen zuzuschreiben sind. Damit verbunden sind auch altruistische Überzeugungen, die in Kapitel 7.2 ausgiebig erläutert wurden. Hier sollen sie als Auslöser beschrieben werden, durch die die Entwickler:innen zum ersten Mal an einem FLOSS-Projekt partizipierten:

> „Also bei [FLOSS-Projekt] ist es ganz einfach, ich habe einfach meinem Freund geholfen (..) ja genau, also ich wollte halt, dass das, was er macht Erfolg hat und das (.) hat mir auch ein bisschen Spaß gemacht. Ich wollte auch ein bisschen was lernen dabei. Ich habe halt dann dabei meine erste Webseite quasi von Null programmiert. Und das war schon, ja, also hauptsächlich Lerneffekt und um ihm zu helfen.“ (Dokument I-3; Abs. 17)

Hier ist die treibende Kraft für das FLOSS-Engagement die partnerschaftliche Hilfsbereitschaft. Die Entwicklerin übernahm die Pflege der Webseite, die Übersetzung und eignete sich die Programmiersprache *PHP* für den Aufbau der Webseite an. In einem andere Beispiel ergab sich das initiale Engagement bei einem bestimmten FLOSS-Projekt aus einer Bitte, die einem Freund zuliebe erfüllt wurde:

> „Da war ich im Prinzip auf den Chemnitzer Linux-Tagen[119] unterwegs und auf dem Rückweg habe ich dann [Name des Freundes] mitgenommen und er hielt mir irgendwann das Notebook so unter die Nase: Kannst du mal darüberschauen, du kennst dich ja so ein bisschen mit Code-Review[120] aus, was sagst du zu dem und dem Quelltext und der Stelle da drinnen? Und ich hatte halt raufgeguckt und gleich binnen kürzester Zeit mehrere Vorschläge, was man da eigentlich komplett anders machen müsste. (.) Das ist der/ Und irgendwann waren wir halt zur Software-Telko[121] eingeladen, die es halt von dem Entwicklerteam regelmäßig mit gab. Und ja, von einem kam das nächste und irgendwann erst normaler Entwickler und irgendwann dann auch die Teamleitung dort übernommen gehabt.“ (Dokument I-6; Abs. 16)

Beide Beispiele zeigen, wie stark FLOSS ein gemeinschaftliches Produkt ist. Dies wird bei der Betrachtung der kooperativen und kollaborativen Arbeitsweise im nächsten Kapitel noch einmal verdeutlicht werden. Ebenso sind die geschilderten Auslöser ein Indikator dafür, wie eng teilweise die sozialen Bindungen zwischen den Entwickler:innen in einer Entwicklungsgemeinschaft sind. Obwohl die Kommunikations- und Arbeitsstrukturen internetbasiert sind, bedeutet dies nicht notwendiger Weise, dass dies auch für die sozialen Beziehungen gilt.

7.3.6 FLOSS als Lerngegenstand in Bildungsinstitutionen und im beruflichen Kontext

Nicht zu vernachlässigen ist auch der Fakt, dass über Bildungsinstitutionen und durch berufliche Anforderungen ein FLOSS-Engagement initiiert werden kann. Beispielsweise beinhalten viele Informatikstudiengänge ein Softwarepraktikum, dass die selbständige Softwareentwicklung im Team zum Ziel hat. Auf diesem Weg engagieren sich Studierende auch in FLOSS-Projekten beziehungsweise initiieren eigene Projekte. Bei dem qBittorrent-Projekt war Dominique eine Person, die eine Komponente der Software zusammen mit anderen Kommiliton:innen im Rahmen des Studiums implementiert hatte. In

[119] Eine jährlich stattfindende, mehrtägige Konferenz mit einem breiten Themen- und Workshop-Angebot rund um GNU/Linux. Ein Teil der Interviews wurde mit Besucher:innen auf solchen Konferenzen geführt.

[120] Ein Code-Review ist ein Qualitätssicherungsverfahren, bei dem die Software systematisch auf Fehler untersucht wird. Ziel ist die Verbesserung der Qualität von Software. Der Quellcode wird dabei durch eine weitere Person geprüft.

[121] Die Software-Telko war eine regelmäßige Telefonkonferenz aller Verantwortlichen vom Softwareteam.

den Interviews fanden sich zwei weitere Personen, die einen solchen Auslöser beschrieben. Eine schilderte, wie sie zehn bis 15 Studierende betreute, die eine gemeinschaftliche Umsetzung eines FLOSS-Projektes realisierten und dies mit der Anerkennung von Studienleistungen durch die Fakultät anerkannt wurde.

Auch im beruflichen Kontext spielt FLOSS ein Rolle. Dieser Aspekt wird noch einmal ausführlich bei der FLOSS-Entwicklung als Lohnarbeit beziehungsweise als Reputation im beruflichen Kontext im Abschnitt 7.4.3 dieses Kapitels behandelt. Als Auslöser für ein FLOSS-Engagement spielt dies insofern eine Rolle, als das Unternehmen und andere Organisationen selbst auch FLOSS einsetzen und diese den eigenen Anforderungen entsprechend anpassen.

7.4 Beweggründe von erfahrenen FLOSS-Entwickler:innen

Neben den Beweggründen, die für das erste FLOSS-Engagement ausschlaggebend waren, können sich diese Interessen im Rahmen einer längerfristigen Mitarbeit in einer Entwicklungsgemeinschaft verändern oder durch andere Interessen ergänzt werden. Dahinter verbergen sich immer auch Werte und Hoffnungen, die sich im zeitlichen Verlauf ändern können. In diesem Abschnitt soll aber der Fokus auf die Änderung von Beweggründen im Vergleich zum ersten FLOSS-Engagement gesetzt werden.

7.4.1 Freude am Lernen

Die Arbeit in einem FLOSS-Projekt wird von Vielen als freudvolle Tätigkeit beschrieben. Hintergrund dabei ist ein Interesse an Funktionszusammenhängen zwischen Hard- und Software. Um Fehler beheben zu können, muss das Problem und die Funktionsweise der Software verstanden werden. Dieser Prozess kann einen Großteil der Entwicklungsarbeit ausmachen. Es ist ein Kreislauf aus Recherche, Implementation und Testen, der immer wieder bis zu einer zufriedenstellenden Lösung durchlaufen werden muss. Dies erfordert Frustrationstoleranz und Ausdauer, die durch die Freude an problemzentrierten Lernen aufrecht erhalten werden:

> „[...] also für viele und auch für mich teilweise ist es halt ja auch einfach ein Hobby und du willst wie gesagt ja auch Spaß haben, du willst dir diese Technik erkunden/ [...]" (Dokument I-2; Abs. 20)

Die Freude am Lernen ist aber nicht nur auf rein technische Aspekte begrenzt. Insbesondere der Austausch mit anderen Projektmitgliedern ist für einige Entwickler:innen für ihr Engagement ausschlaggebend:

> „Also bei [FLOSS-Projekt] [...] hat es mir auch ein bisschen Spaß gemacht. Ich wollte auch ein bisschen was lernen dabei, ich habe halt dann dabei meine erste Webseite quasi von Null programmiert. Und das war schon, ja, also hauptsächlich der Lerneffekt und um ihm zu helfen. [...] [Bei einem anderen

> Gemeinschaftsprojekt:] Aber für mich ist es hauptsächlich, ja, es macht halt Spaß mit den Leuten zu arbeiten, es ist/ (.) es macht Spaß, das was ich da tue im Projekt. Ja, ich lerne halt auch viel. Ich habe viele Gelegenheiten Sachen auszuprobieren und Dinge zu testen, ja." (Dokument I-3; Abs. 17 - 25)

Viele Entwickler:innen gaben an, dass Lernen eines ihrer Hauptinteressen ist. Lernen ist jedoch keine ziellos ausgeübte Tätigkeit. Es folgt konkreten Erwartungshaltungen an die zu benutzende Software oder ist eingebettet in gemeinsame Gruppenziele und die gemeinschaftlichen Praktiken und sozialen Beziehungen.

7.4.2 Erwartungen durch Andere

Im Rahmen eines längerfristigen Engagements kommt es zur Übernahme bestimmter Rollen, die jeweils mit bestimmten Aufgaben verknüpft sind. Mit der Zeit etabliert sich eine feste Aufgabenteilung, die mit einer Erwartungshaltung an die Entwickler:innen einhergeht:

> „Naja dadurch, dass ich halt in manchen Bereichen tatsächlich richtig Verantwortung übernommen habe, kann ich mich irgendwie auch schon gar nicht rausschleichen und sagen: Nee, mache ich nicht. Das ist ja auch kein schlechtes Gefühl, da Verantwortung zu haben." (Dokument I-3; Abs. 88)

Die Entwicklerin ist in Schlüsselpositionen der Entwicklungsgemeinschaft in einem größeren FLOSS-Projekt vorgerückt. Sie organisiert das Schreiben eines Handbuchs, die Übersetzung der FLOSS in mehrere Sprachen und hilft Nutzer:innen bei ihren technischen Problemen, die spezifische Fachkenntnisse aus der Entwicklungsgemeinschaft erfordern. Sie begründet ihr Engagement unter anderem damit, dass ihr Rückzug aus dem FLOSS-Projekt eine gravierende Belastung für die Entwicklungstätigkeit darstellen würde. Um der Erwartungshaltung der anderen Entwickler:innen sowie der Nutzer:innen zu entsprechen, will sie sich auch weiterhin in diesen Aufgabenbereichen engagieren.

Aber auch außerhalb einer konkreten Entwicklungsgemeinschaft kann sich das Engagement aus einer allgemeinen Rollenerwartung heraus manifestieren:

> „Das geht dann fließend ineinander über, wenn man einmal Paket-Manager ist für so ein Paket, dann kommt zum einen auch eine gewisse Erwartung, man hat immer mit dem Paket zu tun, dass man das auch nach Upstream versucht zu kommunizieren und da mitzuarbeiten." (Dokument I-8; Abs. 54)

Als Paket-Manager ist die eben zitierte Person allein für die Aktualisierung der durch sie betreuten Software in der Distribution zuständig. In diesem Sinne ist sie nicht direkt Teil einer Entwicklungsgemeinschaft, leitet aber Fehlerberichte an die ursprüngliche Entwicklungsgemeinschaft der Software weiter. Wenn sie doch einmal einen Patch beisteuert,

nimmt sie nur sporadisch an der Entwicklungsarbeit teil. Dennoch spürt sie eine Erwartungshaltung an ihre Person, die sich aus der Rolle als Paket-Maintainer ergibt: Sie ist verantwortlich für die Aktualisierung eines Softwarepakets im Rahmen dieser GNU/Linux-Distribution, weil dies die Rollenerwartung an Paket-Maintainer:innen ist. So lange sich keine andere Person für diese Aufgabe findet, ist sie dafür verantwortlich oder das Paket veraltet und wird aus der Distribution entfernt.

7.4.3 FLOSS als Lohnarbeit

Wie schon die Studien von Ghosh (2007), Hertel et al. (2003) oder Lakhani und Wolf (2003) zeigten, werden viele Entwickler:innen für ihr Engagement in FLOSS-Projekten direkt bezahlt oder können einen Teil ihrer bezahlten Arbeitszeit für die FLOSS-Entwicklung nutzen. Insofern existieren für einen nicht unbedeutenden Teil der Entwickler:innen auch monetäre Anreize für das Engagement. Dieses Anreizsystem kann sich aber auch erst mit der Zeit etablieren, wie es dieser Entwickler beschreibt:

> „Na, das war früher tatsächlich auch nur Freizeit, wobei ich mittlerweile auch/ Also es gibt da eine Firma zu dem Projekt, die dann halt professionellen Support verkaufen an irgendwelche größeren Firmen und da bin ich mittlerweile angestellt als Werksstudent. Und mache das dann halt auch tatsächlich einfach bezahlt diesen Kalender. Früher war es tatsächlich nur Freizeit. Es war halt einfach wirklich/ Ich brauchte halt diese Anwendung weil ich wollte persönlich die Alternative haben. Es ist also eigentlich quasi was, (.) naja, fast Egoistisches, weil ich wollte halt/ Ich brauchte halt diese Software und deshalb habe ich sie einfach geschrieben. [...] also ich bin da in einer sehr glücklichen Position, dass ich quasi bezahlt werde, aber ich kann meine eigene Roadmap aufstellen. Also ich kann halt entscheiden, was ich am Kalender machen will. Also es ist klar, wenn halt irgendein Bug-Report von einem Kunden kommt, dann hat der natürlich Priorität. Aber das kommt auch eher seltener vor, das heißt, wenn halt gerade nichts ansteht, dann entscheide ich halt, was ich implementieren will und wann ich irgendwelche Releases mache und so weiter.“ (Dokument I-10; Abs. 55 und 107)

Aus einem persönlich empfundenen Mangel an einem spezifischen Softwaremodul hat dieser Entwickler in seiner Freizeit dieses fehlende Modul geschrieben. Durch einen Zufall konnte er seine Software einem der Kernentwickler dieses FLOSS-Projektes bei einem Vortrag präsentieren und wurde daraufhin für die weitere Entwicklung dieses Moduls bezahlt.

Aber auch das Engagement bei nicht profitorientierten FLOSS-Projekten beziehungsweise bei FLOSS, auf deren Basis Firmen Dienstleistungen anbieten, kann für Entwickler:innen die Perspektive auf eine Lohnarbeit eröffnen. Viele Personalagenturen nutzen auch FLOSS-Portale, um für ihre Kund:innen Arbeitskräfte zu rekrutieren. Dabei dienen FLOSS-Portale auch der Selbstdarstellung von Entwickler:innen:

> „Na, man wird ab und zu mal angeschrieben, zum Beispiel wenn man bei GitHub ist. Ich habe auch bei GitHub einige Projekte drin, unter anderem wurde ich von Google angeschrieben und habe ein Jobinterview gemacht deswegen. Hat nicht geklappt aber deswegen habe ich zwei Telefoninterviews gehabt. Einfach nur, weil sie bei GitHub Leute gesucht haben, die irgendwie in die Richtung Python irgendwas gemacht haben. (..) Ja, kommt durchaus vor. (...) Und für meinen Job, den ich jetzt mache, hat es in der Bewerbung auch nicht geschadet, dass ich gesagt habe: Ich habe Ahnung von vielen Projekten und habe auch Softwareentwicklung schon gemacht. Und praktisch geholfen hat es mir auf jeden Fall sehr viel, bei Softwareprojekten zu arbeiten.“ (Dokument I-8; Abs. 103)

Die Portale dokumentieren die Entwicklungsaktivitäten und lassen damit Rückschlüsse auf die Expertise der Entwickler:innen zu. Zusätzlich können sich die Entwickler:innen mit ihren Vorlieben und Kenntnissen selbst darstellen, sich in Gruppe organisieren und sich untereinander vernetzen. Demzufolge können die Aktivitäten auf FLOSS-Portalen auch für die Selbstvermarktung der Entwickler:innen dienlich sein. Sie können sich dadurch potentiellen Arbeitgeber:innen präsentieren.

Nichtsdestotrotz ist aber auch die gewonnene Expertise durch die Softwareentwicklung relevant, wenn kein neues Arbeitsverhältnis angestrebt wird:

> „Und es ist auch für die eigene Reputation, denke ich, ganz gut. Das war jetzt nicht irgendwie mein (.) ursprüngliches Ziel, aber ich merke es halt immer wieder, dass es auch gerade im Berufsleben, dass es ziemlich anerkannt wird. Also wenn man da engagiert ist und (.) auch Expertise auf Gebieten hat, die eben viele andere nicht haben, das hilft eigentlich für die eigene Reputation. [...] Also das habe ich einfach durch Diskussionen mit Leuten mitbekommen. Und/ Sind Themen, mit denen ich mich früher nicht beschäftigt habe, aber die mir jetzt auch im Berufsleben sehr helfen. Also auch Expertise, die ich sonst einfach nicht gekriegt hätte, weil so eine/ Das man eine Umgebung, wie jetzt bei [FLOSS-Projekt], wo dann mehrere hundert Server sind, mit denen man dann zu tun hat. Also das hat man halt so im Privatbereich eher nicht.“ (Dokument I-9; Abs. 21 und 153)

Dieser Entwickler hebt die von anderen anerkannte Reputation innerhalb des Kreises der Kolleg:innen hervor, die er sich durch sein Engagement aufgebaut hat. In diesem Fall ist es sein Fachwissen im Bereich der Serveradministration in einer komplexen Arbeitsumgebung, mit denen nur wenige Kolleg:innen Erfahrung haben und deswegen seine Expertise geschätzt wird. Die FLOSS-Entwicklung leistet dieser Entwickler vollständig ehrenamtlich, profitiert aber dennoch beruflich von dem Nischenwissen und der daraus hervorgehenden Reputation.

7.4.4 Anerkennung

Viele Befragte betonen den individuellen Sinnbezug für die eigene Arbeit. Je mehr die selbst geschriebene Software später von anderen verwendet wird, desto höher wird der Nutzen für die eigenen Mühen eingeschätzt. Damit verknüpft ist auch die Sichtbarkeit des eigenen Handelns nach außen und damit auch die Darstellung der eigenen Leistung:

> „Ich denke hauptsächlich auch, weil man sich irgendwie besser fühlt, dass, wenn man etwas gemacht hat und es da mehrere Leute benutzen können. Es ist nicht nur für einen selber, wenn man verdammt viel Arbeit reingesteckt hat. Meist über Nacht irgendwie rumgefummelt und et cetera, ganz viel ausprobiert [hat]. Man fühlt sich einfach auch besser, wenn man natürlich das dann beisteuern kann und andere es auch benutzen können und man die Zeit nicht nur einfach so weggedaddelt hat." (Dokument I-8; Abs. 22)

Ein anderer Entwickler freut sich über die Verbreitung seiner Software, weil sie von anderen großen FLOSS-Projekten als Softwarebibliothek verwendet wird und sie deshalb einen großen Kreis von Nutzer:innen hat:

> „Und es ist durchaus auch eine Sache, wenn man sagen kann: Ich habe Software geschrieben, die läuft halt auf Servern, die von Millionen von Leuten halt genutzt werden. Ist durchaus mal ein leicht anderer Standpunkt als: Ja, ich habe ein 08/15-Programm geschrieben, was außer mir noch keiner gesehen hat. Ist durchaus eine andere Dimension und durchaus etwas, was halt auch ein bisschen stolz macht." (Dokument I-6; Abs. 148)

Der Stolz auf die eigene Leistung leitet sich aber nicht nur aus der anonymen Zahl der Nutzer:innen ab. Ebenso zeigen Beteiligungsangebote beziehungsweise konkrete Hilfe bei der Entwicklung den Entwickler:innen das Interesse an dem FLOSS-Projekt und der Fortsetzung der Entwicklungsarbeit:

> „Damals war das halt Einzelkämpfertum und einfach so machen und veröffentlichen und sich freuen, wenn sich einer mal zurückmeldet und sagt: Da gibt es ein Problem! Das ist dann immer schön. (.) Jetzt ist es halt mehr/ Ich weiß nicht ob es an der Software liegt oder der Art, wie sich so Ökosysteme gestalten, halt mehr Umgang mit Menschen, die was einreichen und weniger/ Also weniger selbst erschaffende Entwicklung, sondern eher reaktive Entwicklung." (Dokument I-4; Abs. 88)

Hier wird die Veränderung der Entwicklungsarbeit im zeitlichen Verlauf beschrieben. Die Person hat die Entwicklung als alleiniger Entwickler an einer FLOSS wieder aufgenommen, nachdem sie vom vorherigen Projektmaintainer nicht mehr weiterentwickelt worden war. Erst mit der Zeit gesellten sich weitere Entwickler:innen dazu und es bildete sich wieder eine Entwicklungsgemeinschaft heraus.

Der Prozess der ersten Beteiligungsangebote an einer zuvor als Einzelperson betriebenen Entwicklung wurde von vielen Personen als wichtiges Schlüsselerlebnisse benannt. Auch das Einreichen von Fehlerberichten stellt schon eine Anerkennung dar, da es zum einen den Testaufwand für die Entwickler:innen reduziert und zum anderen ein besonderes Interesse der Nutzer:innen signalisiert, da sie trotz des Aufwands zur Einreichung eines Fehlerberichtes diese Rückmeldung auf sich nehmen.

Anerkennung durch relevante Andere

Die Quellen von Anerkennung lassen sich noch einmal differenzieren nach der Relevanz der Personen, die eine jeweilige Leistung anerkennen. Als *relevante Andere* werden die Personen bezeichnet, die als Vorbilder für das eigene Engagement dienen können und damit die Art und Weise des Engagements von Entwickler:innen maßgeblich beeinflussen können.

> „Und dann bin ich, es war halt so Mitte des Jahres und dann bin ich irgendwann später so in der zweiten Hälfte des Jahres zu einem Vortrag von dem Projektleiter gegangen. Der war in Berlin und einfach mal angequatscht und ich meinte: Hey, ich habe übrigens ein Interface für den Kalender geschrieben.
> [Nachfrage durch den Interviewer] Und den hattest du aber vorher schon veröffentlicht?
> [Antwort] Nee, noch gar nicht. Das war bis dahin noch alles bei mir selbst. Weil ich vorher noch keine Ahnung von Git oder irgendwelchen Versionierungssystemen hatte, das war wirklich alles ganz neu für mich. Und dann hat er mich am nächsten Tag direkt eingeladen zu so einer Hackweek[122] und dann bin ich da halt hin und dann haben wir so ein bisschen den Code auch eingepflegt im eigentlichen Projekt und so bin ich mehr oder weniger Teil davon geworden.“ (Dokument I-10; Abs. 27 - 31)

Diese Begegnung mit dem Projektmaintainer bei einem Vortrag führte zu einem langfristigen Engagement dieses Entwicklers in diesem FLOSS-Projekt, für welches er im späteren Verlauf auch bezahlt wurde.

Ähnlich erging es dem nachfolgend zitierten Entwickler, der ebenfalls durch ein direktes Zusammentreffen mit anderen Entwickler:innen in den inneren Kreis der Entwicklungsgemeinschaft vorrückte:

> „Also, ich muss sagen, ich war 2009 in Berlin auf dem Linux-Tag, da habe ich das erste Mal richtig persönlich auch Kontakt zu vielen Leuten aus der Debian-Community gehabt. Das hat mich damals dazu gebracht, eben dort

[122] Die Hackweeks in diesem FLOSS-Projekt sind einwöchige Treffen, an dem die Entwickler:innen zusammenkommen und gemeinschaftlich vorher festgelegte Probleme an der FLOSS bearbeiten.

mich weiter zu engagieren. Und (.) dann eben auch die DebConf[123] und die Chemnitzer Linux-Tage. Das sind eigentlich schon so Events, wo man sich auch mal mit den Leuten trifft und (.) dann auch mal so Ideen spinnt, was könnte man denn da mal noch weiter machen. (..) Da ist es oft so, dass ich mir dort irgendwelche neuen Ideen mitnehme. Was kann ich machen? Wo kann ich mich noch weiter engagieren? Zum Beispiel bei der Debconf 2013 bin ich (.) in Debian auch in dieses Team reingekommen, was sich darum kümmert, neue Leute in das Projekt aufzunehmen und da gibt es so einen Bewerbungsprozess in Debian und den mit zu organisieren. Genau, das ist eigentlich nur dadurch gekommen, das ich eben auf dieser Konferenz war und mich da die Leute, die das gemacht haben, direkt angesprochen haben, weil ich da so ein Tool gebaut habe, mit dem man so ein bisschen die Aktivitäten der Leute rauskriegt (.) und solche Sachen. Auch so die Sachen, gerade die DebConf 2013 und eben 2009 da in Berlin der Linux-Tag, da (.) habe ich wirklich eigene Motivation mehr gehabt, da wieder mich stärker zu engagieren. Vorher habe ich mich halt primär nur um meine paar wenigen eigenen Packages gekümmert, wo ich irgendetwas damit gemacht habe und da ist schon mehr auch dieser Aspekt gekommen, mich dort in der Community zu engagieren.“ (Dokument I-9, Abs. 106)

Beide Entwickler berichten von richtungsweisenden Begegnungen mit relevanten Anderen, die auf Grundlage ihrer zuvor erbrachten Leistungen möglich wurden. Die Anerkennung dieser Leistungen mündete in die Einladung in den inneren Kreis der Entwicklungsgemeinschaft. Diese Prozesse können natürlich auch innerhalb einer Entwicklungsgemeinschaft selbst ablaufen, wenn Entwickler:innen von der Peripherie ins Zentrum vorrücken. Dies entspricht dem Übergang von äußeren zu den inneren Schalen im Schalenmodell von Nakakoji et al. (2003).

7.4.5 Interesse am Thema

Das Interesse an einer bestimmten FLOSS ist einer der wichtigsten Faktoren für die Entscheidung über eine Mitarbeit. Das FLOSS-Projekt muss in irgendeiner Art und Weise für die Entwickler:innen von Relevanz sein. Das Interesse kann dabei in zwei Varianten unterschieden werden: Zum einen das pragmatische Interesse und zum anderen das thematische Interesse.

Das pragmatische Interesse wurde bereits bei den Auslösern für ein FLOSS-Engagement beschrieben. Meist handelt es sich um eine FLOSS, die als tägliches Werkzeug von den Entwickler:innen genutzt wird und nicht wie erwartet funktioniert. Daraufhin wird sich an der Fehlerbehebung beteiligt, um die FLOSS wieder in der gewünschten Weise benutzen zu können.Das thematische Interesse hingegen fokussiert sich nicht nur auf das

[123] Die DebConf ist eine jährlich stattfindende Konferenz, auf der sich die Entwickler:innen der GNU/Linux-Distribution Debian treffen und die weitere Entwicklungsrichtung diskutieren.

Werkzeug, sondern auch auf das Themengebiet im Allgemeinen, in dessen Umfeld die FLOSS angesiedelt ist. Ziel ist eine tiefe beziehungsweise breite Auseinandersetzung mit der Thematik. Die beiden folgenden Zitate stammen von zwei Entwicklern, die beide im gleichen FLOSS-Projekt aktiv sind:

> „[...] ich fand die Idee selber an Netzwerkinfrastruktur zu arbeiten und diese Probleme, die man da angeht, halt erst einmal recht interessant. Ich bin halt Diplominformatiker und habe mich viel mit Kombinatorik beschäftigt und da sind solche Netzprobleme halt quasi eine ganz natürliche Erweiterung vom eigenen Feld.“ (Dokument I-7; Abs. 12)

Das Interesse von diesem Entwickler ist stark theoretischer Natur und entstammt aus dem Kontext seines Studiums. Er arbeitet an den Grundlagen für neue Kommunikationsprotokolle für sich spontan vernetzende Computer und implementiert diese in Testnetzwerken, die als zukünftige Erweiterung in das FLOSS-Projekt aufgenommen werden sollen.

Ein anderer Entwickler betont die Möglichkeiten der sozialen Teilhabe, die durch dieses Projekt ermöglicht wird. Seine Betätigungsfelder waren die Softwareentwicklung für die Endgeräte und deren Installation für den praktischen Einsatz. Er verbindet das technische Interesse mit den Potentialen, die das FLOSS-Projekt für die Nutzer:innen bietet:

> „Genau, aber währenddessen habe ich mich dann halt auch schon die ganze Zeit, bevor ich angefangen habe zu studieren, interessiert für [FLOSS-Projekt]. Fand ich eine interessante Idee, wie man halt so Internet teilt. Wie kann man Internetanschlüsse teilen, kann man gemeinsam ein Netzwerk aufbauen und Zugang zu Informationen geben. [...] Aber (.) also für viele und auch für mich teilweise ist es halt ja auch einfach ein Hobby und du willst, wie gesagt, ja auch Spaß haben, du willst dir diese Technik erkunden/“ (Dokument I-2; Abs. 4 und 20)

Der zitierte Entwickler arbeitet mit unterschiedlichen Routern, die er in das Netzwerk integriert. Das Interesse bezieht sich stark auf die Hardware, die durch das FLOSS-Projekt unterstützt wird. Beide Entwickler haben Informatik studiert und bringen damit auch viele formal erworbenen Kompetenzen in das FLOSS-Projekt mit ein. Beide verbinden soziale Beweggründe und thematisches Interesse in ihrem FLOSS-Engagement. Für sie ist das FLOSS-Projekt nicht nur ein Instrument, sondern auch Gegenstand ihres Interesses.

7.4.6 Austausch mit Gleichgesinnten

Die soziale Einbettung des eigenen Engagements ist für fast alle befragten Entwickler:innen ein wichtiger Punkt. Der Austausch mit Gleichgesinnten erfüllt verschiedene Funktionen, die anhand der Interviews in diese drei Kategorien differenziert wurden: fachlicher Austausch, Kommunikation über gemeinsame Interessen und die Pflege freundschaftlicher Beziehungen, die im Rahmen der gemeinsamen Entwicklungsarbeit entstanden sind.

Die fachliche Kommunikation mit den Nutzer:innen der FLOSS dient der Fehlermeldung und der Kommunikation von Verbesserungsvorschlägen. In größeren Softwareprojekten gibt es eigene Kommunikationskanäle, in denen Nutzer:innen von Entwickler:innen oder erfahrenen Nutzer:innen bei Anwendungsproblemen Unterstützung finden. Daneben gibt es auch verschiedene Kommunikationskanäle zwischen den Entwickler:innen. Diese dienen der Absprache über Aufgabenbereiche oder Aufforderungen, neuen Quellcode zu bewerten beziehungsweise Hilfestellung bei Problemen zu geben:

> „Ansonsten (.) beim Support ist es hauptsächlich, ja (.) das positive Feedback halt der Benutzer, die sich freuen einfach, wenn irgendwas wieder klappt oder wenn sie so ein Aha-Erlebnis haben und sagen: Oh toll. Oder manche posten dann auch, was sie Tolles mit [FLOSS-Projekt] hergestellt haben oder gebastelt haben oder weiß der Teufel. Das ist halt auch spannend zu sehen, was die alles künstlerisch-kreatives da zusammenbauen. (..) Ansonsten, ja, was die Geschichten angeht, wo ich mit den Entwicklern zusammenarbeite, also was weiß ich, es gibt halt einen, der mich häufiger mal fragt irgendwie, ob ich irgendwas Neues, was er gerade gemacht hat, testen könnte oder so. Das ist einfach/ Weil, ja (.) weil es einfach Spaß macht, mit dem zusammenzuarbeiten und zu sehen, wie da irgendwas Schönes bei entsteht oder irgendeine tolle Funktion entsteht, die dann hinterher/ Wo man schon weiß, was den Leuten Spaß macht oder so, das einfach cool ist. [...] Es gibt zum Beispiel einen Mitübersetzer, mit dem führe ich dann detaillierte Diskussionen über irgendwelche winzigen Formulierungsabstufungen und das macht/ Ich glaube für 90% aller Leute wäre es total nervig, aber wir beide haben Spaß dabei. Oder (..) ja alleine möchte ich daran nicht arbeiten, das ist einfach öde." (Dokument I-3; Abs. 88 und 161)

Diese Person ist sowohl im Support aktiv und berät die Nutzenden einer Grafiksoftware bei technischen Problemen und ist gleichzeitig im engen Kontakt mit Entwickler:innen. Sie kommuniziert also innerhalb wie auch außerhalb der Entwicklungsgemeinschaft und vernetzt gegebenenfalls die Hilfesuchenden mit den entsprechenden Entwickler:innen der FLOSS-Gemeinschaft. Der Austausch mit anderen ist für sie eines der wichtigsten Beweggründe für ihr Engagement.

Ähnlich dazu beschreibt die nachfolgend zitierte Person die Funktion der Community aus ihrer Perspektive:

> „Ja, ich mag das soziale Gefüge da herum. Das ist halt nicht nur/ Ich glaube/ Ich denke, wenn Leute diese Projekte von außen sehen, dann sehen sie halt meistens nur ein Produkt (.) und was für mich halt viel mehr ausmacht, sind halt die Gemeinschaften um die Projekte herum. Die Leute treibt ja meist irgendwas, das ist meistens halt eine Idee und die versuchen sie halt mit dem Projekt irgendwie zu lösen. Und zwar nicht für sich im stillen Kämmerchen,

> sondern gezielt in der freien Fläche. [...] Na, die Leute sind halt meistens in irgendeiner Art und Weise sehr motiviert in ihren Projekten und reißen einen dann mit. Und man fühlt sich dann schon irgendwo in so einer Gemeinschaft aufgenommen, das ist schon so/ Das sind schon so soziale Bindungen wie man sie teilweise in Familien wiedererkennen würde. Nicht unbedingt jetzt, dass man sein Privatleben dort teilen würde, aber das hat halt ähnliche Bindungseffekte, finde ich immer. Also man wächst da so ein bisschen zusammen und (.) so richtig Schlüsselmomente dazu (..)/ Ja es gibt halt immer wieder diese Momente wo man mit irgendeinem Problem da steht und eigentlich irgendwie sich selber damit rumschlagen würde und im nächsten Moment sitzen drei Leute neben dir, die versuchen deine Probleme zu lösen, weil sie es halt einfach mal interessant finden." (Dokument I-7; Abs. 76 - 80)

Die Entwicklungsarbeit ist nicht der alleinige Gegenstand für den Austausch: Die Projektgemeinschaft bildet auch einen eigenen Begegnungsraum, der über die reine Entwicklungsarbeit hinausgeht. Der Austausch über nicht entwicklungsrelevante Themen fördert die Gruppenkohäsion. Insbesondere, wenn Möglichkeiten geschaffen werden, dass die Projektbeteiligten sich im Rahmen einer Konferenz oder eines Arbeitstreffens im direkten Kontakt austauschen können. Diese Treffen der Projektgemeinschaft werden vielfach als wichtige Schlüsselerlebnisse beschrieben, die zu einer weiteren Annäherung an die Kerngruppe führen und auch mit der Übernahme neuer Aufgaben verbunden sein können:

> „Die Community dort war sehr interessant, weil es alles meist so Musikliebhaber waren und Leute, die generell irgendwie eine große Diskographie selber haben, ganz viele CD's und sehr interessiert in die Richtung waren und deswegen hat man auch Gemeinsamkeiten gehabt. Eine Community, die über das Projekt hinausgeht und das schweißt irgendwie auch die Community zusammen. [...] Es gibt jährlich den [FLOSS-Projekt]-Summit und der fand einmal auch in Berlin statt, da war ich auch dabei. [...] Wenn man den einmal in der Nähe hat, das Event, das schweißt natürlich auch zusammen, dass man die Leute alle tatsächlich einmal zusammen in einem Raum sieht, die man sonst nur international aus Foren kennt oder so. Das ist auf jeden Fall eine Sache, die zusammenschweißt, ja." (Dokument I-8; Abs. 79 - 83)

Neben dem jährlichen Treffen dieser Gemeinschaft, die sich um eine Musikdatenbank herum aufgebaut hat, gibt es auch einen regelmäßige Chat-Termin für die gemeinsame Koordinierung der Entwicklungsarbeit. Diese dienen aber auch anderen Zwecken, wie das folgenden Zitat belegt:

> „Die wöchentlichen Chats sind relativ strikt, da werden auch nur die Probleme, Lösungen und aktuelle Sachen besprochen, weil es einfach sonst mit

> einer Stunde sehr knapp bemessen ist. Es sind zwanzig Leute, die da beteiligt sind, meist so circa. Und wenn man da zu sehr abweicht/ Ja, (.) wird es schwierig. Der Chatraum ist ganzjährig offen und die Leute sind trotzdem da und reden auch über alle möglichen anderen Sachen, teilweise auch über Musik generell und ja/ (.) Führt nicht dazu, dass die Arbeit zwingend schneller passiert, aber dass man sich mehr auch ins Projekt reingezogen fühlt. Also dann führt es dazu, dass man irgendwie in diesem Chatraum/ Ich war teilweise wirklich andauernd in diesem Chatraum drin, dass man dabei ist und dann auch mitkriegt, wenn andere Sachen passieren [...]“ (Dokument I-8, Abs. 87)

Diese Person unterstreicht den Stellenwert der nicht-entwicklungsrelevanten Kommunikation innerhalb der Gemeinschaft, um neue Entwickler:innen an das FLOSS-Projekt zu binden. Gleichwohl sieht sie im späteren Interviewpassagen in Abgrenzung zu den beiden zuvor zitierten Befragten eine Community nicht als notwendige Voraussetzung für ein FLOSS-Projekt an und betont eher strukturelle Bedingungen wie gute Dokumentation und klare Strukturen, um Patches einzureichen und ein leicht verständlicher Programmierstil. Sie argumentiert eher utilitaristisch, während die beiden anderen die sozialen Prozesse in den Mittelpunkt ihres Engagements stellen. Nichtsdestotrotz schätzt auch diese Person die sozialen Austauschprozesse, was aus ihrem Interesse an dem FLOSS-unabhängigen Austausch mit anderen Entwickler:innen auf dem Chatkanal hervorgeht.

Eine andere Dimension von sozialen Austauschprozessen zeigt sich in den Interviewpassagen der folgenden Person:

> „Insbesondere internationale Projekte sind natürlich immer schön, weil, weiß ich nicht, man fühlt sich halt in einem größeren Kontext relativ grenzenfrei. Also man kann halt interessante Gespräche/ Weiß ich nicht, in [FLOSS-Projekt], in dem ich früher war, das waren halt Menschen aus Südafrika. Das war halt eine interessante kulturelle Austauschphase, so ein bisschen, wenn man mal auch so über nicht-technische Sachen gesprochen hat. Wenn man so ein bisschen in der Debian-Welt ist, da sind halt auch viele Leute, weiß ich nicht, aus Indien oder aus fernen Teilen dieser Welt, von denen man kulturell eher nichts hört und kriegt man dann ab und zu so ein bisschen mit, (.) was die Menschen halt so schreiben, persönliche Einblicke in fremde Leben, ferne Welten, das ist ganz hübsch. (..) Das Mitmachen in freien Softwareprojekten gibt mir halt ein großes Gefühl von kooperativer Schaffenskraft und der, sagen wir, moralischer Fragwürdigkeit von Dingen, die man nicht der Gemeinschaft zur Verfügung stellt.“ (Dokument I-4; Abs. 208)

Die territoriale Ausdehnung der Entwicklungsgemeinschaft ermöglicht auch die Wahrnehmung unterschiedlicher Perspektiven in Bezug auf interkulturellen Austausch. Auch in diesem Zitat deuten sich die beiden Ebenen im Austausch an: Kommunikation über die Entwicklungsarbeit und Austausch über Themengebiete, die außerhalb der FLOSS-Entwicklung liegen. Beide Ebenen sind untrennbare Teile von Austauschprozessen in

FLOSS-Projekten und es obliegt den jeweiligen Entwickler:innen, welche Bedeutung sie diesen Ebenen beimessen.

7.4.7 Erhalt von Software

Bei einigen Entwickler:innen zeigt sich ein besonderes Interesse am langfristigen Erhalt von FLOSS. Dieses Interesse verbindet die Wertorientierung zum Schutz von Kulturgütern und dem Auslöser, eine vorhandene FLOSS den eigenen Wünschen anzupassen. Da es sich hierbei in der Regel um die Wiederaufnahme der Entwicklungsarbeit nach einem Entwicklungsstopp handelt, werden höhere Anforderungen an das Engagement gestellt. Diese gehen weit über die reine Mitarbeit bei einer bereits bestehenden Entwicklungsgemeinschaft hinaus, weshalb dieses Interesse in dieser Forschungsarbeit als eigener Beweggrund für ein FLOSS-Engagement klassifiziert wurde. Das folgende Zitat verdeutlicht diesen Beweggrund:

> „Also im Groben, wie ein Projekt so langsam stirbt: Die Leute, die sich darum kümmern, stecken weniger Zeit rein, Bug-Reports laufen auf, auch Beteiligungsangebote laufen auf und werden halt nicht abgenommen und entgegengenommen und befürwortet und vorangetrieben. Ja und diese Rolle fehlte halt in dem Projekt dort. Und diese Rolle habe ich jetzt gerade ein bisschen übernommen. Und es wird so ein bisschen/ Um halt wieder Leben in die Sache einzuhauchen, wird es halt ein Release geben und dann merken die Leute, dass es noch lebt und dann gibt es dann wieder mehr Energie. Also nach außen hat Software ja immer einen Zustand von: Das benutzen wir noch solange, bis es dann durch was Anderes ersetzt wird. Oder etwas, das ist was Schönes, wenn ich da Dinge bemerke, die besser werden möchten, dann werde ich sie einreichen und werde es dann noch weiter verbessern und das ist ja ein/ Also das ist ja in der Außenwahrnehmung halt auch irgendwie relevant, wenn Software als tot wahrgenommen wird, dann wird sie halt auch nur noch ihr Schattendasein fristen und dann irgendwann sterben, weil Leute nichts mehr dazu beitragen. Und wenn Software als lebendig wahrgenommen wird, dann stecken Leute auch noch von außen ihre Zeit rein und verbessern sie.“ (Dokument I-4, Abs. 44)

Das Interesse dieser Person ist nicht nur die einfache Anpassung einer existierenden Software, um sie für eigene Anwendungsbereiche einzusetzen. Ihr geht es vielmehr darum, den zuvor eingestellten Produktentwicklungszyklus neu zu initiieren. Dafür ist in der Regel die Übernahme der Rolle des:der Projektmaintainer:in erforderlich, was neben der Softwareentwicklung auch die Projektpräsentation und der Akquise von weiteren Entwickler:innen beinhaltet. Hieran zeichnet sich schon ab, dass die Wiederaufnahme der Entwicklungsarbeit und der Aufbau einer neuen Entwicklungsgemeinschaft bedeutend anspruchsvoller sind als die Mitarbeit bei einer bestehenden Entwicklungsgemeinschaft.

Aus diesem Grund wird dies eher von Entwickler:innen praktiziert, die bereits zuvor in anderen FLOSS-Projekten die Praxis der Entwicklungsarbeit kennengelernt haben.

Neben der Werteorientierung von Software als schützenswertes Kulturgut kann die Wiederaufnahme der Entwicklungsarbeit auch aus ökonomischen Erwägungen begründet werden:

> „[...] wenn ich selber ein Problem habe und es irgendwie nicht anders lösen kann, würde ich was Neues starten. Wenn es aber gute andere laufende Projekte gibt, dann würde ich immer vorziehen, weil es einfach natürlich viel weniger aufwändig ist und es ist auch viel hilfreicher, wenn man andere Sachen mitbenutzt. Also es ist nicht gut, wenn man immer nur neue Sache startet, ohne die Bestehenden versuchen zu integrieren oder weiter zu benutzen. Und wenn man schon was bestehendes Gutes hat, sollte man auf jeden Fall daran weitermachen [...]“ (Dokument I-8; Abs. 161)

Neben der Werteorientierung zum Erhalt von Software und ökonomischen Kosten-Nutzen-Relationen ist es jedoch eine Einzelfallentscheidung, ob Entwickler:innen die Wiederaufnahme oder die Neuentwicklung favorisieren. Ausschlaggebende Faktoren sind da oft auch struktureller Art. Dies bezieht sich laut Interviewaussagen von verschiedenen Entwickler:innen beispielsweise auf die Lesbarkeit des vorhandenen Quellcodes, Kenntnisstand über die damals verwendete Programmiersprache oder ob die Programmstruktur modular aufgebaut war, was die verteilte Entwicklungsarbeit erleichtert und die Anpassung einzelner Komponenten des Softwaresystems ermöglicht. All diese Faktoren spielen bei einer derartigen Entscheidung eine Rolle.

7.5 Zusammenfassung

Dieses Kapitel leistet einen wichtigen Beitrag dazu, die Beweggründe von Entwickler:innen in der FLOSS-Bewegung zu differenzieren. Obwohl es schon mehrere Studien gibt, die ebenfalls die Beweggründe von Entwickler:innen für ihr Engagement erfragten, wurde sich bislang auf einen undifferenzierten Katalog von Aussagen beschränkt.

Anhand der Auswertung der Leitfadeninterviews kann die Komplexität der Beweggründe für eine FLOSS-Engagement beleuchtet werden. Die Interessen der Entwickler:innen sind eingebettet in spezifische Werteorientierungen, die neben einigen Vorbedingungen zu einer ersten Mitarbeit in solchen Projekten führen können. Dazu bedarf es eines konkreten Auslösers, der das Handeln auf ein oder mehrere bestimmte FLOSS-Projekte kanalisiert. Durch ihre Mitarbeit erfahren die Entwickler:innen ein breites Spektrum an zuvor unbekannten Aspekten in Bezug auf die Praxis in FLOSS-Gemeinschaften. Es erfolgt eine erfahrungsbasierte Neubewertung der Auslöser, die zu einer Diversifikation der Beweggründe führen. Die Interessen ändern sich nicht grundlegend, werden aber um neue Aspekte aufgrund der gesammelten Erfahrungen bei der Entwicklungsarbeit ergänzt.

Die folgende Abbildung repräsentiert das Bedingungsgeflecht der Interessenlagen von Entwickler:innen. Es ist keine erschöpfende Darstellung aller möglichen Einflussfaktoren, sondern nimmt die Faktoren zur Kenntnis, die im Rahmen der gemeinschaftlichen FLOSS-Entwicklung von den befragten Entwickler:innen thematisiert wurden.

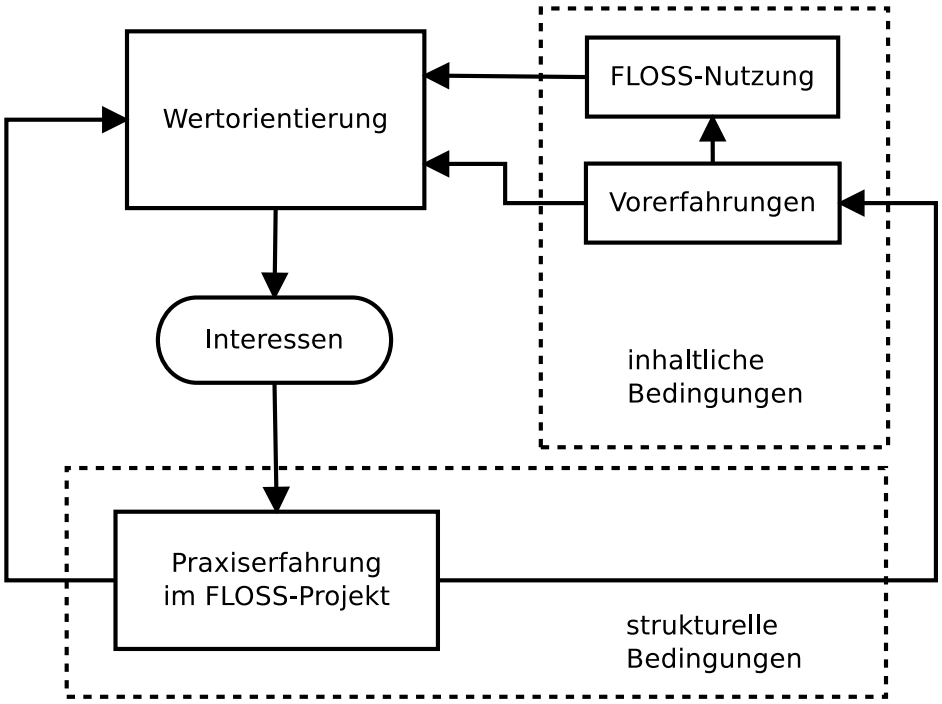

Abbildung 7: Einflussfaktoren für ein FLOSS-Engagement

In der Abbildung 7 ist das Bedingungsgeflecht der Beweggründe für ein FLOSS-Engagement dargestellt. Es zeigte sich, dass viele Entwickler:innen zuvor Nutzer:innen von FLOSS waren beziehungsweise die Praktiken und Wertvorstellungen, wie sie in Entwicklungsgemeinschaften der FLOSS-Bewegung üblich sind, auch schon in anderen Kontexten erfahren haben. Diese beiden Vorbedingungen sind im Sinne der Situiertheit von Lernprozessen bedeutsam. Nutzt man das Interessenkonstrukt von Krapp (vgl. 1992b, S. 41), entsprechen die inhaltlichen Bedingungen dem aktualisierten Interesse.

Die durch Erfahrung gewonnenen inhaltlichen Vorbedingungen formen die Wertorientierung. Bei der Befragung der Entwickler:innen nach den Beweggründen für ihr Engagement wurden verschiedene Werte identifiziert, die für ihr Handeln relevant waren. Dies waren die Förderung sozialer Teilhabe, der Schutz von Kulturgütern, reziproke Hilfsbereitschaft und die Partizipation. Die Werteorientierung wirkt sich im Verbund mit anderen biographischen Faktoren auf die Interessen aus. Diese sind latent vorhanden, wie zum Beispiel ein thematisches Interesse, können sich aber auch in einer bestimmten Situation konkret ausformen (pragmatisches Interesse). Um die Bedeutsamkeit der situativen Bedingungen herauszuarbeiten, in denen es zu einem ersten Engagement in einem FLOSS-Projekt kommt, wurden diese als Auslöser für ein Engagement beschrieben. Folgende Interessen wurde für ein erstes FLOSS-Engagement identifiziert: Software den eigenen

Ansprüchen anpassen, eine bestimmte FLOSS in eine Distribution integrieren, Sicherheitslücken in der Software beseitigen, ein neues FLOSS-Projekt starten, Unterstützung von Personen im sozialen Umfeld und das Engagement in einem FLOSS-Projekt als Lerngegenstand in einer formalen Bildungssituation.

Die Auslöser als situativ kanalisierte Interessen führen zu einem ersten Engagement in einem FLOSS-Projekt. Dabei ist zu berücksichtigen, dass dies nur beim Vorhandensein struktureller Bedingungen möglich ist. Das ist vor allem Zeit. Diese ist notwendig, um sich vor einem eigenen Beitrag zunächst einmal die von der Entwicklungsgemeinschaft genutzten Werkzeuge anzueignen und die Praktiken der Gemeinschaft zu erlernen. Ist dieser Schritt vollbracht, braucht es auch Zeit für die eigentliche Entwicklungsarbeit selbst. Bei beiden Schritten ist die Unterstützung anderer Mitglieder der Gemeinschaft hilfreich und kann für eine schnellere Partizipation des neuen Mitglieds sorgen. Die Partizipation wiederum ermöglicht durch das Erleben der Praxis wieder neue Erfahrungen und neues Wissen, was sich wiederum auf die inhaltlichen Bedingungen und die Wertorientierung auswirkt. Als Rückkopplung steuert diese das zukünftige Engagement in der Gemeinschaft oder das Engagement in anderen FLOSS-Projekten.

Praxiserfahrungen führen zu einer Veränderung der Interessen. Die Entwickler:innen gleichen ihre Erwartungen an die gemeinschaftliche Entwicklungsarbeit mit ihren Erlebnissen ab. Sie bewerten daraufhin ihre Interessen neu. Teilweise kanalisieren sich dadurch auch Interessen, die beim ersten Engagement in einem FLOSS-Projekt noch nicht Teil der Erwartungen waren. Folgende Interessen wurden von Entwickler:innen thematisiert, nachdem sie bereits längere Zeit an Entwicklungsgemeinschaften partizipiert hatten: Freude am Lernen, Erwartungen von anderen, FLOSS als Lohnarbeit, Anerkennung, Interesse am Thema, Austausch mit Gleichgesinnten und der Erhalt von Software. Dies zeigt, dass Interessen kein statisches Gebilde sind, sondern vor dem Hintergrund von Erfahrungen, Vorwissen und dem situativen Kontext immer wieder neu bewertet werden und sich in einer Änderung der individuellen Praktiken niederschlagen.

8 Lernen durch die Entwicklungsarbeit

Soeben wurde dargestellt, warum sich Entwickler:innen in FLOSS-Projekten engagieren. In dem nun folgenden Kapitel wird der Frage nachgegangen, wie innerhalb der Entwicklungsarbeit gelernt wird. Zu diesem Zweck werden die Praktiken der Entwicklungsarbeit genauer betrachtet. Der Fragenkomplex *Prozess* des Leitfadeninterviews zielt auf die Befragung der Entwickler:innen zu ihren Strategien bei der Problemlösung. Um eine Beschreibung des Lernens zu ermöglichen, braucht es Kenntnisse über die Routinen und Hilfsmittel bei der Entwicklungsarbeit. Insbesondere der Einfluss anderer Entwickler:innen auf das individuelle Lernen wird hier analysiert. Hintergrund ist der soziale Kontext von Lernprozessen, der, wie es im Kapitel 3.2 begründet wurde, eines der zentralen Forschungsfragen dieser Arbeit darstellt. Dementsprechend wird die Entwicklungsarbeit aus vier verschiedenen Perspektiven betrachtet.

Die erste Perspektive gilt den verschiedenen Praktiken, die von den Entwickler:innen beim Engagement in ihrem FLOSS-Projekt ausgeübt werden. Anhand der Erkenntnisse aus den Interviews und der Beitragsanalyse aus den Mailinglisten werden die wichtigsten Abläufe und Rollen beim FLOSS-Engagement vorgestellt. Die Praxis der Entwicklungsarbeit ist immer wieder von Störungen betroffen. Deswegen werden diese in die Betrachtung mit einbezogen. Die zweite Perspektive beschreibt die Problemorientierung bei der Entwicklungsarbeit. Es werden die Abläufe aus Sicht der Entwickler:innen vorgestellt, die von einem Problem bis hin zu dessen Lösung durchlaufen werden. Im Anschluss daran sollen in der dritten Perspektive die Lösungsstrategien beim problemorientierten Lernen genauer vorgestellt werden. Es werden drei grundlegenden Strategien identifiziert, die bei der Problemlösung Anwendung finden. Oft erfolgt das Vorgehen unter Zuhilfenahme mehrerer Strategien, weswegen auch die Ursachen für einen Strategiewechsel dargestellt werden. Die vierte Perspektive gilt den genutzten Ressourcen bei der Entwicklungsarbeit: Es werden die verschiedenen Informationsquellen aufgeführt, die für die Softwareentwicklung relevant sind. Ein besonderer Fokus liegt dabei auf der Hilfe im Rahmen von sozialen Austauschprozessen zwischen den Entwickler:innen.

8.1 Praxis der Entwicklungsarbeit

Um das Lernen bei der Entwicklungsarbeit zu beschreiben, soll zunächst einmal die Entwicklungsarbeit selbst in den Fokus genommen werden. Hierbei wird sich auf die wesentlichen Praktiken der Softwareentwicklung konzentriert. Dabei ist aber nicht zu vernachlässigen, dass die Praxis mitunter ein für die Entwickler:innen frustrierender Prozess darstellen kann. Aus diesem Grund werden auch Hindernisse bei der Entwicklungsarbeit thematisiert. Der Schwerpunkt wird hierbei auf die sozialen Aushandlungsprozesse gelegt, die sowohl für die Problemlösung als auch für die Koordinierung der Entwicklungsarbeit relevant sind.

8.1.1 Praktiken

In der Darstellung der Praktiken werden die Tätigkeiten beschrieben, die in den Interviews den meisten Raum einnahmen: das eigentliche Programmieren und die damit verbundenen Aushandlungsprozesse in der Entwicklungsgemeinschaft. Als essentieller Bestandteil des Softwareentwicklungszyklus beinhaltet dies auch das Testen von Software. Dazu gehört auch die besondere Rolle von Projektmaintainer:innen, da sie neben der eigentlichen Programmierarbeit auch eine Schlüsselposition in der Entwicklungsgemeinschaft einnehmen und maßgeblichen Einfluss auf die Projektentwicklung nehmen. Dabei soll aber nicht außer Acht gelassen werden, dass Softwareentwicklung nicht nur aus Programmieren und Projektmanagement besteht. Beim qBittorrent-Projekt wurde bereits die Bedeutung der Sprachübersetzung vorgestellt. Daneben gibt es noch eine Vielzahl anderer Tätigkeiten wie beispielsweise Dokumentation, Benutzungsfreundlichkeit oder Präsentation des FLOSS-Projektes auf Webseiten und Konferenzen. Exemplarisch soll hierfür die Unterstützung von Nutzer:innen vorgestellt werden, da die Erfahrungen von Nutzer:innen sich direkt auf die Entwicklung der Software auswirkt. Entweder direkt durch Fehlerberichte oder gezielte Ansprachen an die Entwickler:innen des FLOSS-Projektes oder indirekt durch Diskussionen über die FLOSS auf Foren, die nicht Teil der Kommunikationswerkzeuge der Entwicklungsgemeinschaft sind.

Bugfix / Softwarepatch

Eine Fehlerbehebung im Quellcode kann eine einmalige Aktion darstellen oder auch in ein längerfristiges Engagement eingebunden sein. Bei einer längerfristigen Tätigkeit werden seriell Programmfehler bearbeitet oder das Programm wird um weitere Funktionen ergänzt. Auch der Grad der Interaktion mit anderen Entwickler:innen kann stark variieren. Eine kleine Änderung am Quellcode, eine Übersetzung oder die Korrektur eines Rechtschreibfehlers in der Bedienoberfläche kann oft ohne Interaktion mit anderen durchgeführt werden, während andere Tätigkeiten die Hilfe weiterer Personen erfordern.

> „Das erste ist natürlich, irgendwie diesen Bug-Report bestmöglich zu verstehen. Wenn ich ihn nicht verstehe, dann frage ich natürlich solange nach, bis ich ihn Eins-zu-Eins reproduzieren kann. Der Schritt dauert ja manchmal schon lange genug, wenn man nicht genau die gleiche Distro[124] oder genau die gleiche Python-Version zur Hand hat. Ich frage dann: Ja was verwendest du? Wie genau ist der Bug zu reproduzieren? Und wenn ich dann den Bug zumindest reproduzieren kann, dann muss man halt irgendwie zurückführen, wo im Code es ist. Zumindest bei Python spuckt er ja recht schnell die Zeile aus, wo es/ Wenn er da jetzt irgendwie explizit eine Exception[125] schmeißt,

[124] Distro ist die Abkürzung für die GNU/Linux-Distribution, unter welcher der Softwarefehler auftrat.

[125] Eine 'Exception' ist eine Fehlermeldung vom Interpreter der Programmiersprache bei der Ausführung vom Quellcode. Der Interpreter unterbricht dann die Programmausführung und verweist auf die Stelle im Algorithmus, bei der der Fehler auftrat.

wo das passiert. Dann, wenn ich das nicht verstehe, dann wird halt genau diese Exception in DuckDuckGo[126] eingegeben und dann schaut man mal, was so alles rauskommt.“ (Dokument I-11; Abs. 64)

Der nächste Schritt ist dann die Korrektur des Fehlers und das Veröffentlichen des modifizierten Programmcodes. Da die befragte Person selbst auch Projektmaintainer:in der Software ist, kann sie die Änderung direkt selbst einpflegen. Andernfalls wird der Bug-Fix zum Beispiel als Pull-Request bei GitHub veröffentlicht und die jeweiligen Projektmaintainenden pflegen diesen dann ein.

Interaktion mit der Entwicklungsgemeinschaft

Mit der Größe des Projektes nehmen auch die sozialen Austauschprozesse zu. Dies ist besonders dann der Fall, wenn das Selbstverständnis der Gruppe eine gemeinschaftliche Entscheidungsfindung vorsieht. Die individuelle Praxis beschränkt sich dann nicht mehr nur auf die eigene Entwicklungsarbeit, sondern beinhaltet auch die Gestaltung der sozialen Beziehungen und Aushandlungsprozesse. Das folgende Zitat ist aufgrund seiner Länge in zwei Abschnitte unterteilt, um das differenzierte Vorgehen bei Entscheidungsfindungsprozessen in der Entwicklungsgemeinschaft zu beschreiben:

„Ja (.) ja also ich meine grundsätzlich ist es ja so ein offenes, freies Projekt, wo halt auch irgendwie mit drinnen steckt, dass man alle beteiligt, die ein Interesse daran haben oder dass/ Die Idee ist halt nicht, dass einfach nur einer eine Entscheidung trifft und dann wird die umgesetzt, sondern dass wir halt schon irgendwie konsensbasiert vielleicht Entscheidungen treffen und da ist es natürlich wichtig, dass man irgendwie zu diesem Konsens kommt oder zumindest mal eine Diskussion über die Themen anregt, führt. Ja und/ Also bei der Softwareentwicklung jetzt ist es halt so, dass/ Also wir haben halt so einen Issue-Tracker, da kann man dann, wenn man jetzt irgendwie eine Idee hat, kann man die dort vorschlagen. Dort erreicht sie dann erst einmal nur sozusagen die Entwickler. Also die kriegen dann mit: Okay, da hat jemand was eingestellt, gibt es eine Idee, dann kann es da zu einer Diskussion kommen. Im Idealfall ist es halt so, dass es halt für alle überzeugende Argumente gibt, die dann dazu führen, dass man so die Richtung einschlägt. Kann aber auch mal sein, dass man dort zu keiner Lösung kommt oder einfach das Thema als so wichtig erachtet, weil das irgendwie eine grundlegende Änderung ist, dass man das mit einem größeren Kreis besprechen will. Da gibt es/ Skalieren wir es dann quasi an unsere Mailingliste, was so ein bisschen das Kommunikationsorgan für unsere Community ist. Da sind alle Leute drauf eigentlich, die sich interessieren und aktiv sind. Da kann man dann versuchen eine Diskussion zu führen und was halt auch immer geht, dass man natürlich auch mal einlädt dazu sich auf einem Treffen jetzt hier mit dem Thema zu befassen. (..)

[126] 'DuckDuckGo' ist eine Suchmaschine.

> Ja, es gibt da jetzt irgendwie keine so formalen [unverständlich] oder so 'ne. Ich weiß nicht ob wir das überhaupt mal so aufgeschrieben haben, wie wir uns das wirklich vorstellen diese Firmware[127]-Entwicklung. Also das kommt immer mal wieder auch zur Sprache." (Dokument I-2; Abs. 56)

Die Interaktionen dienen nicht nur der Klärung von Problemen bei der Entwicklungsarbeit wie es in Kapitel 6.5.5 im Zuge der Problemlösung bei der Entwicklungsarbeit näher ausgeführt wurde, sondern sie ermöglichen auch die Koordinierung der gemeinsamen Unternehmung. Größere FLOSS-Projekte mit mehreren hundert Aktiven gliedern sich beispielsweise in lokale Gruppen und gehen arbeitsteilig vor. Hier in diesem Beispiel gibt es eine strukturelle Trennung von Firmware-Entwickler:innen und den Personen, die die Endgeräte mit der Firmware betreiben. Eine Änderung der Firmware kann sich stark auf die Nutzung der Endgeräte auswirken, so dass ab einem bestimmten Punkt die Entscheidung mit den anderen Mitgliedern der FLOSS-Gemeinschaft abgestimmt werden muss. Damit gibt der Entwickler zu erkennen, dass die Entscheidungsfindung im Projekt nach dem Subsidaritätsprinzip erfolgt. Wie genau jedoch der Prozess der Entscheidungsfindung erfolgt, ist dem Zitat zufolge in der Entwicklungsgemeinschaft nicht klar festgelegt, aber durch eine Art undokumentierte Praxis als Handlungsmaxime anerkannt.

> „Oder dann sagt einer der Entwickler oder Entwicklerinnen: Hey, hier, wir sollten das mal zum Beispiel auf der Mailingliste besprechen. Genau, ansonsten gibt es da auch viele Absprachen auch teilweise so privat. Also auf den Treffen bespricht man auch einfach mal was zwischen den Entwicklern oder man ist halt in so einem privaten Chat, oder [...] Es kann auch mal sein, dass mal einfach zwei Leute was besprechen beim Bier oder so oder man fährt vom Treffen zusammen zurück und dann da mal was bespricht oder es gibt dann da auch so/ Also von Leuten, die man besser kennt, hat man meistens noch einen Chatkontakt irgendwie für Jabber[128] oder so. Dass man da dann auch einfach was bespricht, das führt natürlich auch dann dazu, dass das teilweise/ Also es wird so ein bisschen intransparent, was da passiert. (.) Genau, also wir versuchen schon aktiv sozusagen das in der Öffentlichkeit zu besprechen, das alles irgendwie transparent ist und unsere Entscheidung für die anderen nachvollziehbar ist." (Dokument I-2; Abs. 56 - 60)

In FLOSS-Projekten werden verschiedene Wege der Entscheidungsfindung praktiziert. Diese reichen von einer klaren Hierarchie mit einer Person[129] oder einer Personengruppe[130] an der Spitze bis zu großen FLOSS-Projekten die ohne klar spezifizierte Entschei-

[127] Als 'Firmware' wird Software bezeichnet, die in elektronische Geräte eingebettet ist und dort zur Erfüllung der Grundfunktionen des Gerätes notwendig ist. Sie grenzt sich von der Anwendungssoftware ab, da sie durch die Nutzenden nicht oder nur mit speziellen Mitteln verändert werden kann. Solche Firmware befindet sich beispielsweise in Mobiltelefonen, Waschmaschinen, Druckern oder Routern.

[128] 'Jabber' der veraltete Name für das quelloffene Protokoll 'Extensible Messaging and Presence Protocol (XMPP)'. Es dient zum Austausch von Daten als Instant-Messenger bzw. als Chat-Dienst.

[129] In dem Fall ist es oft die Autorität des:der Projektmaintainer:in.

[130] Dies sind im Regelfall die Kernentwickler:innen.

dungsstrukturen auskommen (vgl. Taubert 2008, S. 72f). Entscheidungen werden aber auch von den Leuten getroffen, die sich für zuständig halten beziehungsweise von der Entwicklungsgemeinschaft als solches wahrgenommen werden, wie es in diesem Beispiel gezeigt wurde. Dabei gibt es unterschiedliche Möglichkeiten, in welchen Rahmen Entscheidungen besprochen oder bestimmte Entscheidungsoptionen von wenigen Entwickler:innen vorbereitet werden. In jedem Fall kann aber festgestellt werden, dass mit einer Zunahme an involvierten Personen der Aufwand für die Entscheidungsfindung und die Pflege der sozialen Beziehungen steigt. Einige FLOSS-Projekte nutzen dafür auch den sozialen Austausch auf eigens dafür organisierten Events oder Konferenzen. Teilweise werden auch thematisch ähnlich gelagerte Konferenzen, wie zum Beispiel die Linux-Tage, für Treffen von Entwicklungsgemeinschaften genutzt.

Softwaretest

Ebenso Teil der Entwicklungsarbeit ist das Testen von Software. Im Gegensatz zur Implementation von Software kann dies auch von Personen ohne Programmierkenntnisse durchgeführt werden. Auch wenn die Fehler nicht selbst behoben werden können, ist deren umfassende Dokumentation eine wichtige Vorbereitung für deren Behebung.

> „Ich habe beispielsweise auch die/ Was auch viel noch war, dass die Entwickler halt Sachen neu entwickelt haben und dann Leute brauchten, die das Zeug gegentesten. Dann schauen: Läuft es denn auf deiner Plattform oder ist es gegen die und die Sachen anfällig? Ich habe halt ein Arsenal an verschiedenen Betriebssystemen und Clients zur Verfügung gehabt, wo man einfach Dinge dann durchtesten konnte. So was habe ich noch viel gemacht, also eher Software-Testing.“ (Dokument I-5, Abs. 69)

Die Kenntnis der genauen Umstände von einem Softwarefehler geben Hinweise auf die Ursache des Fehlers. Wie die Person es beschreibt, benötigt ein möglichst vollständiger Softwaretest eine Vielzahl von unterschiedlichen Testumgebungen und Testeingaben. Teilweise können Tests auch mit eigens dafür geschriebener Testsoftware ausgeführt werden. Dies erfordert jedoch einen großen Aufwand und ist nicht für alle Testszenarien möglich. Aus diesem Grund ist Unterstützung beim Testen ein nicht zu unterschätzender Beitrag für die Softwareentwicklung.

Projektmaintainer:innen

Die Aufgaben von Projektmaintainer:innen wurden teilweise schon im Kapitel 6.6.2 bei der Beschreibung von Marlins Engagement bei dem FLOSS-Projekt qbittorrent und dem Rollenwechsel zum:zur Projektmaintainer:in vorgestellt. Die Inhaltsanalyse der beiden Mailinglisten zeigte, dass die Planung und Veröffentlichung einer neuen Softwareversion seitens der Projektmaintainenden eine der wesentlichen Kommunikationsinhalte waren.

Neben der Kommunikation, die erst die koordinierte Entwicklung möglich macht, haben die Projektmaintainenden ihren Arbeitsschwerpunkt bei der Programmierung. In den meisten Projekten sind sie es, die den eigentlichen Fortschritt am Quellcode des Projekts vorantreiben. Dies erfolgt aufgrund der eigenen Beiträge, aber bei zunehmender Bekanntheit des Projekts auch auf Grundlage der Beiträge anderer Entwickler:innen.

> „Also früher waren ja Beteiligunsprozesse so ein bisschen schwergängiger, kann man sagen. Also solche Konzepte wie dieses GitHub oder so etwas, wo halt relativ leicht Menschen Änderungsvorschläge einreichen können, das erleichtert ja eine relativ aufwandsarme Art der Beteiligungsmöglichkeit. Das erfordert, dass die Menschen halt in freien Softwareprojekten nicht mehr so die Dinge selber machen, sondern auch mehr Community-Management machen und mit Einwürfen umgehen. (..) Und damit auch versuchen, neue Leute vielleicht reinzuzerren. Was da aber dann weniger der Fall als früher ist, dass man dort eher so Individualinteraktion hat. Also es gibt nicht mehr, so wie früher, so ein gemeinsames Kommunikationsmedium, indem man unter den Entwicklern irgendwie noch sich austauscht und Dinge miteinander diskutiert oder so etwas oder sagt, was man getan hat. Sondern der Anstoß zur Kommunikation kommt typischerweise von außen und darüber gibt es dann Individualkommunikation. Also die Sache hat sich so ein bisschen verändert im Sinne von (.) ein bisschen mehr nach außen aber auch losere Beziehungen nach innen, weil es keine gemeinsame Kommunikationskanäle eigentlich mehr gibt.“ (Dokument I-4, Abs. 84)

Vor dem Hintergrund eines langjährigen Engagements in mehreren FLOSS-Projekten, als Contributor[131] wie auch als Projektmaintainer, berichtet dieser Entwickler von einem Wandel in der Art und Weise, wie sich die sozialen Beziehungen in Entwicklungsgemeinschaften aufgrund der Nutzung von FLOSS-Portalen ändern. Der Wandel führt dazu, dass anstelle von projekteigenen Kommunikationsmedien, wie beispielsweise die beiden untersuchten Mailinglisten vom qBittorrent-Projekt, vermehrt strukturierte Kommentarfunktionen auf FLOSS-Portalen genutzt werden. Beispielsweise werden dafür die Issues genutzt, also die Fehlerberichte, die auf dem Bug-Tracker gesammelt werden. Die damit einhergehende Vereinheitlichung der Kommunikation kann dazu führen, dass die Partizipation an der Entwicklungsgemeinschaft sich nur auf ein singuläres Ereignis beschränkt. Dementsprechend reduzieren sich auch die Austauschprozesse in der Gemeinschaft.

> „[...] die meisten Menschen schicken halt doch eher so eins, zwei Pull-Requests und dann war es das. Also es gibt halt quasi zwei Maintainer beim Kalender, aber sonst sind es eigentlich (.) nur vereinzelte Contributions.“ (Dokument I-10; Abs. 75)

[131] Als ’Contributor’ werden Entwickler:innen bezeichnet, die Beiträge zum Projektfortschritt einreichen, sich aber nicht längerfristig im Projekt engagieren.

Diese Person füllt ebenfalls die Rolle als Projektmaintainer:in aus. Sie berichtet von ihren Erfahrung bei einem FLOSS-Projekt, welches ebenfalls GitHub nutzt, und bestätigt die Interaktionstiefe, die auch von der vorher zitierten Person geschildert wurde. Diese Entwicklung kann aber nicht als repräsentativ für die FLOSS-Bewegung gesehen werden, sondern eher als eine Veränderung durch ein zusätzliches Werkzeug wie GitHub. Dieses ist insbesondere für kleinere Projekte aufgrund des Funktionsumfangs attraktiv, verändert aber auch die Interaktion in Entwicklungsgemeinschaften. Größere Projekte etablieren ab einer kritischen Masse an Entwickler:innen eigene Kommunikationskanäle und diversifizieren damit ihre Arbeits- und Kommunikationsstrukturen.

Nichtsdestotrotz bleibt es die Aufgabe von Projektmaintainer:innen, die Beiträge von anderen zu prüfen und sie in den Programmquellcode aufzunehmen oder sie gegebenenfalls konstruktiv zurückzuweisen. Die Kommunikationsbeziehungen laufen dann über das FLOSS-Portal anhand eines konkreten Fehlerberichtes zwischen Contributor:in und Maintainer:in. Bei größeren Projekten gibt es dafür eigene Mailinglisten oder andere Kommunikationskanäle, wo alle interessierten Entwickler:innen die Inhalte mitverfolgen können.

Unterstützung von Nutzer:innen

Größere FLOSS-Projekte nutzen verschiedene Kommunikationskanäle, um die Kommunikationsinhalte thematisch zu trennen. Auf separaten Kommunikationskanälen für die Nutzer:innen in Form von beispielsweise Foren oder Mailinglisten können deren Fragen beantwortet werden. Damit kann die Kommunikation der Entwicklungsgemeinschaft von der der Nutzer:innen separiert werden. Dennoch ist die Sphäre der Nutzung nicht von der Sphäre der Entwicklung getrennt.

Die folgend zitierte Person nutzt ihr Fachwissen bezüglich der FLOSS, um Nutzer:innen mit schwierigen Problemen zu helfen. Das Forum wurde für die Nutzer:innen einer Grafiksoftware mit reichhaltigem Funktionsumfang erstellt. Es ist von Nutzer:innen frequentiert, die Probleme im Umgang mit der Software haben. Hilfe erfahren sie dabei von erfahrenen Nutzer:innen wie auch von einigen Entwickler:innen, die gegebenenfalls die Probleme an die Kommunikationsstrukturen der Entwickler:innen weiterleiten:

> „Und ich mache Nutzer-Support. Grase also regelmäßig die Foren ab nach Fragen, wovon ich weiß, dass andere sie nicht beantworten können. Also, es ist halt so, dass von den Leuten, die Support geben, ich diejenige bin, die am meisten technisches Verständnis hat und dann suche ich halt die Fragen raus, die sozusagen für mich dann übrigbleiben würden. [...] Aber wenn es jetzt um irgendwas ganz Spezielles geht in den Support-Anfragen, dann weiß ich halt auch meistens, welcher Entwickler sich damit auskennt. Oder wer das verbockt hat im schlimmsten Falle und kann die Leute dann einfach direkt ansprechen und fragen.“ (Dokument I-3, Abs. 37, 100)

Da die Entwicklerin auch Kenntnisse vom Fachwissen und den Themenbereichen der

anderen beteiligten Entwickler:innen hat, kann sie das Problem an die entsprechenden Personen weiterleiten. Damit stellt sie die Verbindung zwischen den Problemen bei der Nutzung und dem Entwicklungsprozess her. In anderen Projekten gibt es dagegen regelmäßig lokale Treffen. Diese fungieren als Schnittstelle zwischen beiden Sphären:

> „[...] wir haben regelmäßige Treffen, so jeden Mittwoch und da können alle hinkommen. Da gibt es auch einfach sehr viele neue Menschen, dass man dort halt da ist und halt dann Fragen beantwortet. Sozusagen, ja (.) neuen Menschen irgendwie einen einfachen Einstieg ermöglicht oder so.“ (Dokument I-2; Abs. 28)

Diese wöchentlichen Treffen dienen der Koordinierung der lokalen Entwicklungsgemeinschaft. Gleichzeitig sind sie auch ein Anlaufpunkt für neue Nutzer:innen. Da eines der Ziele des Projektes der Aufbau von Netzinfrastruktur ist, sind neue Nutzer:innen ein wichtiger Bestandteil für das angestrebte Wachstum des Netzwerkes. Aus diesem Grund wird der Unterstützung von neuen Nutzer:innen ein hoher Stellenwert beigemessen. Diese Unterstützung wird auch durch viele teilnehmende Entwickler:innen gegeben, die auf diesen Treffen präsent sind.

8.1.2 Hindernisse bei der Entwicklungsarbeit

Die soeben dargestellten Praktiken erfordern entweder den direkten Kontakt oder den internetbasierten Austausch mit anderen Mitgliedern der Entwicklungsgemeinschaft. Diese führen jedoch nicht immer zu zufriedenstellenden Aushandlungsprozessen zwischen den Entwickler:innen. In dem folgendem Abschnitt sollen die Hindernisse benannt werden, die Teil der individuellen Entwicklungsarbeit sein können. Hierbei wird differenziert nach Hindernissen, die sich aus dem Mangel an Ressourcen ergeben und denen, die sich aus den direkten Aushandlungsprozessen innerhalb der Entwicklungsgemeinschaft ableiten. Sie werden hier kurz erwähnt, da sie großen Einfluss auf das individuelle Engagement haben können. Insbesondere die sozialen Konflikte können zu einem Rückzug aus der Entwicklungsgemeinschaft führen. Hindernisse sollten aber nicht nur defizitär betrachtet werden. Konflikte sind immer ein Teil von sozialen Aushandlungsprozessen und zugleich auch eine produktive Kraft, die zur Anpassung von Praxisgemeinschaften an sich ändernde Bedingungen führen kann (vgl. Wenger 1998, S. 84).

Ressourcenprobleme

Einer der wichtigsten limitierenden Faktoren für das FLOSS-Engagement ist die Zeit. Dies betrifft vorrangig Entwickler:innen, die in ihrer Freizeit aktiv sind.

> „Da so richtig eingestiegen bin ich halt, während ich Student war beziehungsweise später Doktorand und ich glaube, das hat sich insofern geändert, als das ich halt weniger Zeit für die Sache aufwenden kann. Man hat halt Arbeit, die raubt einem/ Na was heißt raubt? Die Zeitverteilung, die man selber

> vornimmt, die verändert sich halt über die Zeit. (.) Man könnte behaupten, dass ich immer mehr nur organisatorische Maßnahmen wahrnehme und weniger an den Projekten selber [arbeite] und eher die Leute zusammenbringe. Also es gibt da schon eine Verschiebung von dem selber Wirken zu eher so koordinierenden Funktionen.“ (Dokument I-7; Abs. 28)

Das FLOSS-Engagement steht in Konkurrenz zu anderen Anforderungen im Lebensalltag. Viele FLOSS-Entwickler:innen beginnen ihre Entwicklungsarbeit in Lebensphasen, wo sie über relativ viel Freizeit verfügen. Wendepunkte in der Biographie wie beispielsweise die Aufnahme einer Lohnarbeit oder eine Familiengründung können sich zum Nachteil für das FLOSS-Engagement auswirken. Dies wirkt sich besonders dann negativ auf die gemeinschaftliche Entwicklungsarbeit aus, wenn dies Personen betrifft, die wichtige Aufgaben wahrnehmen oder über spezielle Kenntnisse verfügen. Viele der befragten Entwickler:innen haben die Rolle als Projektmaintainer:in nur deshalb übernommen, weil diese Position von keiner Person mehr ausgefüllt wurde oder der:die Projektmaintainer:in aus Zeitmangel diese Rolle gerne abgeben wollte. Die Ressource Zeit macht sich besonders bei kleinen Projekten bemerkbar, wenn sich viele Aufgaben auf eine Person konzentrieren. Dies ist auch einer der Gründe, weswegen ein Zustrom von neuen Entwickler:innen für FLOSS-Projekte mitunter überlebenswichtig sein kann.

> „[...] es wäre natürlich sehr schön irgendwann so eine gewisse feste Basis an Programmierern und Usern zu haben, die immer bereit sind, wenn was Neues fertig ist, auch entsprechend zu testen. Ich habe einfach alleine nicht die Ressourcen irgendwie auf sechs virtuellen Maschinen alle Distributionen durchzuprobieren. Und automatisierte Tests sind nicht immer genau das Wahre für ein grafisches Programm.“ (Dokument I-11; Abs. 100)

Diese Person schätzt ihre Kapazitäten für das Testen der von ihr entwickelten Software als unzureichend ein. Gern würde sie mehr testen, wozu ihr allerdings die Zeit fehlt. Die Unterstützung von Nutzer:innen mit möglichst diversen Computersystemen definiert sie als Engpass in der Entwicklungsarbeit. Auf Grundlage von deren Testergebnissen könnte die Software auch für andere Computersysteme zufriedenstellend genutzt werden. Dies würde wiederum den Kreis der Nutzenden und damit auch von potenziellen Entwickler:innen vergrößern. Ein Zeitmangel kann durch Unterstützung weiterer Entwickler:innen kompensiert werden. Dabei müssen Anforderungen an die notwendige Hilfe klar kommuniziert werden. Nicht immer ist jedoch die Unterstützung qualitativ ausreichend, um ein komplementärer Ersatz für Ressourcenmangel zu sein. Dieses Beispiel zeigt, dass unzureichende Beiträge von Softwaretester:innen keine Hilfe bei der Entwicklungsarbeit darstellen.

> „Was mich immer ein bisschen hemmt, wenn jetzt irgendwelche Leute dort nur hier melden: Ich habe hier einen Fehler gefunden! Ohne konkret zu sagen,

> warum und was die Ursprünge waren. Also ich finde so unqualifizierte Bug-Reports oder eben auch, (.) irgendwie Sachen, wo jemand (.) etwas nicht selbst bis zu Ende sich überlegt, sondern irgendwie nur ja 'Das ist irgendwie blöd' oder 'Das ist Mist' sagt. Das demotiviert mich dann, [...] da hast du halt irgendwelche Leute gehabt, die haben das mal benutzt, dann ging irgendwas bei ihnen nicht und (.) hast dann einen Fehler gekriegt: Ja, geht nicht! Aber ohne irgendwas. Wozu und warum und dann, wenn die Leute dann auch auf Mails nicht reagieren, dann ist es halt relativ schwierig da was zu tun. Aber, ja, im Großen und Ganzen muss ich sagen, sind die Sachen, gerade, was bei Debian kommt, auch an Fehler-Reports, relativ konstruktiv." (Dokument I-9; Abs. 119)

Fehlerberichte sind nur dann für die Entwicklungsarbeit förderlich, wenn sie auch den Kontext beschreiben. Der Fehler muss reproduzierbar sein, damit seine Ursache eindeutig bestimmt werden kann. Ohne diese Informationen ist ein Fehler nicht zu beheben und der Fehlerbericht damit nutzlos. Dies zeigt, dass sich auch eine relativ einfach erscheinende Aufgabe, wie eine Fehlermeldung nach einem Test, an Anforderungen halten muss, um einen Beitrag zum Projektfortschritt zu leisten.

Soziale Konflikte in der Entwicklungsgemeinschaft

Die wohl gravierendsten Probleme sind die sozialen Konflikte innerhalb der Entwicklungsgemeinschaft. Unterschiedliche Arbeitsweisen, Zielvorstellungen und Kommunikationsweisen der Entwickler:innen treffen aufeinander und bedürfen der Aushandlung. Dieser Prozess verläuft nicht immer störungsfrei, wie die folgenden Beispiele zeigen:

> „Hin und wieder gehen einem schon Sachen auf den Keks. Also ich/ Ich sage mal so, wenn es dann Reibereien sozialer Natur gibt, von wegen, dass man sich mit irgendwelchen Leuten schlecht versteht oder sich halt denkt: Ja, war das jetzt wirklich notwendig? Ich will da jetzt nicht so richtig mit nackten Finger zeigen, weil auch Leute im Team die dann, statt irgendwas zu machen, sich dann halt auch teilweise im Bug-Tracker darüber auslassen: Mimimi[132], das ist doch so blöd. Und dann halt so: Aber ich verstehe das nicht, warum soll ich das machen? Und dann denke ich mir dann so: Junge, halt doch einfach die Fresse, das bringt [uns] so nicht wirklich weiter. So vor solchen sozialen Problemen ist leider auch kein Projekt so richtig gefeit. Das nervt dann halt." (Dokument I-12; Abs. 42)

Neben Konflikten zwischen einzelnen Entwickler:innen, die nicht immer einer Intervention bedürfen, können Konflikte auch so eskalieren, dass die Handlungsfähigkeit der Gemeinschaft beeinträchtigt wird:

[132] Mimimi ist eine Wortschöpfung, die übertriebene Wehleidigkeit oder Gejammer bezeichnet.

„Dann ist aber auch bei mir, glaube ich, im Moment so ein bisschen Unzufriedenheit, wie das überhaupt mit den anderen Entwicklern abläuft. Der ganze Entwicklungsprozess, wie da Absprachen im Moment geführt werden, oder einfach auch nicht, zwischen den Entwicklern oder/ Also bei [FLOSS-Projekt] ist halt auch noch so das Problem, es gibt da halt eine größere Community und sage ich mal zwanzig, dreißig Leute, die ein Grundinteresse an dieser Firmware-Entwicklung haben, teilweise aber einfach selber nichts machen können. Die haben natürlich Vorstellungen davon, wie sich die Firmware entwickeln soll oder die Software. Und dann gibt es da halt drei, vier, fünf Entwickler, die haben auch noch einmal eigene Vorstellungen und (.) dann da diese Kommunikation ist halt schwer. Ja (.) und im Moment bin ich, glaube ich, einfach enttäuscht davon, wie das auch abläuft, weiß nicht/ Dass das nicht wirklich basisdemokratisch abläuft oder man dann nicht auf die verschiedenen Bedürfnisse eingeht oder so, sondern einige Leute einfach das machen, was sie meinen, was richtig ist oder so. [...] Das ist halt auch ein altes Projekt, da gibt es so was öfter mal, das dann manche alte Teilnehmer oder so dann auch mal aufgrund von ihrer Stellung irgendwelche Entscheidungen durchdrücken oder so. Da macht man auch mal negative Erfahrungen. Und die sind dann auch entscheidend für das folgende Engagement oder man hält sich dann vielleicht auch irgendwann aus gewissen Dingen raus oder engagiert sich dann noch viel mehr.“ (Dokument I-2; Abs. 36, 88)

Der Interviewte spricht hier Zielkonflikte zwischen einzelnen Mitgliedern der Entwicklungsgemeinschaft an. Es gibt keinen Konsens über die Methodik der Entscheidungsfindung. Meritokratische und basisdemokratische Entscheidungsstrukturen prallen hier aufeinander und sorgen bei Teilen der Entwickler:innen für Unzufriedenheit. Die vermeintliche Konfliktlösung bei diesem Beispiel erfolgt durch eine Gruppe von Entwickler:innen, die trotz der Polarisierung der Konfliktparteien über den Lösungsweg auf eigene Faust eine Lösung implementieren.

Dieser Ausweg bei einer Stagnation im Prozess der Entscheidungsfindung scheint keine Seltenheit in FLOSS-Projekten zu sein, in denen es keine klaren Entscheidungsstrukturen gibt. Ein ähnliches Vorgehen bei Schwierigkeiten in der gemeinschaftlichen Entscheidungsfindung beschrieb bereits Taubert (vgl. 2008, S. 84f). Nichtsdestotrotz ist eine solche Lösung auch mit Kosten verbunden. In den geführten Interviews wurden auch Zielkonflikte innerhalb von Entwicklungsgemeinschaften beschrieben, die zu einem Ausscheiden vieler Entwickler:innen aus dem FLOSS-Projekt führten. Diese haben dann ein neues FLOSS-Projekt initiiert und die Entwicklungsarbeit nach eigenen Zielvorstellungen auf Grundlage des alten Quellcodes fortgesetzt. Solche Abspaltungen führen nicht immer zu einem Aufbau einer neuen, langfristig wachsenden Entwicklungsgemeinschaft. Dennoch sind sie eine wichtige Möglichkeit für Entwickler:innen, um auf tiefgreifende Veränderungen im FLOSS-Projekt zu reagieren und entsprechend gegensteuern zu kön-

nen. Eine weiterführende Übersicht über bisherige Forschungsarbeiten zu Abspaltungen in FLOSS-Projekten mit ihren Vor- und Nachteilen liefern Gamalielsson und Lundell (2014).

8.2 Problemorientiertes Lernen

Wie schon bei den Interessen für ein FLOSS-Engagement dargelegt wurde, erfolgt das Lernen in Entwicklungsgemeinschaften in erster Linie problemorientiert. Egal, um welche Tätigkeiten es sich konkret handelt, es folgt dem Interesse an einer bestimmten Zustandsänderung an einer Software. Auch wenn das Interesse an der Softwareentwicklung aus Lohnarbeit, Hilfe für Personen aus dem sozialen Umfeld oder aus Spaß am Lernen resultiert, so sind die Handlungen immer auf die Lösung eines konkreten Problems ausgerichtet. In einem Interview wurde die FLOSS-Entwicklung wie folgt zusammengefasst: „Also Problemlösen ist das, was man die ganze Zeit tut. Insofern ist es irgendwie ein nicht hervorzuhebender Prozess.“ (Dokument I-4; Abs. 112).

Die problemorientierte Sichtweise auf Softwareentwicklung findet sich auch strukturell in der Funktion der genutzten Werkzeuge wieder. Bug-Tracker dienen der Sammlung von Problemen, die eine bestimmte Version von Software noch aufweist. Ebenso entspricht die Formulierung von Leistungsanforderungen für die Veröffentlichung einer neuen Softwareversion einem Katalog an Funktionen, die die Software in diesem Entwicklungsstand aufweisen soll. Die problemorientierte Vorgehensweise ist also im Prozess der Softwareentwicklung tief verankert. Sie steuert in weiten Teilen das Vorgehen bei der individuellen Entwicklungsarbeit. Am Ausgangspunkt steht die Problemwahrnehmung, an die sich eine Kette von Handlungen anschließt. Da die überwiegende Mehrheit der Interviewten sich bei bereits bestehenden FLOSS-Projekten in der Weiterentwicklung engagierten, stelle ich zunächst einen typischen Ablauf dar. Dieser geht von einer bereits existierenden Software aus, die Verbesserungspotential hat. Ähnliche Szenarien sind vorstellbar, wenn zum Beispiel falsche Übersetzungen oder fehlerhafte Grafiken korrigiert oder noch nicht implementierte Softwarefunktionen realisiert werden sollen. Das folgende Flussdiagramm zeigt schematisch den Weg von der Problemwahrnehmung bis zur Problemlösung.

Am Anfang vom Entwicklungsprozess steht ein Problem, dessen Lösung die betreffende Person herbeiführen möchte. Wie diese Lösung gefunden wird, hängt stark von den individuellen Kompetenzen und den genutzten Lösungsstrategien ab. Drei Kernfragen sind ausschlaggebend:

1. Wird das Problem als so wichtig eingeschätzt, dass es gelöst werden soll?

   An dieser Stelle gibt es die Möglichkeit, statt der Verbesserung der betreffenden Software auf eine alternative Software auszuweichen. Bei diesem Vorgehen muss sich ein Überblick über existierende Softwarelösungen gemacht werden und diese auf ihr Potenzial für eine Problemlösung hin getestet werden.

2. Gibt es andere Personen, die sich für eine Problemlösung interessieren?

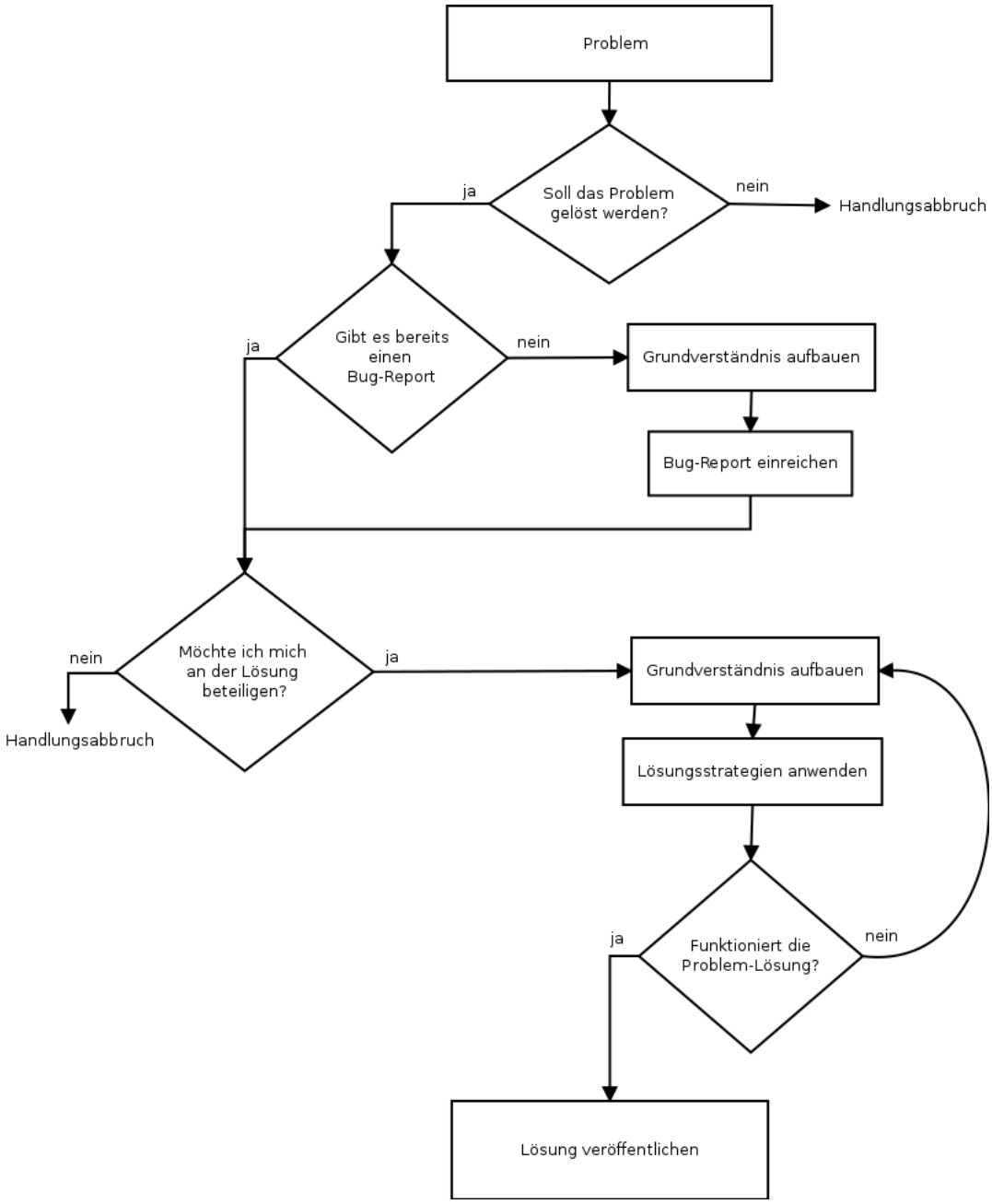

Abbildung 8: Prozess der Problemlösung bei der Softwareentwicklung

Die Suche nach einem Bug-Report steht synonym für das Interesse anderer Nutzer:innen an einer Problemlösung. Ebenso ist eine neue Funktionalität einer Software eine Problemlösung, da sie das Einsatzspektrum einer Software erweitert. Diese wird im Regelfall nicht als Fehlerbericht dokumentiert, sondern beispielsweise über Foreneinträge oder E-Mail an die Projektmailingliste mitgeteilt oder über separate Werkzeuge gesammelt. Ein hohes Interesse an bestimmten Erweiterungen erhöht die Wahrscheinlichkeit, dass diese auch implementiert werden.

3. Möchte ich mich an der Lösung beteiligen?

   Die Antwort hängt stark von der Einschätzung der eigenen Kompetenzen und Ressourcen ab. Wer nicht selbst am Quellcode Veränderungen vornehmen kann, kann dennoch durch qualifizierte Fehlerberichte die Arbeit anderer Entwickler:innen unterstützen. Dies ist oft die Rolle von Softwaretester:innen, die vor der Veröffentlichung einer neuen Softwareversion gezielt die vorgenommenen Änderungen prüfen.

Das Vorgehen bei der Problemlösung ist nicht immer erfolgreich. Auch fehlgeleitete Versuche verbessern das Grundverständnis des Problems, indem sie die Zahl der möglichen Lösungen einschränken. So eignet sich die Person auch bei Fehlschlägen Wissen und Kompetenzen an. Doch kann es an jeder Stelle jedoch aus unterschiedlichen Gründen zu einem Handlungsabbruch kommen. Dies liegt nur selten an der Komplexität der Programmieraufgabe. Oft ändern sich Lebensumstände der Entwickler:innen oder es gibt Unstimmigkeiten in der Projektgemeinschaft, die zu einer Abwanderung von Entwickler:innen zu anderen Projekten oder zur Stagnation bei der Entwicklung führen.

8.3 Lösungsstrategien

Bei der Lösung von Problemen bedienen sich FLOSS-Entwickler:innen unterschiedlicher Strategien. Die in den Interviews beschriebenen Wege zur Problemlösung lassen sich in drei verschiedene Strategien unterteilen. Die Differenzierung erfolgt entlang des Einbezugs anderer Entwickler:innen im Sinne des Lernens im sozialen Kontext. Im Regelfall wird jedoch ein Mix aus den Strategien eingesetzt.

8.3.1 Solitäre Strategie

Zur Problemlösung wird auf eigene Kompetenzen zurückgegriffen. Reichen diese nicht aus, werden im Rahmen einer Internetrecherche zusätzliche Informationen beschafft. Diese Lösungsstrategie eignet sich, um sich ein Grundverständnis aufzubauen. Auf dessen Grundlage kann entschieden werden, ob das Problem selbst gelöst werden kann oder auf externe Informationsressourcen zurückgegriffen werden muss. Die solitäre Strategie stellt oftmals den Beginn des Prozesses der Problemlösung dar. Ist das Problem zu komplex, um allein gelöst zu werden, muss auf eine der anderen Strategien zurückgegriffen werden.

Häufig ist sie bei kleinen FLOSS-Projekten anzutreffen, die nur von einer Person getragen werden oder bei hochspezialisierten Aufgaben, die ein fundiertes Nischenwissen erfordern. Auch bei einmaligen Beiträgen durch eine:n Entwickler:in zu einem Projekt, bei der nur geringfügige Änderungen vorgenommen werden, ist diese Strategie der schnellste und einfachste Weg zur Problemlösung. Diese Person gehört meist nicht zum Kernteam und betreut als Paket-Maintainer:in beispielsweise Spiegelpakete für andere GNU/Linux-Distributionen, die diese Software nutzen.[133]

> „[...] halt einfach auch im Internet suchen, nach Informationen suchen (.) in Suchmaschinen. Ja genau, also (..) aber/ (.) Also gerade wenn man viel macht oder ein tieferes Verständnis davon hat, dann ist es eigentlich so, dass man sehr lange einfach vor dem Problem sitzt, viele Informationen versucht zu sammeln und meistens das Problem eher alleine löst und sich dann noch einmal zur Lösung vielleicht ein Feedback holt." (Dokument I-2; Abs. 44)

Die Hauptaufgabe ist der Aufbau eines Grundverständnisses, um das Problem und den Kontext überhaupt verstehen zu können. Dann kann eine zielgerichtete Suche nach der Lösung erfolgen. Der Aufbau eines Grundverständnisses ist bei vielen Problemen oft ein zyklischer Prozess. Durch Versuch und Irrtum erfolgt eine schrittweise Annäherung an die Problemlösung.

> „Okay, ja da habe ich tatsächlich sehr lange daran getüftelt. Wie ich dann die Webseite da vernünftig aufsetze, wie ich das hinkriege, dass das richtig dargestellt wird. Das Schlimmste war, glaube ich, irgendwie, wie man dieses Thema da, den Seitenstil sozusagen, wie man den da reinbekommt und dann/ Da habe ich sehr, sehr lange daran herumgetüftelt und versucht, wie das geht und war da schon voll am Verzweifeln. Bis ich dann alles Mögliche ausprobiert habe einfach, solange bis es ging. Und ja, (.) an Ressourcen da zur Hilfe gezogen habe ich vor allem irgendwie StackOverflow und die Dokumentation von Read_the_Docs[134] selbst." (Dokument I-3, Abs. 133)

Die Person beschreibt in diesem Beispiel ihr Vorgehen im Rahmen des explorativen Lernens. Sie erlebt eine Differenzerfahrung bezüglich ihres eigenen Wissensstand und dem benötigtem Wissen für die Problemlösung. In einem iterativen Prozess aus Informationssuche, Anwendung und Evaluation arbeitet sie sich schrittweise bis zu einer zufriedenstellenden Problemlösung vor.

[133] GNU/Linux-Distributionen bieten eine Vielzahl verschiedener Softwarepakete an, die auf die Umgebung der betreffenden Distribution angepasst ist. Ändert sich das Softwareoriginalpaket, passt es möglicherweise nicht mehr in eine bestimmte Distributionsversion und muss dafür erneut angepasst werden. Dafür sorgen die Paket-Maintainer:innen der Distribution, die die Entwicklung der Originalsoftware verfolgen und die Kompatibilität gewährleisten.

[134] Read_the_Docs ist eine Webplattform zur Erstellung und Veröffentlichung von Softwaredokumentation.

8.3.2 Selektiv soziale Strategie

Diese Strategie setzt auf die direkte Hilfe durch andere kompetente Personen. Diese werden zu bestimmten Anlässen angefragt, um Antworten auf spezifische Fragen in einem klar abgesteckten Themengebiet zu liefern. Dies kann durch themenbasierte Foren, Chats oder Mailinglisten erfolgen. Dabei wird das Problem und sein Kontext beschrieben und um Hinweise bei der Problemlösung gebeten. Hilfestellung wird oftmals von unbekannten Personen gegeben, die mit der Programmiersprache oder der Softwaretechnologie vertraut sind, jedoch in der Regel keine Verbindung mit dem FLOSS-Projekt haben, wo das Problem aufgetaucht ist.

Alternativ kann Hilfe auch von Personen im sozialen Umfeld gegeben werden, wie es im folgenden Zitat beschrieben ist. Der Übergang zur selektiven sozialen Strategie setzt an dem Punkt ein, an dem der:die Entwickler:iin aus Mangel an Wissen das Problem nicht selbst zu lösen vermag.

> „Wenn ich das nicht verstehe, dann frage ich halt Freunde oder Kollegen, die sich mit Python auch auskennen. [...] Bei Kollegen ist es einfach der Fall, da kann ich mit meinem Laptop zu denen rübergehen. Oder er sitzt im selben Büro, dann frage ich: Hey, hast du diese Exception schon einmal gesehen, kennst du dich damit aus? Wenn sie nicht im gleichen Raum sind, dann entweder per E-Mail oder Telegram[135]. Je nachdem, wie gut ich diese Person kenne.“ (Dokument I-11; Abs. 64 - 68)

Der Entwickler kennt die Kompetenzen der Personen in seinem sozialen Umfeld und spricht sie gezielt an. Sie sind nicht Teil der Entwicklungsgemeinschaft, betreiben das Programmieren aber entweder als Hobby oder als Lohnarbeit. Dennoch unterstützen sie sich bei Problemen gegenseitig aufgrund der sozialen Bindung. Ebenso verhält es sich bei der Anfrage in einem thematisch passendem Forum beziehungsweise anderen internetbasierten Austauschmöglichkeiten, die von Entwickler:innen mit ähnlichen Interessen frequentiert werden. Das verbindende Element ist aber nicht das FLOSS-Projekt, sondern ein gemeinsames Interessengebiet.

Diese Strategie findet in der Regel dann Anwendung, wenn die solitäre Strategie nicht zum erwünschten Ergebnis führt. Besonders im Internet ist die Hilfsbereitschaft anderer Entwickler:innen an die Erwartung geknüpft, dass die hilfesuchende Person zuvor selbst relevante Anstrengungen unternommen hat, um das Problem selbstständig zu lösen.

8.3.3 Kollaborative Strategie

Bei der kollaborativen Strategie gibt es im Unterschied zur selektiv sozialen Strategie eine gemeinsame FLOSS-Entwicklungsgemeinschaft. Hier verbinden sich die Interessen der Entwickler:innen und bündeln sich in dem gemeinschaftlichen Engagement für eine bestimmte Software. Anwendung findet diese Strategie entsprechend nur bei FLOSS-

[135] *Telegram* ist ein Instant Messenger.

Projekten, wo es einen Kern von mehrerer Entwickler:innen gibt, die über längere Zeit zusammenarbeiten. Die Mitglieder einer Entwicklungsgemeinschaft unterstützen sich gegenseitig bei dem Ziel, die betreffende Software zu verbessern. Durch die enge Zusammenarbeit gibt es bei allen Beteiligten ein Überblickswissen über die Software samt ihrer Funktionsweise. Einzelne Entwickler:innen besitzen bei bestimmten Teilaspekten der Software ein spezifisches Fachwissen, was den anderen bekannt ist. Diese können durch gezielte Nachfragen darauf zurückgreifen und sich so bei Problemen gegenseitig helfen. Diese Lösungsstrategie findet nicht nur internetgestützt statt. Vor allem größere FLOSS-Projekte organisieren eigene Konferenzen oder Projekttreffen, bei denen den Projektmitgliedern gezielt die Möglichkeit zur gemeinsamen Arbeit an dem Projekt gegeben wird:

> „Das war ein Sprint, wie sich dies nennt [...] um quasi ein Problem zu lösen. [...] an der Stelle ging es darum, das alte System des kollaborativen Entwickelns von Debian. Also, da gab es einen Server der Dinge tut, der aber auf einer alten technischen Basis stand und irgendwie ganz schlecht wartbar war und seinem technischen Ende entgegen ging (.) an Software. Und da gab es einen Sprint, auf das sich Menschen zusammengefunden hatten, um dann halt die nächste Entwicklungsumgebung vorzubereiten und/ Ja, also quasi das Bootstraping[136], das Loslegen am Anfang, Technik aussuchen und dann loslegen. Und da bin ich dann hingefahren, um dort als Admin quasi meine Dinge einzubringen.“ (Dokument I-4, Abs. 184 - 188)

Ein Sprint ist ein Verfahrensschritt in dem Projektmanagementsystem *Scrum*, welches ursprünglich aus der Softwareentwicklung stammt. Ein kleines Team von Entwickler:innen fand sich in diesem Beispiel zusammen, um gemeinsam diesen Server neu anzupassen. Teilweise kannten sie sich vorher untereinander. Im Rahmen dieses Treffens konnten die Entwickler:innen ihre Fachkenntnisse einbringen und kollaborativ das Ziel umsetzen. Vorteile einer solchen Arbeitsumgebung liegen in der störungsfreien, gemeinsamen Arbeitszeit der Entwickler:innen und den kurzen Kommunikationswegen. Nicht unbedeutend ist auch der Aufbau von sozialen Beziehungen zwischen den Entwickler:innen. Besonders mit Blick auf das zukünftige Engagement dienen solche Treffen dem Aufbau einer Gruppenidentität.

Bei kleineren FLOSS-Projekten ohne finanzielle Förderung für Treffen der Entwicklungsgemeinschaft dominieren internetgestützte Kommunikationskanäle. Meist sind es ein bis zwei Kanäle, die sich in der Entwicklungsgemeinschaft durchsetzen.

> „Wir haben zum einen den IRC-Chat, wo teilweise Entwicklungsdiskussionen stattfinden, wo im Moment noch die Vorstandstreffen stattfinden, die quasi einmal im Monat ein großes Community-Ereignis darstellen. Dann gibt es den neuen Chat-Server, den Rocket.Chat[137]-Server, wo wir mehrere Kanäle

[136] In diesem Kontext beschreibt das 'Bootstraping' Start und Durchführung einer umfassenden Softwareinstallation.

[137] 'Rocket.Chat' ist eine FLOSS-Kommunikationsplattform, die verschiedene Dienste wie zum Beispiel Chat oder Dateiaustausch für das kollaborative Arbeiten bereitstellt.

eingerichtet haben. Wo aber nicht alle Entwickler mit hin umziehen wollen vom IRC leider und deswegen ist es noch nicht unser Standardkanal. Deswegen ist es da halt inzwischen weiterhin aufgespalten und dann findet viel Diskussion auch in den Merge-Requests[138] statt. Auf GitLab[139], wo dann über den Code gesprochen wird oder auch in den Fehlerberichten findet auch relativ viel Kommunikation und Diskussion statt. Nicht irgendwie dass nur einer der Kanäle einen besonderen Schwerpunkt hätte, ich glaube es ist relativ gleich verteilt.“ (Dokument I-3; Abs. 189)

Das Zitat zeigt, dass sich Kommunikationsgewohnheiten in dem FLOSS-Projekt ändern können und sich in einer Anpassung des Repertoires der Praxisgemeinschaft niederschlagen. Welche konkreten Kommunikationskanäle und Inhalte sich innerhalb einer Entwicklungsgemeinschaft etablieren, ist von den Vorlieben der Entwickler:innen und der Größe der Entwicklungsgemeinschaft abhängig. Die Inhalte reichen von fast ausschließlicher Kommunikation zur Entwicklungsarbeit, wie dies bei qBittorrent der Fall war, bis hin zu Diskussionen über nicht entwicklungsrelevante Inhalte, wie zum Beispiel Hobbys, die neben der eigentlichen Entwicklungsarbeit die sozialen Bindungen innerhalb der Gemeinschaft stärken.

8.3.4 Strategiemix

Im folgendem Interviewausschnitt wird das prinzipielle Vorgehen bei der Problemlösung beschrieben. Die Person ist Teil eines größeren FLOSS-Projektes mit mehreren Dutzend Entwickler:innen.

„Also, einmal hast du einfach Probleme, die auftauchen, sozusagen Bugs in deiner Software, wo du vielleicht nicht verstehst, warum der da ist. Oder erst einmal, es funktioniert irgendwas nicht, dann gibt es/ (.) Also muss man natürlich erst einmal ein Grundverständnis überhaupt von den ganzen Abläufen haben oder das entwickeln. Da muss man sich halt ganz genau angucken, was diese Firmware oder Software halt macht. Wie, also gerade bei [Name vom FLOSS-Projekt] sind das mehrere Komponenten, die irgendwie interagieren. Da muss man das komplette technische Thema verstehen. Genau, sozusagen Informationen darüber sammeln, wie das funktioniert und was da nicht funktioniert und dann versucht man das erst einmal irgendwo festzuhalten halt, in so einem Issue-Tracker oder so. Und versucht dann natürlich, das auch mit den anderen Entwicklern zu besprechen, also wenn man jetzt gar nicht selber weiterkommt, dann versucht man so viele Informationen wie möglich zu sammeln, damit dann eine andere Person vielleicht arbeiten kann. Dann gibt

[138] Ein Merge-Request bei 'GitLab' ist das Pendant zum Pull-Request bei 'Git'.

[139] 'GitLab.com' ist eine FLOSS-Plattform mit einer Weboberfläche für die Versionsverwaltung Git mit erweitertem Funktionsumfang.

es viele so Mailinglisten zu den verschiedenen Projekten, zu den verschiedenen Softwares die da laufen, wo man dann auch um Hilfe bitten kann. Das ist immer/ Also je nachdem, wie komplex das Problem ist, kommt man da auch nicht unbedingt weiter. Aber, also es ist halt eine Möglichkeit. Dann gibt es halt auch viele so Chats zu den Projekten, wo man auch noch mal direkt in Interaktion treten kann. Ja, oder halt einfach auch im Internet suchen, nach Informationen suchen (.) in Suchmaschinen. Ja genau, also (..) aber/ (.) Also gerade wenn man viel macht oder ein tieferes Verständnis davon hat, dann ist es eigentlich so, dass man sehr lange einfach vor dem Problem sitzt, viele Informationen versucht zu sammeln und meistens das Problem eher alleine löst und sich dann noch einmal zu Lösung vielleicht ein Feedback holt." (Dokument I-2; Abs. 44)

Die Person beschreibt verschiedene Strategien, auf die sie bei der Problemlösung zurückgreift. In einem mehrstufigen Prozess setzt sie alle drei beschriebenen Strategien ein. Sie versucht so weit wie möglich, allein das Problem zu lösen. Scheitert dieses Vorgehen, werden weitere Personen um Unterstützung gebeten. Erst kurz vor der finalen Implementation der Lösung fordert sie sich Rückmeldung zum Lösungsansatz innerhalb der eigenen Projektgemeinschaft ein.

Bedingung für den Strategiewechsel

Dieser Ansatz, so weit wie möglich ohne fremde Hilfe eine Lösung zu finden, ist in der FLOSS-Bewegung weit verbreitet. Bei der Kommunikation in Foren und Mailinglisten wird implizit auf die Einhaltung der Netiquette[140] geachtet. Diese beschreibt Verhaltensregeln bei der Internetkommunikation. Ziel ist es, einen respektvollen und effizienten Umgang untereinander zu gewährleisten. Dies beinhaltet, dass vor dem Stellen einer Frage zunächst andere verfügbare Informationsressourcen durchsucht werden sollen. Erst wenn die Antwort nicht aus eigener Kraft gefunden werden kann, soll um Hilfe gebeten werden (vgl. Hambridge 1995). Diese Handlungsnorm schlägt sich auch in dem folgenden Zitat nieder. Befragt nach den genutzten Ressourcen bei der Problemlösung, antwortete diese Person:

„Hauptsächlich Google und StackOverflow und bei allem, wo ich dann wirklich nicht weiterkam/ Also ich habe halt so viel wie möglich versucht selber rauszukriegen. Und wenn ich es wirklich dann nicht geschafft habe, dann habe ich halt den Admin[141] gefragt. Und der hat mir es dann auch meistens so erklärt, dass ich es verstanden habe. Spätestens nach dem dritten Mal." (Dokument I-3; Abs. 112)

[140] Die Netiquette ist ein Kofferwort aus 'Net' und 'Etiquette', die in den Internetstandards Request for Comments (RFC) Nr. 1855 erstmals formal verfasst wurde.

[141] Admin ist die Kurzform für Administrator:in und bezeichnet eine Person, die ein Computersystem betreut.

Diese Norm bei der Kommunikation markiert die Schwelle vom Übergang von der solitären Strategie zu den beiden anderen Lösungsstrategien. Das eigentliche Ziel der Netiquette ist zu verhindern, dass neue Gemeinschaftsmitglieder mit bereits beantworteten Fragen die Kommunikation innerhalb der Entwicklungsgemeinschaft behindern und damit Zeit für die wichtigen Diskussionen fehlt (vgl. Bergquist 2020, S. 232). Aus der Perspektive des Lernens heraus, stellt der Strategiewechsel auch eine Abwendung vom Lernen durch Versuch und Irrtum hin zum Lernen durch sozialen Austausch dar.

8.4 Ressourcen

Nachdem die Strategien bei der Suche nach einer Problemlösung aufgezeigt wurden, soll jetzt der Fokus auf Informationsressourcen gesetzt werden. Dieser Abschnitt beleuchtet die verschiedenen Ressourcen, die von den Entwickler:innen bei Problemen genutzt werden, die sie nicht mittels eigener Kompetenzen lösen können.

> „Ein bisschen gecoded habe ich auch, aber ehrlich gesagt habe ich bei dem Projekt sehr wenig tatsächlich an dem bestehenden Code geändert, was leider im Moment auch so das Hauptproblem ist. [...] wir haben eigentlich von dem Projekt nicht so richtig Ahnung, wie das intern funktioniert.“ (Dokument I-12; Abs. 14)

Dies ist ein Beispiel dafür, dass Entwicklungsarbeit je nach Kenntnisstand der beteiligten Personen immer auch ein Handeln in Situationen erfordert, in denen kontinuierliche Informationen akquiriert werden müssen, um sich ein Grundverständnis aufzubauen. Obwohl viele Entwickler:innen im Rahmen von formalen Lernprozessen Fachkompetenzen erworben haben, sind viele Probleme bei der Softwareentwicklung so spezifisch, dass der eigene Wissensbestand nicht ausreicht. Um sich ein Grundverständnis für die Problemlösung zu erarbeiten, werden verschiedene Informationsquellen genutzt. Das Spektrum reicht von didaktisch aufbereiteten Lernmaterialien bis hin zum Austausch mit kompetenten anderen.

8.4.1 Quellcode lesen und verstehen

Oft erden auch anhand vom Quellcode ähnlicher Software Lösungsideen für das eigene Projekt abgeleitet. Diese Art des Lernens ist eines der vielen Vorteile von FLOSS, da der Quellcode selbst als Lernmaterial genutzt werden kann.

> „Bei [FLOSS-Projekt] habe ich zum Beispiel eine Weile damit gekämpft, wie ich Polygone mit einem Offset versehen kann, um halt Pocketing oder irgendwelche anderen interessanten Sachen zu machen. Und das stellte sich für mich als sehr herausfordernd dar, weil das halt Mathematik ist, mit der ich wenig zu tun habe. Und da habe ich ein bisschen bei anderer Software geguckt, wie andere Software das so tut, also andere Freie Software eben und

> auch geschaut, was ich da so halt verwenden könnte. Was sich an der Stelle schwierig herausstellte, weil es aus einem anderen Ökosystem kam, also es wäre mit gutem Aufwand verbunden gewesen, vorhandene Software dort zu integrieren. [...] Ja also einfach nach anderer Software gucken, die so etwas tut, mir dort Ideen abschauen.“ (Dokument I-4; Abs. 104)

In diesem Beispiel soll eine Funktionalität implementiert werden, deren Umsetzung in einen Algorithmus dem Entwickler Probleme bereitet. Seine gewählte Lösung ist das Lesen vom Quellcode einer anderen Software, bei der diese Funktion bereits implementiert wurde.

Neben diesen konkreten Problemen in der Entwicklungsarbeit hat das Lesen und Verstehen von Quellcode besonders beim Erlernen von Programmiersprachen eine herausragenden Bedeutung. Einige Entwickler:innen erlernten anhand des Quellcodes von Computerspielen in ihrer Kinder- und Jugendzeit das Programmieren. Vorhandener Quellcode diente dabei als Modell, um ihn an eigene Bedürfnisse anzupassen oder daran die Umsetzung einer bestimmten Funktionalität zu erlernen. Wie wichtig dabei die Praxis des Programmieren ist, erklärt dieser Entwickler:

> „Also ich denke halt schon, dass du irgendwie Programmieren oder so/ Also die Softwareentwicklung ist halt so eine Art Handwerk, würde ich sagen. Das muss man halt einfach ausüben. Und (.) Freie Softwareprojekte, (.) das ist halt ein guter Ort zum Üben oder so. Du kannst natürlich Informatik studieren und dann programmierst du auch ein bisschen im Studium, aber wenn du darüber hinaus nichts machst, dann hast du im Endeffekt dieses Handwerk vielleicht nicht gelernt oder sage ich mal (.) ja, also du kannst die Grundlagen ein bisschen, aber hast einfach keine Übung. Und ich glaube, zu so einem Handwerk gehört einfach die Übung. Programmieren ist das dann. Da braucht man einfach Übung für. Man muss viel Code gelesen haben, gesehen haben, selbst geschrieben haben und ja, Freie Software hilft dabei natürlich.“ (Dokument I-2; Abs. 130)

Er betont die Bedeutung vom Lernen in der Praxis, wofür ein FLOSS-Projekt seiner Meinung nach eine geeignete Lernumgebung ist. Dadurch erfolgt das eigentliche Lernen des Programmierens nicht nur auf rein theoretischer Basis, sondern ist eingebunden, also situiert, in den Kontext eines FLOSS-Projektes.

8.4.2 Dokumentation und Tutorials

Um das Erlernen einer Programmiersprache zu erleichtern, nutzen viele Entwickler:innen auch didaktisch aufbereitete Lehrmaterialien. Das Spektrum reicht von Lehrbüchern zu spezifischen Programmiersprachen, die digitalisierte Variante als E-Book, Online-Tutorials bis zu eigens konzeptionierten Webinaren[142].

[142] Das Wort Webinar setzt sich aus den Wörtern ’Web’ und ’Seminar’ zusammen. Synonym wird auch die Begriffe Web-Seminar oder Online-Kurs gebraucht.

> „[...] es ging damit weiter, dass ich viel auf der Webseite oder mit der Webseite gearbeitet habe. Dadurch, dass ich die Webseite übersetzt habe, habe ich auf der Webseite endlos viele Bugs gefunden. Und weil ich auch gleichzeitig irgendwie noch so einen Online-Python-Kurs gemacht hatte gerade und die Webseite zufällig in Python geschrieben ist, wurde ich dann beim Bugs melden auch gleich gefragt: Ja, willst du denn nicht auch mal einen reparieren? Und so ging es dann halt weiter.“ (Dokument I-3; Abs. 65)

Auch hier erfolgte das Erlernen der Programmiersprache in der direkten Anwendung in einem FLOSS-Projekt. Bei fortgeschrittenen Kenntnissen rücken andere Informationsquellen in den Vordergrund: Dokumentationen von Soft- und Hardwarekomponenten. Für die Softwareentwicklung sind dies in erster Linie die Dokumentation der Programmbibliotheken. Sie entsprechen einer Sammlung von Unterprogrammen oder Routinen, die eine spezifische Funktion erfüllen und von Programmen für diesen Zweck integriert werden können. So können Standardfunktionen modular in den Quellcode eingebunden werden, ohne dass diese jedes Mal neu implementiert werden müssen.

> „Ah, das sind meistens die Dokumentationen von den Schnittstellen, die man da benutzt, also das sind halt die technischen Dokumentationen der anderen Softwarebibliotheken, die man da dann einbindet. Also ja (.) das sind halt im Endeffekt wieder offen verfügbare/ Also zugängliche Informationsquellen, die dann meistens zu einer Lösung führen. [unverständlich] immer wieder diese technischen Dokumentationen, die einem dann weiterhelfen. Und wenn das nicht hilft (.) der Blick in den Quellcode, um dann die fehlenden Dokumentation am Code abzulesen.“ (Dokument I-7; Abs. 64)

Wie die technische Dokumentation von Programmbibliotheken sind auch die Referenzen zu den jeweiligen Programmiersprachen in einem vergleichbar nüchternen Stil gehalten. Diese sind nicht didaktisch aufbereitet und beschreiben prägnant die Funktion und Nutzungsweise von Programmierkonstrukten. Sie dienen dem schnellen Nachschlagen beim Programmieren. Viele Entwickler:innen haben sich mehrere Programmiersprachen angeeignet, die jeweils unterschiedliche Semantik und Syntax besitzen. Aus diesem Grund ist eine Programmierreferenz eine wichtige Informationsquelle beim Programmieren.

8.4.3 Archivierte Forenbeiträge

Bei komplexeren Fragen werden thematisch passende Foren konsultiert. Auf ihnen können Programmierer:innen Fragen stellen die von anderen Nutzer:innen der Foren beantwortet werden. Die Forenbeiträge sind öffentlich und werden über einen langen Zeitraum gespeichert. Dadurch werden die Foreninhalte auch von Suchmaschinen indexiert, welches die folgende Lösungsstrategie bei Problemen ermöglicht:

> „Na ja, manchmal gibt es halt/ Also es ist nie so, dass ich gezielt zu einer Plattform hingehe und sage: Ich möchte jetzt hier irgendwie Informationen

haben. Außer ich arbeite mit einer konkrete API[143] zusammen, die eine Dokumentation hat. Dann ist es halt immer eine Google-Suche. Und da ist es in den meisten Fällen StackOverflow, aber es gibt auch andere XY-Overflow-Projekte[144], die dann mit hochkommen. Aber neben StackOverflow gibt es noch ServerOverflow[145] oder so. Also für verschiedene Bereiche der Softwareentwicklung gibt es Overflow-Plattformen.“ (Dokument I-13; Abs. 110)

Die in den *Overflow*-Foren gespeicherten Beiträge sind so reichhaltig, dass für die meisten Problembeschreibungen bereits Lösungswege dokumentiert sind. Dadurch ist es nicht nötig, selbst in einem neuen Forenbeitrag das eigene Problem zu beschreiben. Durch eine präzise Suchanfrage in einer Suchmaschine wird oft direkt der passende Beitrag in einem Overflow-Forum in den Suchergebnissen angezeigt. Insofern kann die Suche in Foren auch im Rahmen der solitären Strategie angewendet werden, wenn das gesuchte Problem bereits in der Vergangenheit gelöst wurde. Der Stellenwert der Overflow-Foren bei der Softwareentwicklung muss als bedeutend klassifiziert werden, da diese Informationsquelle in fast jedem Interview erwähnt wurde.

8.4.4 Kommunikation außerhalb des FLOSS-Projektes

Eine weitere Kategorie von Ressourcen ergibt sich aus dem sozialen Netzwerk der Entwickler:innen. Dies kann das soziale Umfeld des:der Entwickler:in, die Entwicklungsgemeinschaft oder der Kontakt zu Entwickler:innen außerhalb der eigenen Entwicklungsgemeinschaft sein.

Die soeben angesprochenen Overflow-Foren können natürlich auch aktiv als Kommunikationsplattform genutzt werden. Wurde das eigene Problem in der Vergangenheit noch nicht zufriedenstellend gelöst, ermöglichen sie den Hilfesuchenden, selbst Fragen zu verfassen. Die Antworten können nach ihrem Nutzen für die Problemlösung bewertet werden. Je nützlicher die Antworten, desto näher werden sie unter der Frage angezeigt. Insbesondere bei der Suche nach archivierten Problemschilderungen verbessert dies die Übersichtlichkeit. Während die Overflow-Foren nach bestimmten Programmiersprachen oder Soft- und Hardwareumgebungen strukturiert sind, existieren auch andere Kommunikationskanäle zwischen Entwickler:innen. Dies können beispielsweise Mailinglisten oder Chat-Kanäle sein. Diese fungieren, wie auch die Foren, als Frage-Antwort-Medium zwischen den dort aktiven Entwickler:innen.

Neben diesem strukturierten Austausch mit eigens dafür eingerichteten Kommunikationskanälen existieren auch Formen des direkten Austauschs zwischen Entwickler:innen.

[143] Das Akronym API steht für 'Application Programming Interface'. Dies ist eine Programmierschnittstelle, um von einem Softwaresystem auf ein Anderes zuzugreifen.

[144] Die von der Person intendierte Bedeutung von 'XY' steht als ein Platzhalter für eine der diversen Overflow-Foren.

[145] Die Person meint das Forum https://serverfault.com, das ebenso wie https://stackoverflow.com von der gleichen Firma betrieben wird. Darüber hinaus bietet die Firma noch zahlreichen andere Foren-Webseiten zu diversen Softwaresystemen wie z.B. Spieleentwicklung, Webadministration, Datenbanken und diverse andere Soft- und Hardwareumgebungen.

Sei es durch direkte Kontakte mit thematisch interessierten Freund:innen und Kolleg:innen oder im Rahmen von Strukturen, die einen solchen Austausch befördern. Eine solche Struktur sind beispielsweise *Hackerspaces* beziehungsweise *Makerspaces*. Die Begriffe werden oft synonym gebraucht, weisen allerdings bei genauerer Betrachtung strukturelle Unterschiede auf. Auch die Bezeichnungen *FabLab*[146], *TechShop* oder *Repair Café* werden genutzt. Die Schnittmenge zwischen diesen Netzwerkstrukturen ist das selbstorganisierte informelle Lernen.

Diese können als Lernraum für selbstorganisiertes Lernen nach dem Grundsatz *Learning by doing* betrachtet werden. Sie stellen einen organisatorischen Rahmen dar, um entweder alleine oder gemeinsam ein Vorhaben umzusetzen, wofür gelernt werden muss (vgl. Schön & Ebner 2020, S. 35). Diese Orte dienen auch der kollaborativen FLOSS-Entwicklung, wie es das folgende Zitat zeigt:

> „Ja, es gibt halt immer wieder diese Momente, wo man mit irgendeinem Problem da steht und eigentlich irgendwie sich selber damit rumschlagen würde. Und im nächsten Moment sitzen drei Leute neben dir, die versuchen, deine Probleme zu lösen, weil sie es halt einfach mal interessant finden. Wobei das alles eher eigentlich weniger Open-Source-spezifisch und eher mehr Hacker-Community-typisch ist. [...] ich finde den Austausch, den persönlichen Austausch mit den Leuten immer ein bisschen produktiver, als wenn man sich dann immer nur mal mailt oder so. Weil man in so einem Gesprächsfluss noch einmal viel eher noch einmal Sachen, Facetten erkennt zusammen oder doch noch reinrutscht.“ (Dokument I-7; Abs. 80, 88)

Ein Vorteil einer solchen Lernumgebung ist die direkte Interaktion zwischen den Entwickler:innen. Dies bietet aber nicht nur der Hackerspace sondern auch Konferenzen. Die Person betont die Bedeutung des direkten Austauschs mit anderen Entwickler:innen. Auch wenn dieser nicht immer auf eine konkrete Problemlösung hin ausgerichtet ist, hilft der Austausch auf Konferenzen, neue Entwickler:innen für ein FLOSS-Projekte zu begeistern.

8.4.5 Kommunikation innerhalb des FLOSS-Projektes

Die zweite hier untersuchte Form der Kommunikation zwischen den Entwickler:innen spielt sich innerhalb der Entwicklungsgemeinschaft selbst ab. Bevor die verschiedenen Kommunikationskanäle beschrieben werden, soll zunächst einmal die Funktion der Kommunikation innerhalb der Gemeinschaft dargestellt werden. Im Gegensatz zur Auswertung der Kommunikationsinhalte bei der Beitragsanalyse in Kapitel 6.5.3 *Themen auf den Mailinglisten*, lassen sich bei den Interviews auch die Wirkung der Kommunikation auf die Entwickler:innen selbst herausstellen. Zusätzlich bieten die Interviews auch

[146] FabLab steht für 'Fabrication Laboratory'.

Rückschlüsse über Kommunikationskanäle, die bei der Analyse von der Projektkommunikation beim qBittorrent-Projekt nicht berücksichtigt werden konnten.

Zugang zu verteiltem Wissen

Die praktische Bedeutung der Entwicklungsgemeinschaft beschreibt diese Person für ihre Entwicklungsarbeit wie folgt:

> „Also wenn man halt in so große Code-Basen reinguckt, die halt schlecht dokumentiert sind, dann kann man sich natürlich stundenlang festfressen und versuchen, das Gesamtkonstrukt zu verstehen. Aber Projekte sind manchmal so riesig, oder machen so viel, dass man den kleinen Teil, den man selber nutzt/ (.) Ja das ist halt so die Suche nach der Nadel im Heuhaufen. Und da kommt dann halt meistens die Community oftmals in Form von Mailinglisten mit [unverständlich] zum Einsatz, wo man dann halt hingeht und: Ja, ich habe hier folgendes Problem und ich habe da und da schon geguckt aber ich sehe halt die Stecknadel nicht. Die einem dann halt meistens entweder selber direkt die Lösung liefern können oder auf jeden Fall in die richtige Richtung schubsen, um dann halt zu sehen, dass man dann halt selber irgendwie in der Lage ist, das Problem dann auch (.) zu finden, wo es denn eigentlich hakt.“ (Dokument I-7; Abs. 68)

Hier wird auf das verteilte Wissen der Gemeinschaft verwiesen. Im Gegensatz zur Kommunikation außerhalb der Entwicklungsgemeinschaft kennen die Projektmitglieder die inneren Strukturen und den geschichtlichen Kontext, in dem sie entstanden sind. Dieses Wissen kann entweder bei einem konkreten Problem angefragt werden oder liegt in irgendeiner Form als gemeinsames Repertoire vor. Ähnlich wie die archivierten Forenbeiträge bieten auch Mailinglistenarchive diese Möglichkeit. Anhand vergangener E-Mail-Diskussionen können Positionen und Argumente auch von den Entwickler:innen nachvollzogen werden, die damals nicht an den Diskussionen beteiligt waren. Insofern kann die Entwicklungsgemeinschaft sehr viel detaillierter bei Problemen unterstützen, als es für Entwickler:innen außerhalb des eigenen FLOSS-Projektes möglich wäre.

Feedback zum Quellcode / Code-Review

Dass der Quellcode an sich bereits als Lernmaterial genutzt wird, wurde bereits herausgearbeitet. Lernen durch Quellcode kann auch dadurch realisiert werden, dass der Quellcode kollaborativ in der Entwicklungsgemeinschaft entwickelt wird. Mehrere Entwickler:innen unterstützen sich dann gegenseitig bei der Implementation:

> „Für mich selbst dann auch noch aus Eigennutz, weil ich habe damit Programmieren mehr oder weniger gelernt und ich bin sehr viel besser geworden dadurch, dass es Freie Software war, andere Leute einen Patch geschrieben haben oder mir Feedback geben. Das wäre einfach nicht möglich, wenn ich

> dies als kleines eigenes Projektchen (..) versuche Closed-Source zu/ Ja, nicht einmal wirklich zu veröffentlichen oder nur Binary Code herzugeben, weil dann würde ich nie Feedback über Code-Qualität bekommen.“ (Dokument I-11; Abs. 20)

Andere Entwickler:innen geben also Rückmeldung zum eingereichten Quellcode. Entweder in Form von Hinweisen oder direkt in Form von korrigiertem Quellcode als Bugfix oder Patch. Mit Hilfe eines solchen korrektiven Feedbacks können Entwickler:innen ihre technischen Kompetenzen verbessern und anhand ihrer Fehler lernen. Solch ein Code-Review durch andere Entwickler:innen ist besonders bei sicherheitsrelevanter Software ein etabliertes Verfahren in der Softwareentwicklung. Es ist eine Methode der analytischen Qualitätssicherung.

Koordinierung der gemeinschaftlichen Unternehmung

Diese Funktion der Kommunikation dient nicht direkt der Lösung eines konkreten Problems. Dennoch ist sie wichtig, um die Entwicklungsgemeinschaft handlungsfähig zu machen.

> „Und (.) wir haben ja dort bei Debian auch sehr viele spezielle Mailinglisten. Es gibt welche für das Python-Team, dann für/ eigentlich für jegliches Team, was mit irgendeinem größeren Set an Packages hantiert. Dann gibt es Mailinglisten für Nutzer-Support (.) und Übersetzung, alles mögliche also (.) das ist ja auch ein Riesenprojekt, also Debian sind ja über tausend Leute. (.) Da ist es schon sehr/ Werden die Mailinglisten schon sehr intensiv genutzt. Auch solche Sachen wie größere technische Entscheidungen werden da diskutiert. Oder (.) auch Sachen wie bei der Projektleiterwahl, da stellen sich die Kandidaten auf den Mailinglisten vor.“ (Dokument I-9; Abs. 134)

Die strukturelle Differenzierung ist mit der Projektentwicklung gewachsen. Ebenso hat sich eine bestimmte Kultur der gemeinsamen Entscheidungsfindung etabliert. Diese soll sicherstellen, dass die Softwareentwicklung ohne größere Störungen erfolgen kann. In diesem Beispiel hat sich eine demokratische Führungskultur entwickelt. Wie bereits beschrieben, ist dies von Projekt zu Projekt unterschiedlich.

Genutzte Kommunikationskanäle

Ähnlich wie der Modus der Entscheidungsfindung sind auch die genutzten Kommunikationskanäle der Entwicklungsgemeinschaft vor dem Hintergrund der jeweiligen Projektgeschichte entstanden. Als Erfahrungswert kann festgehalten werden, dass sich mit ändernder Anzahl an Entwickler:innen in der Gemeinschaft auch Anforderungen an die Kommunikationskanäle ändern. Bei wenigen Entwickler:innen reichen E-Mail und Chat aus. Vergrößert sich die Entwicklungsgemeinschaft, braucht es zusätzliche Kommunikationskanäle, um die Kommunikation von größeren Gruppen übersichtlich zu gestalten. Bezüglich der Frage nach wichtigen Kommunikationskanälen, antwortete diese Person:

> „Also ich denke auf diesen jährlichen Konferenzen von den größeren Projekten. Und (.) was so über das Jahr verteilt ist, ist primär IRC, also Chat. [...] Ansonsten bei [FLOSS-Projekt], gerade haben wir uns primär schon direkt per Mail ausgetauscht, [Name PM] und ich oder eben über den Bug-Tracker. Wenn da irgendwelche Nutzer was geschrieben haben, haben wir dann halt gleich den Bug-Tracker benutzt, um dort mit den Fehlern zu kommunizieren, der lässt sich halt auch per E-Mail ansprechen, aber ist an sich nicht die Mailingliste. Bei Debian ist die Mailingliste schon so, dass man auch größere Projektziele, die über einzelne Packages hinausgehen, diskutiert.“ (Dokument I-9; Abs. 130 - 134)

Besonders für kleine Projekte sind die Kommunikationskanäle der FLOSS-Portale attraktiv. Wie im Zitat angedeutet, finden dadurch viele Diskussionen problembezogen auf dem Bug-Tracker statt. Diese Veränderung der Kommunikation durch FLOSS-Portale beschreibt dieser Entwickler vor dem Hintergrund seiner Erfahrungen:

> „Bei anderen Systemen wäre es ja ebenso, also GitHub ist da ja nicht/ Also die Technik der/ Es gibt keinen gemeinsamen Kommunikationskanal, sondern es gibt nur das Einreichen von Issues. Also in modernen Welten werden ja auch eben Ticket-Systeme[147] dazu verwendet, eine Frage zu stellen, was halt immer so ein bisschen absurd ist, weil früher hätten die Menschen eine Mail an die Mailingliste geschickt oder eine Frage im IRC-Kanal gestellt. Jetzt machen sie ein Ticket auf und stellen eine Frage, das ist halt ein bisschen schräg. Das ist aber so ein bisschen die Konstruktion dieser Kommunikationskanäle. Das ist halt immer so ein bisschen eine/ Ja also es fokussiert die Leute darauf, dass sie die Plattform benutzen und nicht darauf, dass sie miteinander reden.“ (Dokument I-4; Abs. 160)

Die Kritik der Person zielt auf die kommerzielle Ausrichtung der FLOSS-Portale ab. Deren Geschäftsmodelle bauen darauf auf, dass möglichst viele Entwickler:innen das Portal nutzen. Die dadurch entstehenden Vermarktungsmöglichkeiten bilden dann die Grundlage ihres Geschäftsmodells. Dies ist wahrscheinlich auch einer der Gründe, warum größere Projekte dazu tendieren, eigene Kommunikationskanäle wie Foren, Chats oder Mailinglisten einzurichten. Dieser zusätzliche Aufwand wird von kleineren Projekten zunächst vermieden, um sich auf die Entwicklungsarbeit konzentrieren zu können. Ebenso können FLOSS-Projekte ab einer kritischen Größe auch Möglichkeiten für physische Treffen der Entwickler:innen finanzieren. Diese dienen dem gegenseitigen Kennenlernen, dem Lösen von Problemen und dem Austausch über die zukünftige Entwicklung des Projekts.

[147] Ein Ticket-System dient der Verwaltung von Problemanfragen oder Aufgaben. Ein Fehlerbericht als sogenannter Bug oder Issue wird in einem solchen System als zu erledigende Aufgabe erfasst und nach erfolgreicher Bearbeitung wieder gelöscht.

8.5 Zusammenfassung

Die Befragung der Entwickler:innen liefert eine umfassende Deskription über die Art und Weise, wie beim Engagement in FLOSS-Projekten gelernt wird. Anhand der Darstellung des heterogenen Aufgabenfeldes von FLOSS-Entwickler:innen zeigt sich, dass neben dem eigentlichen Programmieren selbst, eine Vielzahl anderer Bereiche existieren, in denen ein Kompetenzaufbau zu erwarten ist. Eine ausführliche Betrachtung dieses Tätigkeitsspektrums erfolgt im nächsten Kapitel.

Die Untersuchung der Tätigkeitsbereiche zeigt, dass die Sphären zwischen Nutzer:innen und Entwickler:innen viele Überschneidungen haben. Entwickler:innen unterstützen auch Nutzer:innen in der Anwendung der FLOSS. Dies dient nicht nur der Verbreitung der FLOSS, sondern hat auch einen Einfluss für die Entwicklungsarbeit selbst: Als Softwaretester:innen geben die Nutzer:innen Feedback über die Verbesserungspotentiale der FLOSS und lassen sich möglicherweise für die Mitarbeit in der FLOSS-Gemeinschaft gewinnen. Dafür existieren in den meisten Projekten mehrere Kommunikationskanäle. Je größer die Entwicklungs- und Nutzer:innen-Gemeinschaft eines FLOSS-Projektes ist, desto mehr werden eigene Kommunikationskanäle für bestimmte Zielgruppen und Themen genutzt. Kleinere Projekte nutzen dagegen hauptsächlich den Funktionsumfang von FLOSS-Portalen und pflegen Formen der direkten Kommunikation wie E-Mail und Chat.

Die sozialen Interaktionen innerhalb von Entwicklungsgemeinschaften, insbesondere bei größeren FLOSS-Projekten, haben bei den Entwickler:innen eine nicht zu unterschätzende Bedeutung: Sie dienen nicht nur der Koordinierung der gemeinschaftlichen Unternehmung und der gegenseitigen Hilfe. Die soziale Dynamik in solchen Gemeinschaften kann in hohem Maße das Engagement von Entwickler:innen positiv so wie auch negativ beeinflussen. Werden soziale Konflikte bei der Entscheidungsfindung nicht gelöst, können sie auch zur Stagnation in der Projektentwicklung oder Abspaltung von Teilen der Entwicklungsgemeinschaft führen. Unklare Entscheidungsstrukturen, verdeckte Hierarchien und Mangel an Ressourcen in Form von Zeit, Fachwissen oder Testumgebungen wurden von den Entwickler:innen als hemmende Faktoren benannt.

Das Lernen erfolgt problemorientiert. Im Fokus der Tätigkeiten stehen die Korrektur von Programmfehlern oder die Implementation einer gewünschten Funktionalität der FLOSS. Schon im Vorgehen bei der Softwareentwicklung ist die Problemorientierung ein fester Bestandteil: Probleme einer bestimmten Softwareversion werden im Bug-Tracker gesammelt, ein neues Software-Release orientiert sich an zukünftig behobenen Programmfehlern beziehungsweise an einer neuen Funktionalität. In all diesen Bereichen wird problemorientiert vorgegangen. Dies zeigte sich auch bei der Beitragsanalyse in Kapitel 6.5.5 zur Problemlösung bei der Entwicklungsarbeit: Die Problemorientierung schlägt sich auch in den Kommunikationsinhalten zwischen den Entwickler:innen nieder. Diese Erkenntnis konnte anhand der durchgeführten Interviews bestätigt werden und zeigte zusätzlich, dass dies auch für andere Kommunikationskanäle gilt.

Wissen und Können wird sich durch die Entwickler:innen angeeignet, um ein bestimm-

tes Ziel zu erreichen. Demnach handelt es sich um intentionales Lernen. Dafür muss sich ein Grundverständnis über das Problem selbst und die notwendigen Werkzeuge zur Problemlösung erarbeitet werden. Es erfolgt eine bewusste Auswahl der Informationsquellen, die mit Blick auf ihre Eignung zur Problemlösung hin bewertet werden. Die Bewertung ist der betreffenden Person entweder aufgrund ihres Vorwissens selbst möglich oder es bedarf der Unterstützung anderer Entwickler:innen bei der Auswahl von Informationsquellen in Form von Hinweisen.

Zur Problemlösung werden drei unterschiedliche Strategien eingesetzt: Die *solitäre Strategie* steht oft am Anfang des Prozesses. Sie wird eingesetzt, um sich ein Grundverständnis zu verschaffen und ist fokussiert auf die solitäre Informationsrecherche in archivierten Forenbeiträgen, Dokumentationen und anderen verschriftlichten Informationsquellen. Dies kann auch der Quellcode von anderen FLOSS-Projekten sein. Führt die solitäre Strategie nicht zur erhofften Problemlösung, wird zur *selektiv sozialen Strategie* übergangen. Dabei werden gezielt einzelne kompetente Personen konsultiert oder nach einer Lösung für ein Problem in themenspezifischen Internetforen gefragt. Wird die Softwareentwicklung in einer Entwicklungsgemeinschaft praktiziert, kann die Unterstützung auch von anderen Entwickler:innen aus der Gemeinschaft kommen. In diesem Fall handelt es sich um die *kollaborative Strategie*. Sie birgt den Vorteil, dass die Entwickler:innen derselben Entwicklungsgemeinschaft auch umfassende Kenntnisse von der FLOSS selbst haben und dadurch zielgerichteter unterstützen können. Sie geben Feedback zum Programmcode anderer Entwickler:innen oder unterstützen beim Grundverständnis über Aufbau und Funktion der Software. Bei komplexen Problemen wird ein Mix der Strategien eingesetzt. Viele Entwickler:innen betonen den Anspruch, möglichst mit der solitären Strategie ein Problem zu lösen. Dies ist, neben pragmatischen Gesichtspunkten, auch auf handlungsethische Überzeugungen innerhalb der FLOSS-Bewegung zurückzuführen, die sich auch in den Nutzungsregeln[148] von Internetforen niederschlagen.

Neben dem internetbasierten Austausch leisten auch direkte Treffen zwischen Entwickler:innen einen wichtigen Beitrag zum Lernen. Je nach Projekt sind dies regelmäßige Treffen von Ortsgruppen, organisierte Arbeitstreffen oder Konferenzen. Als informelle Lerngemeinschaften dienen auch Hacker- und Makerspaces, da hier oft Personen miteinander interagieren, die eine Nähe zur FLOSS-Bewegung haben. FLOSS-Entwickler:innen nutzen die sozialen Kontakte in solchen Lernumgebungen bei der Problemlösung.

[148] Eine Anleitung zur Informationsrecherche liefern beispielsweise Raymond und Moen (2014) in ihrem Text 'How To Ask Questions The Smart Way' mit einem starken Fokus auf die solitäre Strategie.

9 Kompetenzaufbau durch die Entwicklungsarbeit

In diesem Kapitel wird die Forschungsfrage nach den Gegenstandsbereichen von Lernprozessen bei der FLOSS-Entwicklung beantwortet. Die Darstellung der Forschungsergebnisse erfolgt anhand von Kompetenzen, „[...] womit auf verschiedene Bereiche eines Bündels aus Fähigkeiten, Fertigkeiten und ihres reflektierten Einsatzes verwiesen wird“ (Nieke 2020a, S. 355). Zur besseren Übersicht erfolgt eine Einordnung der identifizierten Teilkompetenzen in die Kompetenzbereiche der Sach-, Sozial-, Selbst-, Sprach- und Leibkompetenz.

Zu beachten ist hierbei, dass die Darstellung der Kompetenzen rein deskriptiv erfolgt. Die Einschätzung des jeweiligen Kompetenzaufbaus erfolgt aufgrund der Selbstaussage der Entwickler:innen und in Einzelfällen durch meine Interpretation, die entsprechend begründet wird. Es erfolgt keine quantitative Einschätzung des Kompetenzaufbaus, im Sinne einer Performanzmessung, die Rückschlüsse auf das vorhandene Kompetenzniveau ermöglicht. Dies würde ein anderes Forschungsdesign erfordern.

Ziel dieser Beschreibung ist, das Spektrum der verschiedenen Kompetenzen aufzuzeigen, die durch das Engagement in FLOSS-Projekten aufgebaut werden können. Welche Kompetenzen dies sind, ist individuell verschieden und von diversen Faktoren abhängig. Aus diesem Grund wird das Spektrum des Kompetenzaufbaus vor dem Hintergrund der für den:die jeweilige Entwickler:in als bedeutsam erlebten Erfahrungen geschildert.

9.1 Sachkompetenz

Roth definiert die Sachkompetenz als Urteils- und Handlungsfähigkeit in Bezug auf Sachbereiche (vgl. Roth 1971, S. 180). Sachkompetenz ermöglicht die Freiheit, Aufgaben, Situationen und Dinge in der vorgegebenen Umwelt zu meistern. Der Aufbau von Sachkompetenz erfolgt durch erfahrungsbasiertes Lernen aufgrund der Auseinandersetzung mit Dingen der äußeren Natur. Aus vagen Hoffnungen und Erwartungen werden durch sachkompetentes Verhalten realisierbare Ziele. Ausschlaggebend für den Kompetenzaufbau ist dabei das Interesse an den Dingen unserer Umwelt, als Besonderheit von menschlichem Lernen.[149] Durch die Interaktion mit der Umwelt werden Sach-Erfahrungen gemacht, die durch Entdeckungs- und Handlungserfolge das Interesse wiederum verstärken und wodurch wir unsere Umwelt erschließen. Dem Kompetenzmodell von Roth liegt dabei ein normativer Anspruch zugrunde: Der Aufbau von Sachkompetenz erfolgt mit dem Ziel, zu sacheinsichtigem Handeln und intellektueller Mündigkeit zu befähigen (vgl. ebd., 456ff).

Im Gegensatz dazu zielt das Kompetenzmodell vom Arbeitskreis Deutscher Qualifikationsrahmen auf die Vergleichbarkeit von Qualifikationen innerhalb der Europäischen Union ab. Es ist nicht an Mündigkeit, sondern an Einordnung und Vergleich von Zugangs-

[149] Bei Tieren dient die Interaktion mit ihrer Umwelt hauptsächlich der Futtersuche und dem Nestbau. Bei der Beobachtung von spielenden Kindern kann gezeigt werden, dass potentiell alle Objekte in der Umwelt ihre Neugierde wecken und sinnlich erfahren werden (vgl. Roth 1971, S. 456f).

berechtigungen auf einem internationalisierten Arbeitsmarkt ausgerichtet. Das Konstrukt der Fachkompetenz in diesem Kompetenzmodell entspricht dabei der Sachkompetenz von Roth. Fachkompetenz setzt sich zusammen aus Wissen und Fertigkeiten und wird definiert als „[...] die Fähigkeit und Bereitschaft, Aufgaben- und Problemstellungen eigenständig, fachlich angemessen, methodengeleitet zu bearbeiten und das Ergebnis zu beurteilen“ (Arbeitskreis Deutscher Qualifikationsrahmen 2011, S. 8). Unter Fertigkeiten werden dabei kognitive und praktische Fertigkeiten zusammengefasst. Logisches, kreatives und intuitives Denken stellen die kognitiven Fertigkeiten dar, während sich die praktischen Fertigkeiten auf die Verwendung von Werkzeugen, Materialien und Methoden beziehen. Als Wissen werden dabei alle zu einem Sachbereich gehörenden Fakten, Theorien und Grundsätze verstanden. Ebenso wird die Kenntnis der Praxis in einem Lern- und Arbeitsbereich als Teil des Wissens der Fachkompetenz beschrieben (vgl. ebd., S. 8ff).

Beim Vergleich der beiden Kompetenzmodelle zeigt sich die normativ-anthropologische Ausrichtung bei Roth und der utilitaristische Ansatz bei der Definition der Fachkompetenz vom Arbeitskreis Deutscher Qualifikationsrahmen. Bei der Fachkompetenz liegt der Fokus auf den Anforderungen des Arbeitsmarktes und soll die *Employability* von Arbeitnehmer:innen EU-weit vergleichbar machen. Er ist stark auf die Anforderungen in beruflichen Kontexten zugeschnitten. Eine explizite gesellschaftskritische Reflexionsfähigkeit ist beim Qualifikationsrahmen nicht zu finden, da dieser auf die Sphäre formal erworbener Qualifikationen zugeschnitten ist.

Die in dieser Forschungsarbeit analysierten Fähigkeiten ermöglichen sowohl Rückschlüsse in Bezug auf eine gesellschaftskritische intellektuelle Mündigkeit, wie auch auf die Kenntnisse und Fertigkeiten, die für konkrete berufliche Aufgaben relevant sind.

9.1.1 Gesellschaftskritische Reflexion

Sachkompetenz mit Blick auf Urteilsfähigkeit bezüglich gesellschaftlicher Verhältnisse wurde in drei verschiedenen Kontexten gefunden: Sie betreffen die Kritik proprietärer Software, FLOSS als Möglichkeit zur Verbesserung von Partizipationsmöglichkeiten und die Perspektive des interkulturellen Austauschs innerhalb der Entwicklungsgemeinschaft.

Abhängigkeit durch proprietäre Software

Es wurde bereits festgestellt, dass viele der befragten Entwickler:innen das Interesse verfolgen, Alternativen zu proprietärer Software zu fördern. In vorliegender Forschungsarbeit wurde dies in Kapitel 7.3 als ein Auslöser für FLOSS-Engagement identifiziert. Die Vorteile von FLOSS sind jedoch nicht nur pragmatischer Natur. Der folgende Interviewausschnitt beschreibt die gesellschaftliche Bedeutung von FLOSS mit dem Fokus auf den Datenschutz und die Kontrolle über die Software, wie auch indirekt über die dafür erforderliche Hardware:

> „Ich würde es nicht an einem Punkt festmachen, aber über die Jahre kommt immer öfter der Punkt, dass man sich sagt, eigentlich müsste man viel mehr

> Open-Source haben in Verwaltung und allen möglichen anderen Sachen, einfach, damit man auch die Kontrolle über Programme hat. Und Kontrolle über Programme zu haben, die zukünftige Entwicklung, über Fehler, die im Programm drin sind, Einfallstore – dass man da halt immer ein Auge darauf haben muss und kann. [...] Sicherheit der eigenen Daten, des eigenen Rechners und ja, also wenn irgendwie Fehler im Programm drin sind oder Hintertüren. Und die Sicherheit, dass eine Software auch noch weiter funktioniert, wenn man sie noch weiter haben will, was bei Nicht-Open-Source-Software nicht zwingend gegeben ist.“ (Dokument I-8; Abs. 129 - 133)

Beim vorangegangenem Zitat wird die Bedeutung der Möglichkeiten von Freier Software herausgestellt. Die Interviewte Person hat ursprünglich aufgrund ihres pragmatischen Interesses damit begonnen, sich mit der FLOSS-Entwicklung zu beschäftigen. Im Verlauf der Zeit entwickelte sich daraus die Überzeugung, dass die Souveränität über Daten und Programme insbesondere für öffentliche Institutionen einen hohen Stellenwert hat. Durch die Nutzung von proprietärer Software wird eine Abhängigkeit zum Hersteller der Software aufgebaut. Nur dieser hat die notwendigen Kenntnisse zur Anpassung der Software bei sich ändernden Anforderungen. Das bedeutet, dass auch nur der Hersteller der Software darüber entscheidet, wie Fehler behoben, in welcher Art und Weise Daten verarbeitet werden und wie der Funktionsumfang der Software ausgestaltet wird.

FLOSS im gesellschaftlichen Kontext

Die Entwicklungsarbeit in FLOSS-Projekten erfolgt eingebettet in gesellschaftliche Rahmenbedingungen. Daraus ergeben sich auch Chancen für Nutzer:innen oder Kontaktmöglichkeiten mit Personen, die nicht direkt an der Nutzung der FLOSS interessiert sind, aber bei denen es dennoch Überschneidungen mit der Entwicklung und Nutzung von FLOSS gibt.

Anhand des folgenden Interviewausschnitts soll die Einbettung von FLOSS in den gesellschaftlichen Kontext veranschaulicht werden. Die zitierte Person ist nicht nur bei der Softwareentwicklung involviert, sondern auch beim Einsatz der vom FLOSS-Projekt angepassten Soft- und Hardware in Form von Sende- und Empfangsstationen im Stadtgebiet.

> „Das ganze [FLOSS-Projekt], nicht unbedingt jetzt [die] Softwareentwicklung, (.) [da] lernt man halt auch, dass die Welt nicht nur aus Software besteht, ’ne. Das man halt auch/ Also wenn man als Gesamtprojekt irgendwie so Erfolg/ Also wenn das Erfolg haben soll, da gehört halt mehr dazu als nur die Software zu entwickeln. Man muss auch die Leute von dieser Idee überzeugen. Oder wenn man auf Dächer will oder auf Rathäuser will, dann muss man halt im Rathaus vorsprechen, die Idee vorstellen. In einem Ausschuss oder einem Parlament das vorstellen oder gucken, dass da positiv für abgestimmt wird und zu sehen, dass man da dann Erfolg haben kann oder

> auch Misserfolg. Also es ist natürlich auch interessant oder im Rahmen dieser Flüchtlingskrise haben wir halt auch noch einmal gemerkt, wie wichtig kann halt Internet sein, wie wichtig ist halt Kommunikation für Menschen. Es motiviert einen dann natürlich auch wieder, weiter an diesem Projekt zu arbeiten.“ (Dokument I-2, Abs. 122)

Die Person schildert ihre Erfahrungen bei der Erschließung neuer Übertragungsstandorte für das Funknetzwerk. Dabei muss sie sich neue Praxen aneignen, da diese nicht den typischen Praktiken bei der Entwicklungsarbeit entsprechen. Dies bedarf der Auseinandersetzung mit den Bedürfnissen und Interessen der jeweiligen Entscheidungsträger:innen. Kontakte außerhalb der Entwicklungsgemeinschaft erfolgten auch zu Nutzer:innen. Auch das führt zu einer Auseinandersetzung mit deren spezifischen Bedürfnissen und Lebensbedingungen. Insbesondere der Kontakt zu Nutzer:innen marginalisierter Bevölkerungsgruppen ermöglicht eine Sensibilisierung für deren Lebensrealität in Bezug auf die vom FLOSS-Projekt angebotenen Dienstleistungen.[150]

Interkultureller Austausch in der Entwicklungsgemeinschaft

Die durch den Kontakt mit anderen Personen ermöglichte Reflexion kann aber auch schon im Kreise der Entwickler:innen vonstatten gehen, wie es folgender Entwickler beschreibt:

> „Insbesondere internationale Projekte sind natürlich immer schön, weil, weiß ich nicht, man fühlt sich halt in einem größeren Kontext relativ grenzenfrei. Also man kann halt interessante Gespräche, weiß ich nicht/ In dem [FLOSS-Projekt], in dem ich früher war, das waren halt Menschen aus Südafrika, das war halt eine interessante kulturelle Austauschphase. So ein bisschen, wenn man mal auch so über nicht-technische Sachen gesprochen hat. Wenn man so ein bisschen in der Debian-Welt ist, da sind halt auch viele Leute, weiß ich nicht, aus Indien oder aus fernen Teilen dieser Welt, von denen man kulturell eher nichts hört und kriegt man dann ab und zu so ein bisschen mit, (.) was die Menschen halt so schreiben, persönliche Einblicke in fremde Leben, ferne Welten, das ist ganz hübsch.“ (Dokument I-4, Abs. 208)

Die Bedeutung von Austausch in der Entwicklungsgemeinschaft über Themen abseits der Softwareentwicklung wurde schon in Kapitel 7.4.6 hervorgehoben. In diesem Beispiel erwähnt der Entwickler seine Neugierde auf Kontakte zu Personen anderer Kulturräume. Dadurch eröffnet sich für ihn die Möglichkeit, interkulturelle Perspektiven zumindest kommunikativ zu erfahren.

Diese Beispiele zeigen exemplarisch die Ergebnisse von Reflexionsprozessen. Sie dienen als Orientierungswissen dem Zurechtfinden des Individuums in seiner Lebenswelt.

[150] Der Entwickler beschrieb seine Erfahrungen mit Geflüchteten, deren Unterkünfte er mit kabellosen Internet-Zungangsmöglichkeiten ausstattete.

Marotzki und Jörissen grenzen wissenstheoretisch das Orientierungswissen vom Verfügungswissen ab. Die Herstellung von Orientierungswissen ist ein Bildungsprozess, der durch reflexive lebensweltliche Integration von Informationen die individuelle Selbst- und Welthaltung verändert. In Abgrenzung dazu ist die Herstellung von Verfügungswissen ein Lernprozess (vgl. Martozki & Jörissen 2008, S. 51f). Die Unterscheidung zwischen diesen beiden Wissensdomänen geht auf Jürgen Mittelstraß zurück. Er definiert:

> „Verfügungswissen ist ein positives Wissen, ein Wissen um Ursachen, Wirkungen und Mittel, Orientierungswissen ist ein regulatives Wissen, ein Wissen um Ziele und Maxime. Verfügungswissen konstituiert in wesentlichen Aspekten die moderne Welt, nämlich in Form von rationalen, technischen Kulturen.“ (Mittelstraß 1989, S. 19)

Im Grunde ist Verfügungswissen das Wissen darüber, *was wir tun können,* während Orientierungswissen Antworten auf die normative Frage liefert, *was wir tun sollen.* Diese Unterscheidung ist aber nur theoretischer Natur, da beide Wissensdomänen als Handlungswissen eng miteinander verzahnt sind (vgl. ebd., S. 20f). Dementsprechend handelt es sich bei den folgenden Ausführungen zur Sachkompetenz schwerpunktmäßig um Verfügungswissen.

9.1.2 Programmieren und Kontextwissen

Die naheliegende Vermutung konnte bestätigt werden, dass eine Verbesserung der Fähigkeiten im Umgang mit Programmiersprachen erfolgte, sofern dass Programmieren einen relevanten Anteil an der Entwicklungsarbeit ausmachte. Fast alle interviewten Personen berichteten von einer Kompetenzerweiterung in einer oder mehreren Programmiersprachen, die für die jeweiligen FLOSS-Projekte genutzt wurden. Programmiersprachen sind jedoch nur eines der Werkzeuge, um einen Programmfehler zu korrigieren oder eine gewünschte Funktionalität der Software zu implementieren. Dies kann nur geleistet werden, wenn es ein funktionales Verständnis der Umgebung gibt, in der die FLOSS eingesetzt wird.

Ohne dem notwendigen Hintergrundwissen zu einem Problem ist die isolierte Fähigkeit im Umgang mit einer Programmsprache für die Problemlösung nicht ausreichend.

> „Und da hatte ich am Anfang Probleme, dass aus irgendwelchen Gründen manche Knoten irgendwann aufgehört haben, Informationen zu verarbeiten. Und das war halt irgendwie spooky, weil es hätte an allem liegen können und ich habe halt (.) bestimmt wochenlang nach dem Fehler gesucht. Um dann am Ende festzustellen, dass man, wenn man vom Netzwerk-Socket abruft (.) dann auch so lange lesen sollte, wie man da Pakete bekommt. Und nicht irgendwie nach einem Paket aufhören sollte, was bei mir in der Entwicklungsumgebung halt total super funktioniert hat. Weil, wenn das alles auf potenter Hardware läuft, also auf dem eigenen Notebook und man halt zwei Knoten

> oder drei Knoten, die die ganze Zeit irgendwie immer fröhlich immer genau ein Paket hin- und herschicken und es da keinen Stau gibt, weil halt genügend Rechenleistung zur Verfügung ist, dann tritt dieses Problem halt nicht auf. Also in diesem Fall war es halt tatsächlich (.) ein Testumgebungsproblem, mit dem ich dann halt am Ende/ (.) ja.
>
> [Nachfrage Interviewer] Und hast du die Lösung selber gefunden?
>
> [Antwort] Die habe ich tatsächlich selber gefunden, in dem ich halt angefangen habe, mich durch die sämtlichen Dokumentationen von den ganzen Schnittstellen, die ich da benutze, durchzuwühlen. Um da am Ende zu sehen: Ah, okay, wenn ich da eine Schleife darum baue und das halt solange mache, bis dann ein entsprechendes Signal kommt, dass es nichts zu lesen gibt, dann funktioniert das auch. (Dokument I-7, Abs. 56 - 60)

Anhand dieses Beispiels wird deutlich, wie eine Lösung nur durch das Verständnis vom Kontext des Problems möglich wurde. Ausschlaggebend war ein Problem innerhalb der Kommunikation von einzelnen Netzwerkgeräten in einem Netzwerk. Da das Problem beim Test des Quellcodes nicht auftrat, war ein tieferes Verständnis über realistische Paketlaufzeiten und die entsprechende Reaktion der Netzwerkgeräte nicht notwendig. Erst im praktischen Einsatz der Software zeigte sich ein Problem, deren Ursache zunächst unklar war. Die Lösung erforderte ein Verständnis aller möglichen Einflussgrößen, die diese Fehlfunktion auslösen können. Dafür musste sich die Person das zum Verständnis notwendige Kontextwissen anhand der technischen Dokumentation der Netzwerkprotokolle und der Schnittstellen erst erschließen.

Der Lernprozess dieser Person wurde durch ein Problem ausgelöst: Aufgrund der Differenzerfahrung zwischen dem Anspruch an die Software und ihrer beobachteten Dysfunktionalität sowie dem Unvermögen, dies erklären zu können, wurde die Suche nach Kontextwissen favorisiert. Die Person nutzte dafür im Rahmen der solitären Lösungsstrategie die technische Dokumentation für die Informationsrecherche. In einem iterativen Prozess aus *Versuch und Irrtum* wurden potentielle Problemlösungen ausprobiert und deren Effektivität anschließend evaluiert, bis die Software die gewünschte Funktionalität reproduzierbar zeigte.

An diesem Beispiel ist demonstriert, dass alleinige Programmierkenntnisse für die Softwareentwicklung nicht ausreichend sind. Stattdessen ist eine Kombination mit dem Aufbau von Sachkompetenz in anderen Bereichen für die praktische Anwendung von Programmierkenntnissen notwendig. Diese sind je nach FLOSS-Projekt in verschiedenen Bereichen angesiedelt. Im Folgenden wird eine Auflistung der Teilbereiche wiedergegeben, in den die befragten Personen einen Aufbau von Sachkompetenz erwähnten:

- Administration und Aufbau von Netzwerkinfrastruktur
- Netzwerkprotokolle und Netzwerkanalyse

- Erstellung von Webseiten
- Paketierung von FLOSS für verschiedene Betriebssysteme
- Kenntnisse über Vor- und Nachteile diverser Softwarelizenzen
- Softwareergonomie beim Entwurf von Mensch-Maschine-Interaktion
- Umgang mit mehreren Betriebssystemen

Die hier aufgeführten Wissensgebiete sind nicht ausschließlich für die Softwareentwicklung relevant. Einige sind für den Test der FLOSS notwendig, der Präsentation der FLOSS für die Öffentlichkeit oder werden für die Verbreitung der FLOSS auf möglichst vielen Betriebssystemen oder verschiedenen GNU/Linux-Distributionen benötigt.

9.1.3 Gemeinschaftliche Softwareentwicklung

Der Übergang von individueller zur gemeinschaftlichen Softwareentwicklung erfolgt nicht nur rein quantitativ, dass mehrere Entwickler:innen arbeitsteilig Quellcodefragmente produzieren. Auch qualitativ ändern sich die Praktiken und Werkzeuge. Die gemeinschaftliche Unternehmung verlangt irgendeine Form der Koordinierung der individuellen Beiträge. Ebenso bedarf es eines Ortes, an dem die Beiträge gesammelt und an der vorbestimmten Stelle in die Codebasis integriert werden. Die beteiligten Entwickler:innen müssen mit Quellcode umgehen können, der nicht von ihnen selbst geschrieben wurde und dementsprechend auch ihren eigenen Programmierstil anpassen. Mit der Zahl der Projektmitglieder und einer sich zunehmend differenzierenden Codebasis steigen auch die Qualitätsansprüche an die FLOSS selbst. Dies führt dazu, dass die Praktiken und Werkzeuge einer wachsenden Entwicklungsgemeinschaft einem Anpassungsdruck ausgesetzt sind. Welche Sachkompetenzen beim Übergang zu gemeinschaftlicher Softwareentwicklung von den befragten Entwickler:innen als wichtig erachtet wurden, wird in den kommenden Abschnitten ausgeführt.

Umgang mit der Versionsverwaltung

Die essentielle Infrastruktur eines gemeinschaftsorientierten FLOSS-Projektes ist die Versionsverwaltung. Durch diese haben die Entwickler:innen Zugriff auf den Quellcode und andere zum Betrieb der FLOSS notwendigen Dateien. Dafür müssen sie sich zwangsläufig mit deren Funktionsweise auseinandersetzen. Dies ist die Voraussetzung, um überhaupt eigenen Quellcode in den Hauptentwicklungszweig der Software integrieren zu können. Viele FLOSS-Projekte nutzen die Angebote von FLOSS-Portalen. Diese bieten verschiedene Versionsverwaltungssysteme an und ermöglichen dadurch jederzeit den verteilten Zugriff auf die Codebasis. Die folgend zitierte Person betont den Stellenwert eines prominenten FLOSS-Portals für das eigene Engagement in mehreren FLOSS-Projekten:

> „[...] GitHub hat viele Sachen verändert, weil einfach alles mit GitHub ist inzwischen. Und durch GitHub ist es stark vereinfacht, irgendwo Patches beizusteuern und genau zu wissen, wo man irgendwo was machen muss. Also GitHub ist die einfachste Variante von den meisten Projekten, [um] irgendwo was beizusteuern. Bei GitHub geht es sehr sehr schnell. Wenn man alles so drinnen hat, hat man alles schön übersichtlich. Man kennt die Plattform von vielen anderen Projekten und kann dort schnell mitten drin sein. Was sonst sehr schwierig ist, man braucht extra Accounts für die Projekte selber und (.) ja.“ (Dokument I-8, Abs. 149)

Das FLOSS-Portal GitHub nutzt die Versionsverwaltung Git und erweitert diese um zusätzliche Kommunikationsmöglichkeiten zwischen den Nutzer:innen des Portals. Damit entspricht der Umgang mit der Versionsverwaltung einer Schlüsselkompetenz in diesem Kompetenzbereich, um sich überhaupt an anderen FLOSS-Projekten mit eigenen Beiträgen beteiligen zu können.

FLOSS-Portale erfüllen damit ähnliche Funktionen wie Lernmanagementsysteme, die in formalen Lernkontexten Anwendung finden. Informelle, virtuelle Lerngemeinschaften in Form von FLOSS-Projekten haben allerdings im Vergleich zu formalen Lerngemeinschaften andere Anforderungen an ihre CSCL-Umgebung. Der Hauptunterschied zwischen beiden Lerngemeinschaften liegt im Ziel des kooperativen und kollaborativen Lernens: Bei formalen Lerngemeinschaften, wie zum Beispiel universitäre Seminarteilnehmer:innen, steht die Verteilung von Lernmaterialien im Vordergrund, während bei FLOSS-Gemeinschaften die gemeinschaftliche Erstellung von Inhalten in Form von Quellcode den Schwerpunkt der kollektiven Praxis bilden. Als gemeinsame Schnittmenge können aber die diversen Kommunikationsmöglichkeiten zwischen den Nutzer:innen der Lerngemeinschaften eingestuft werden. FLOSS-Portale wie zum Beispiel GitHub, GitLab oder SourceForge sind dementsprechend spezielle Lernmanagementsysteme, die auf die Anforderungen von informellen, virtuellen Lerngemeinschaften angepasst sind. Das Ziel solcher Lerngemeinschaften ist die gemeinschaftliche Softwareentwicklung. Die Wahl der CSCL-Umgebung muss den Zielen und Arbeitsweisen der jeweiligen Lerngemeinschaft angepasst werden. Einen Überblick über die dafür notwendigen Strukturentscheidung liefern Kerres und Nattland (vgl. 2012, S. 256 - 259) bei der Beschreibung von notwendigen Infrastrukturen für E-Learning.

Umgang mit fremden Quellcode und fehlerhafter Dokumentation in unbekannten Projekten

Wollen Entwickler:innen sich zum ersten Mal in einem FLOSS-Projekte engagieren, müssen sie sich zuerst einen Überblick über dessen Struktur verschaffen. Auf dem FLOSS-Portal beziehungsweise auf der Webseite des FLOSS-Projektes sollten dafür idealerweise alle Informationen zu finden sein. Die Entwicklungsgemeinschaften von gemeinschaftsorientierten FLOSS-Projekten sind in der Regel daran interessiert, neue Entwick-

ler:innen für das Projekt zu gewinnen. Dementsprechend hat die Projektgemeinschaft ein Interesse daran, die Hürden für das erste Engagement einer neuen Person so gering wie möglich zu gestalten.

Auf die Frage, was die interviewte Person durch ihr Engagement in verschiedenen FLOSS-Projekten gelernt hat, antwortete diese:

> „Definitiv mit anderer Leute Code und Projekten umzugehen. Also es ist ja immer schwierig, wenn andere Leute etwas programmieren oder irgendwelche Projekte haben, überhaupt durchzusehen. Was passiert wo? Was macht der Code? Was passiert überhaupt? Wie baut man das Ganze? Was soll passieren? Und ja, da habe ich auf jeden Fall gelernt, mich in andere Projekte reinzuarbeiten, weil, wenn man in vierzig Projekten Patches reinhackt, muss man meistens erst einmal gucken, wo überhaupt muss der Patch überhaupt ansetzen. Das ist bei manchen Projekten einfach, bei manchen halt schon schwieriger. Aber ich habe da schon definitiv gelernt, mich bei allen möglichen Unwegsamkeiten auch durchzukämpfen und ja, mich reinzuarbeiten, dass man auch alle möglichen anderen Programme auch verstehen kann.“ (Dokument I-8, Abs. 137)

Bevor ein produktiver Beitrag zu einem FLOSS-Projekt geleistet werden kann, muss erst ein Verständnis von den inneren Strukturen und der Funktionsweise der Daten- und Programmstrukturen des Projekts aufgebaut werden. Neben dem eigentlichen Quelltext gibt es in der Regel eine entsprechende Dokumentation, die diesen Überblick ermöglicht. Auch die Art und Weise, wie Beiträge eingereicht werden können und welche Anforderungen sie erfüllen sollen, ist Teil der Dokumentation.

Unvollständige oder fehlerhafte Dokumentionen sind allerdings ein gravierendes Problem bei FLOSS-Projekten: „Incomplete or outdated documentation is a pervasive problem, observed by 93% of respondents, yet 60% of contributors say they rarely or never contribute to documentation.“ (Open Source Survey 2017). Dieses Problem in der gemeinschaftlichen FLOSS-Entwicklung ist nach dieser Studie vom FLOSS-Portal GitHub omnipräsent. Das folgende Zitat beschreibt die angewandten Strategien eines Entwicklers, um sich trotz fehlerhafter Dokumentation das Kontextwissen über die Softwarestruktur zu erarbeiten:

> „Und wenn das [die Dokumentation] nicht hilft, (.) der Blick in den Quellcode, um dann die fehlenden Dokumentation am Code abzulesen. [...] Also wenn man halt in so große Code-Basen reinguckt, die halt schlecht dokumentiert sind, dann kann man sich dann natürlich stundenlang festfressen und versuchen das Gesamtkonstrukt zu verstehen. Aber Projekte sind manchmal so riesig, oder machen so viel, dass man den kleinen Teil, den man selber nutzt/ (.) Ja das ist halt so die Suche nach der Nadel im Heuhaufen. Und da kommt dann halt meistens die Community oftmals in Form von Mailinglisten mit [unverständlich] zum Einsatz, wo man dann halt hingeht und/ Ja, ich

habe hier folgendes Problem und ich habe da und da schon geguckt aber ich sehe halt die Stecknadel nicht, die einem dann halt meistens entweder selber direkt die Lösung liefern können oder auf jeden Fall in die richtige Richtung schubsen, um dann halt zu sehen, dass man dann halt selber irgendwie in der Lage ist, das Problem dann auch (.) zu finden, wo es denn eigentlich hakt.

[Frage vom Interviewer] Und solche problemorientierte Kommunikation, findet die auf der Projektmailingliste statt?

[Antwort] Meistens ja. Oder man kennt halt durch Zufall einen Dev'er[151] von denen, weswegen ist Vernetzen in dem Umfeld für mich halt auch mittlerweile sehr wichtig, so dass man die Leute auch direkt ansprechen kann und sage kann: Hier, ich habe hier das Problem. Und entweder die Leute wissen das halt selber direkt oder die: Ach nee, das ist hier die und die Person, warte ich leite dich mal weiter beziehungsweise mache dich da mal bekannt." (Dokument I-7, Abs. 64 - 72)

Der Entwickler beschreibt hier zwei unterschiedliche Vorgehen, um den Informationsmangel durch eine fehlerhafte Dokumentation zu beheben. Er versucht zuerst die Fortsetzung der *solitären Problemlösungsstrategie* durch eine Ausweitung der Informationsquellen. Aus dem Quelltext leitet er die fehlenden Teile der Dokumentation ab, erstellt also gewissermaßen selbst die Dokumentation für die Fehlstellen. Dieser Vorgang kann bei Projekten mit überschaubarer Codebasis und nachvollziehbarem Programmierstil ausreichen, um das notwendige Verständnis zu ermöglichen. Scheitert der Versuch, geht er zur *kollaborativen Strategie* über: Er bittet andere Entwickler:innen der jeweiligen FLOSS-Gemeinschaft um Hilfe. Dafür nutzt er entweder die gruppenweiten Kommunikationskanäle oder wendet sich gezielt an als kompetent eingeschätzte Personen. In diesem Fall handelt es sich dabei beispielsweise um Personen einer FLOSS-Gemeinschaft, denen die Rolle von Kernentwickler:innen zuerkannt wurden. Die Wahl der Strategie ist unter anderem von zeitlichen Ressourcen abhängig. Die Bitte um Hilfe kann bei asynchronen Kommunikationskanälen längere Zeit in Anspruch nehmen als bei synchronen Kommunikationskanälen. Erreichen Fragesteller:innen auf einem synchronen Kanal sofort eine kompetente Person, wird der Problemlösungsprozess nicht unnötig verzögert. Dies erklärt die hohe Affinität von FLOSS-Entwickler:innen für Chat und andere Kommunikationskanäle mit Push-Nachrichten.

Nachvollziehbarer Programmier- und Dokumentationsstil

Wie es in dem vorherigen Punkt schon angedeutet wurde, ist die Lesbarkeit und Verständlichkeit vom Quellcode und der Dokumentation für die Entwicklungsarbeit von entscheidender Bedeutung.

„[...] mir ist es mittlerweile außerordentlich wichtig, dass die Dinge, die ich

[151] 'Dev' ist die Abkürzung des Wortes 'Developer'.

schreibe für andere weiter nutzbar sind. Also nicht nur legal[152] nutzbar sind, sondern dass, wenn ich Sourcecode schreibe, den auch so schreibe, dass auch andere Leute den leicht verstehen können und leicht weiterbenutzen können. Und das ist so ein bisschen/ Das passiert in den seltensten Fällen. Also wenn ich für jemand eine kleine Homepage baue, dann kommt kein Dritter und sagt: Ich will dieselbe Homepage nochmal haben. Es ist irgendwie Quatsch, aber der ganze Code-Stil ist mittlerweile einfach so, dass es immer im Hinterkopf bleibt. Und das prägt auch mein Empfinden von schönem Code, also wenn der Code so gebaut ist, dass er nicht modularisierbar ist oder das ein Dritter leicht sein Stück Software dazubauen kann, dann ist mit dem Code irgendwas falsch. Auch wenn er super toll funktioniert. [...] Also, wenn ich ein kommerzielles Produkt für jemanden mache, ist es komplett egal. Dann kann ich einfach den Kram runterschreiben. Der will, dass seine Homepage einmal funktioniert, dann funktioniert sie, also es wäre gar nicht notwendig. Sobald es natürlich ein Projekt ist, was wiederbenutzt werden soll, dann wird es notwendig, aber ich/ (.) Hm, ich empfinde das schon als Kompetenz, aber ich habe jetzt nicht den Vergleichsmaßstab. Muss ich mal gucken/ (.) Ich arbeite gerade halt auch gar nicht mit Leuten zusammen, die geschlossene Projekte machen. Das mache ich halt nicht mehr. (. . .) Ich kann es nur an mir selber beobachten, dass ich ein Bewusstsein dafür habe und das es mir ins Auge springt, wenn Code nicht auf diese Art und Weise geschrieben ist. Ich weiß nicht/ Genau,(.) ich habe nicht das Gefühl, dass es natürlicherweise mitkommt, wenn man Programmieren lernt. Insofern ist es irgendwie ein Aufbau [unverständlich] deswegen eine Kompetenz.“ (Dokument I-13; Abs. 214 - 218)

Ein Kriterium für die gemeinschaftliche Entwicklungsarbeit ist auch die Art und Weise, wie programmiert und dokumentiert wird. Der Quellcode sollte so geschrieben sein, dass andere ihn schnell verstehen können. Damit steigt die Wahrscheinlichkeit, dass Entwickler:innen Feedback für ihren Code bekommen und darüber Anerkennung und auch Kritik erfahren. Dementsprechend ist auch die Modularität von Software ein Kriterium: Ist der Quellcode als monolithischer Block geschrieben, muss dieser bei Änderungen auch komplett überarbeitet werden. Ist dessen Struktur aber in voneinander unabhängige Elemente gekapselt, die über eine Schnittstelle miteinander Daten austauschen, muss nur das betreffende Element geändert werden.

Ähnlich verhält es sich mit der Dokumentation: Sie erleichtert das Verständnis für den Quellcode. Ein guter Programmier- und Dokumentationsstil ermöglicht es Außenstehenden, an der bereits geleisteten Arbeit anzuknüpfen und selbst die Entwicklung fortzusetzen. Die Notwendigkeit, einfach nachvollziehbaren und dokumentierten Quellcode zu

[152] Hiermit meint die Person eine Freie Softwarelizenz, die den Nutzer:innen bestimmt Freiheitsrechten zugesteht.

schreiben, wird als wichtige Voraussetzung für Gemeinschaftsprojekte angesehen. Andernfalls kann die bestehende Implementation nicht weiterentwickelt werden und die Software veraltet.

Aus Sicht des informellen Lernens betrachtet, ermöglicht erst ein nachvollziehbarer Programmier- und Dokumentationsstil für andere Entwickler:innen die Nutzung der FLOSS als Informationsressource, wie es als gängige Praxis bereits im vorherigen Kapitel beschrieben wurde. Ebenso steigt die Wahrscheinlichkeit, Rückmeldungen über die Qualität des eigenen Quellcodes von anderen Entwickler:innen zu bekommen. Das Lesen und Verstehen vom Quellcode ist eine Form des selbstgesteuerten Lernens. Dohmen formuliert die Grenzen solches Lernens wie folgt:

> „Das selbstgesteuerte Lernen ist ein Mittelweg zwischen autonomer Selbstbestimmung auf der einen Seite und fremdbestimmtem Eingepaßtwerden in vorgegebene Lernarrangements auf der anderen Seite, der die Lernenden zu realistischen Möglichkeiten führen will, wie sie ihr Lernen jeweils unter Nutzung verschiedener, meist nicht von ihnen selbst organisierter Anlässe und Lernmöglichkeiten so weit wie möglich bewusst selbst steuern können.“ (Dohmen 2001, S. 41)

Das Lernarrangement in diesem Fall ist die Entwicklungsgemeinschaft als *Communities of Practice* mit ihren gemeinsamen Zielen, Austauschprozessen und dem Repertoire, zu dem auch der Quellcode und die Dokumentation gehören. Das Repertoire ist Teil des distribuierten Wissens, auf das die Entwickler:innen Einfluss nehmen können. Damit erfüllt eine FLOSS-Entwicklungsgemeinschaft die Kriterien einer sozialkonstruktivistischen Lernumgebung, wie sie von Gerstenmaier und Mandl (vgl. 2018, S. 226) aufgestellt wurden.

Projektmanagement

Wie sich FLOSS-Projekte untereinander koordinieren, hängt von den Bedürfnissen ihrer Mitglieder und von den vorherigen Praktiken ab. Das Projektmanagement lässt sich nicht klar einem Kompetenzbereich zuordnen, da die Kommunikation von Management-Strategien innerhalb der Entwicklungsgemeinschaft den Bereich der Sozialkompetenz betrifft. Aus diesem Grund werden in dieser Arbeit darunter Fähigkeiten verstanden, die mehreren Kompetenzbereichen zugeordnet sind. Eine grobe Zusammenfassung über die Besonderheiten einiger Aspekte des Projektmanagements bei FLOSS-Projekten liefert der folgende Interviewausschnitt:

> „Projektmanagement in einer Form, wo man sich jetzt nicht wie, wenn man in einer Firma Scrum[153] macht, wo man sich darauf verlassen kann. Okay,

[153] Scrum ist ein Vorgehensmodell bei der Projekt- und Produktentwicklung. Es entstammt ursprünglich der Softwaretechnik, kann aber auch in anderen Bereichen eingesetzt werden. Oft findet Scrum bei der kommerziellen Softwareentwicklung Anwendung und stellt einen Ansatz agiler Softwareentwicklung dar.

> Scrum, der Sprint[154] läuft 14 Tage, nach 14 Tagen ist es fertig, sondern das sind die Tasks, die zu machen sind. Wie verteilt man die am Besten, dass alle unabhängig voneinander halt an Dingen arbeiten können und wenn es mal da ist, ist es halt da. Das ist eine komplett andere Herangehensweise an Projektmanagement, als was man halt typischerweise aus Firmen her kennt.“ (Dokument I-6, Abs. 168)

Die Person hebt die unterschiedlichen Rahmenbedingungen des Projektmanagements in Firmen im Vergleich mit FLOSS-Projekten hervor: Formale Strategien des Produkt- und Projektmanagements in Unternehmen, wie zum Beispiel Scrum, basieren auf der Voraussetzung, dass die Entwickler:innen feste Rollen übernehmen und eine vertraglich zugesicherte Arbeitszeit für die Entwicklung zur Verfügung stellen. Bei der zum großen Teil ehrenamtlich betriebenen FLOSS-Entwicklung gibt es weder eine formale Sanktionsinstanz noch eine finanzielle Kompensation für die geleistete Entwicklungsarbeit. Dementsprechend sind auch die Entscheidungsprozesse und die Erledigung von Aufgaben nur bedingt planbar.

Dennoch findet in FLOSS-Projekten eine Koordination der gemeinschaftlichen Unternehmung statt: Einerseits als Softwaremanagement im Entwicklungsprozess und andererseits als Management der Entwicklungsgemeinschaft. Dazu bedarf es Sachkompetenz in verschiedenen Bereichen, um folgende Fragen des Projektmanagements beantworten zu können:

- Welche Fein- und Grobziele müssen für einen bestimmten Entwicklungsstand der FLOSS erreicht werden?
- In welcher Reihenfolge müssen Aufgaben erledigt werden?
- Über welche Kompetenzen verfügen die jeweiligen Mitglieder der Entwicklungsgemeinschaft?
- Was unterstützt Entwickler:innen bei ihrem Engagement im FLOSS-Projekt?
- Welche Interessen verfolgen die einzelnen Mitglieder und die Entwicklungsgemeinschaft als Ganzes?
- Welche Strategien und Werkzeuge sind für die Erreichung der Entwicklungsziele notwendig?

Viele der hier aufgeführten Aufgaben des Projektmanagements werden von den Projektmaintainenden wahrgenommen. Doch diese Rollenzuteilung ist nicht starr, wie bei der Untersuchung der E-Mail-Kommunikation beim qBittorrent-Projekt gezeigt werden konnte. Die Interviews belegten, dass in FLOSS-Projekten auch einzelne Entwickler:innen

[154] Ein Sprint ist ein Arbeitsabschnitt im Vorgehensmodell Scrum, bei dem ein Teilziel im Softwareentwicklungprozess geplant, umgesetzt und evaluiert wird.

einige dieser Aufgaben des Projektmanagements übernehmen und dadurch das Projektmanagement als verteilte Verantwortung realisiert wurde.

Im Gegensatz zum problemorientierten Lernen, welches beim Programmieren und Problemverständnis gefordert ist, handelt es sich bei diesem Kompetenzbereich um projektorientiertes Lernen. Diese Art des Lernens hat sich mittlerweile auch in institutionalisierten Lernarrangements in Form von entsprechender Didaktik und Methodik als projektorientierter Unterricht an Berufsschulen bewährt. Beide Lernprozesse werden oft in einem Atemzug genannt, weisen aber Unterschiede auf. Projektorientiertes Lernen ist stark auf das Ergebnis bezogen, vorhandenes Wissen wird angewendet und weiterentwickelt. Währenddessen muss sich beim problemorientierten Lernen neues Wissen angeeignet werden, um ein Problem zu lösen. Diese Unterscheidung macht nur bei einem engen Problembegriff Sinn, denn beide Lernprozesse können problembezogen sein (vgl. Reinmann 2016, S. 228ff).

Softwaretechnik

Die Softwaretechnik ist die „Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Softwaresystemen“ (Balzert 2009, S. 17). Sie wird untergliedert in die Bereiche Softwareentwicklung, Softwaremanagement und Softwarequalität (vgl. ebd., S. 19f). Da sich die FLOSS-Entwicklung in vielen Bereichen den Methoden der Softwaretechnik bedient, ist auch in diesem Bereich mit einem Aufbau von Sachkompetenz zu rechnen.

Der folgend zitierte Entwickler hat selbst theoretisches Vorwissen aus einem IT-Studiengang. Die tatsächliche Anwendung dieses Wissens erfolgte allerdings erst durch das Engagement in einzelnen FLOSS-Projekten und führte später auch zur Gründung eines eigenen FLOSS-Projektes. Der Entwickler beschreibt seinen Kompetenzaufbau durch das FLOSS-Engagement wie folgt:

> „Also rein programmiertechnisch alles, was irgendwie mit Projektaufbau, Umsetzung, Code-Quality, Unit-Testing, Automatic-Testing/ Alles Dinge, die mehr oder weniger on-the-fly immer wieder mal: Ja man könnte doch noch das, man könnte doch noch das, man könnte doch noch das/ Das habe ich alles dadurch gelernt, wie man dies tatsächlich in einem konkreten Projekt umsetzt. Das war sehr hilfreich. [...] man sieht Projekte und die verwenden die und die Library oder sie haben das und das so umgesetzt und dann lernt man Neues dazu.“ (Dokument I-11; Abs. 148)

Hier werden verschiedene Methoden und Anwendungsbereiche beschrieben, deren praktische Umsetzung durch die Aktivitäten in FLOSS-Projekten gelernt wurden. Diese betreffen alle drei Teilbereiche der Softwaretechnik. Auch die Rückmeldungen von Nutzer:innen zu dem eigenen FLOSS-Projekt beschrieb die Person als wichtigen Faktor beim

Lernen. In der Softwaretechnik entspräche dies den Funktionstests beziehungsweise Black-Box-Tests[155], deren Ergebnis ein Abnahmekriterium für die Fertigstellung von Software ist (vgl. Balzert 2009, S. 473). Die von Nutzer:innen eingereichten Bug-Reports sind oft ein Ergebnis solcher Black-Box-Tests. In mehreren Interviews gab es Hinweise, dass der Umgang mit den technischen und sozialen Aspekte von solchen Bug-Reports erfahrungsbasiert erlernt wurde.

Eine andere Person bewertet rückblickend das Transferpotenzial der formal gelernten Theorie über Softwaretechnik folgendermaßen:

> „Also wir haben natürlich auch an der Uni praktische Informatik gehabt, wo es ein bisschen um Java[156]-Entwicklung ging und ein bisschen Projekttheorie. Aber alles, wie man praktisch damit umgeht, vom Ticket-Management übers Source-Management, wie man mit anderen Leuten irgendwie zusammenarbeitet oder wie man darangeht, um überhaupt erst einmal ein Feature zu planen oder überhaupt irgendwas zu planen, habe ich eigentlich fast alles über die Open-Source-Projekte gelernt.“ (Dokument I-8; Abs. 141)

Auch in diesem Beispiel wird deutlich, dass der eigentliche Kompetenzaufbau durch die Anwendung in der Praxis erfolgte. Die Verbindung von eigenen Interessen mit dem Engagement in einem FLOSS-Projekt ermöglichten ein Lernen, was zwar zuvor im Rahmen formaler Lernprozesse beabsichtigt war, aber erst durch die Anwendung in einem realistischen Szenario tatsächlich erfolgte. Wie auch in der zuvor geschilderten Teilkompetenz des Projektmanagements handelt es sich auch hier um den fallbezogenen Transfer von bekannten Methoden. Insofern ist das Lernen der beiden zitierten Personen als projektorientiertes Lernen zu kategorisieren.

Doch auch ohne formale Vorbildung ist es für Laien durch partizipative Medienkulturen möglich, mittels selbstgesteuertem, autodidaktischem Lernen zu Expert:innen in einer gewählten Wissensdomäne zu werden. Medienkultur meint hier ein Zusammenspiel von unterschiedlichsten Medien und Akteur:innen - internetbasiert und nicht internetbasiert. Deren kommunikative Verflechtungen als Voraussetzung von Lernprozessen wird mit dem Konzept der *Kommunikativen Figuration* beschrieben (vgl. Wolf & Wudarski 2018). Dabei handelt es sich um ein Konzept, welches Ähnlichkeiten zu dem Forschungsdesign dieser Arbeit aufweist und ebenso von einer Pluralität von Informationsressourcen bei der Verfolgung selbstgesteckter Lern- und Handlungsziele ausgeht.

9.2 Sozialkompetenz

Um die Sozialkompetenz definitorisch zu fassen, soll wieder auf Roth zurückgegriffen werden. Soziale Kompetenz ist die Summe an Fähigkeiten und Fertigkeiten, die sozia-

[155] Bei einem Black-Box-Test wird die Reaktion der Software auf Testfälle hin geprüft, die eine hohe Fehlerwahrscheinlichkeit beinhalten. Die interne Programmstruktur wird bei solchen Tests nicht betrachtet (vgl. Balzert 2009, S. 473)

[156] Java ist eine objektorientierte Programmiersprache, die mit dem Ziel der Plattformunabhängigkeit entworfen wurde.

le Handlungsfähigkeit ermöglichen und sich in sozialer Intelligenz niederschlagen. Er definiert zwei Teilbereiche der Sozialkompetenz: „[...] Einsicht in gruppendynamische Prozesse, wie sie die Sozialpsychologie untersucht und beschreibt, aber auch in strukturelle Bedingungen des Zusammenlebens, wie sie die Soziologie analysiert“ (Roth 1971, S. 314). Roth grenzt die Sozialkompetenz vom sozialen Mitgefühl als affektive Komponente ab. Dies beschreibt die Fähigkeit, Rücksicht zu nehmen, die Perspektive anderer Personen antizipieren zu können und Solidaritätsbereitschaft mit Schwachen und Unterdrückten zu zeigen (vgl. ebd, S. 314). Beides ist notwendig, um sozialeinsichtiges und sozialkonstruktives Handeln zu ermöglichen. Ziel eines solchen Handelns liegt darin, dass:

> „[...] immer bessere, effektivere, gerechtere soziale Gleichgewichtsformen gedacht werden und als Handlungsanweisungen dienen, die zwischen Mensch und Mensch, Gruppen und Gruppen und dem einzelnen und der Gesellschaft immer freiere und gerechtere Formen der Vermittlung zu realisieren zu erlauben.“ (ebd., S. 477)

Dieser Anspruch an mündiges Handeln kann nicht allein durch Sozialkompetenz umgesetzt werden. Roth betont, dass jegliches auf Sachen und Güter orientierte Handeln in einem gesellschaftlichen Kontext steht und dementsprechend soziale Folgen impliziert. Aus diesem Grund erfordert mündiges Handeln sachkundiges und sozialeinsichtiges Handeln (vgl. ebd., S. 477).

Im Gegensatz zur Ausführung von Roth fehlt der Definition vom Deutschen Qualitätsrahmen die affektive Komponente. Die kritische Beurteilung sozialen Handelns auf der soziologischen Meso- und Makroebene ist ebenso nicht Teil der unter Sozialkompetenz subsumierten Fähigkeiten. Die wichtigen Komponenten dieses Kompetenzbereiches werden wie folgt beschrieben:

> „Sozialkompetenz bezeichnet die Fähigkeit und Bereitschaft, zielorientiert mit anderen zusammenzuarbeiten, ihre Interessen und sozialen Situationen zu erfassen, sich mit ihnen rational und verantwortungsbewusst auseinanderzusetzen und zu verständigen sowie die Arbeits- und Lebenswelt mitzugestalten.“ (Arbeitskreis Deutscher Qualitätsrahmen 2011, S. 9)

Im Vordergrund stehen das individuelle Arbeiten sowie Lernen in Gruppen. Die Niveaueinteilung beim Aufbau von Sozialkompetenz ist in acht Stufen untergliedert. Auf dem untersten Niveau beschreiben sie die Teilnahme an Gruppenprozessen mit besonderem Augenmerk auf die Kommunikation. Die Niveausteigerung verläuft in Richtung der Übernahme von besonderen Rollen innerhalb der Gruppe bezüglich Gruppenleitung und Umgang mit komplexen Aufgabenstellungen. Ebenfalls wird die Nähe zur Sachkompetenz betont. In der höchsten Stufe geht es auch gezielt um die Fähigkeit der Einflussnahme auf das Kompetenzniveau anderer Gruppenteilnehmer:innen (vgl. Arbeitskreis Deutscher Qualitätsrahmen 2011, S. 6f). Hier zeigt sich die strenge Ausrichtung auf die Anforderungen des Arbeitsmarkts, wie sie schon bereits bei der Definition der Fach- beziehungsweise der Sachkompetenz in Kapitel 9.1 aufgezeigt wurde.

Die Definition vom Qualitätsrahmen erinnert an die Formulierung von Lernzielkatalogen, während die von Roth in ihrer inhaltlichen und normativen Ausrichtung weitreichender formuliert ist. Ohne weitere Bestimmung der soziologischen und sozialpsychologischen Untersuchungskategorien ist die Roth'sche Definition von Sozialkompetenz nicht zu operationalisieren. Dennoch benennt Roth wichtige Einflussgrößen auf die Gruppendynamik, die vom Arbeitskreis Deutscher Qualitätsrahmen ausgespart bleiben: Affekte als notwendige Voraussetzung für Solidarität als normatives Ziel, zu welcher Sozialkompetenz befähigen soll.

Normative Ziele sind besonders in ehrenamtlichen Lernnetzwerken relevant, die nicht durch Lohnabhängigkeit und eine Betriebshierarchie strukturiert sind. Derartige Wertvorstellungen wurden bereits in Kapitel 7.2 vorgestellt. Solidarisches Handeln als normatives Handlungsziel ist in der Operationalisierung des Deutschen Qualitätsrahmens nicht explizit aufgeführt. Es kann aber Teil individueller Interessen sein oder sich aus der Reflexion über die Auswirkungen von Handlungen auf die Gesellschaft ergeben.

9.2.1 Kommunikation innerhalb der Community

Fast alle Entwickler:innen betonen die Bedeutung einer gelingenden Kommunikation innerhalb der Entwicklungsgemeinschaft. Was dies genau ist, ist individuell verschieden. Folgende Aufgaben haben die interviewten Personen dem gruppenweiten Austausch zugeschrieben:

- Absprachen über Arbeitsteilung
- Hilfestellung bei Fragen
- Entscheidungen über die Entwicklungsrichtung
- Förderung des Gemeinschaftsgefühls durch respektvollen Umgang
- Information über den Projektfortschritt

Dabei bringen die beteiligten Personen ihre persönlichen Sichtweisen, Erfahrungen und Interessen ein. Dementsprechend bleibt es nicht aus, dass es auch zu Missverständnissen und Konflikten kommt.

Konfliktlösungsstrategien

Angesprochen auf die bedeutenden Probleme bei der gemeinschaftlichen FLOSS-Entwicklung, sind es oftmals nicht die technischen Probleme, die in den Interviews benannt wurden. Soziale Konfliktsituationen können gravierende[157] Folgen für die Entwicklungsgemeinschaft als Ganzes, wie auch für das individuelle Engagement einzelner Entwickler:innen haben.

[157] Neben der Stagnation der FLOSS-Entwicklung durch die Einschränkung des Engagements von mehreren Entwickler:innen ist auch die Abwanderung von Teilen der Entwicklungsgemeinschaft (sogenanntes *Forking*) ein Problem, das die komplette gemeinschaftliche Entwicklung eines FLOSS-Projektes zum Stillstand bringen kann.

> „Sicherlich auch so ein paar zwischenmenschliche Kenntnisse, die man durch die Arbeit mit der Community irgendwie gelernt hat, wie man irgendwie dann (..) am Besten so untereinander umgeht oder halt auch/ Dass man halt auch wirklich irgendwie Grenzen ziehen muss, wenn halt Menschen wirklich anfangen zu beleidigen, dass man auch wirklich aktiv sagen muss, dass die halt irgendwie dann weg müssen. Naja, nicht weg müssen, das klingt jetzt doof. Aber halt, dass sie sozusagen/ Wenn halt Menschen schädlich sind für das Projekt, weil sie halt nur beleidigend sind oder halt schlecht Sachen diskutieren, dass man die auch aktiv ausschließen muss, zum Beispiel/ Damit man halt/ Weil sonst die Gefahr besteht, dass diese Menschen irgendwie andere Menschen abschrecken zum Beispiel. Also es ist halt so, ja schon so eine Art zwischenmenschliche Kenntnisse, quasi.“ (Dokument I-10; Abs. 259)

In Konfliktsituationen müssen sich die Beteiligten nicht nur mit der Perspektive der Konfliktpartei auseinandersetzen, sondern auch ihren eigenen Standpunkt kritisch hinterfragen. Die in diesem Zitat angesprochenen zwischenmenschlichen Kenntnisse beziehen sich also nicht nur auf andere Beteiligte, sondern referenzieren auch immer auf die eigene Person und die Situation, in der die Konfliktlösung verhandelt wird.

Fast jede:r befragte Entwickler:in in einem FLOSS-Projekt mit einer größeren Gemeinschaft kann von sozialen Konflikten berichten. Dies betrifft insbesondere Situationen, in denen Vertreter:innen einer Konfliktpartei sich nicht an den expliziten oder impliziten Zielen und Regeln der Gemeinschaft orientieren. Daraus entwickeln sich individuelle und kollektive Formen der Konfliktlösung, die Teil der Sozialkompetenz sind.

Das folgende Zitat beschreibt eine individuelle Konfliktlösung:

> „Hin und wieder gehen einem schon Sachen auf den Keks. Also ich/ Ich sage mal so, wenn es dann Reibereien sozialer Natur gibt, von wegen, dass man sich mit irgendwelchen Leuten schlecht versteht oder sich halt denkt: Ja, war das jetzt wirklich notwendig? Ich will da jetzt nicht so richtig mit nackten Finger zeigen, weil auch Leute im Team, die dann sich, statt irgendwas zu machen, sich dann halt auch teilweise im Bug-Tracker darüber auslassen: Mimimi, das ist doch so blöd. Und dann halt so: Aber ich verstehe das nicht, warum soll ich das machen? Und dann denke ich mir dann so: Junge, halt doch einfach die Fresse, das bringt so nicht wirklich weiter. So vor solchen sozialen Problemen ist da leider auch kein Projekt so richtig gefeit. Das nervt dann halt. Ich denke mir dann halt: Ja, muss man halt irgendwie darum herum arbeiten. Und ich denke mir dann halt: Ja, musst halt mit den Personen irgendwie klarkommen. Im Zweifel einfach ignorieren, weil solche Sachen, ja, (..) da kann ich jetzt auch gerade nicht mehr zu sagen, außer mich zu verheddern.“ (Dokument I-12; Abs. 42)

Häufig wird die Strategie des Ignorierens als individuelle Konfliktlösung angewandt, wenn der Konflikt nur einzelne Mitglieder der FLOSS-Gemeinschaft betrifft. Oftmals ist dies

das Ultima Ratio, wenn andere Formen der Kompromissfindung zuvor gescheitert sind. Diese Einsicht spiegelt die vielen Entscheidungsprozesse wieder, die im Rahmen der gemeinschaftlichen FLOSS-Entwicklung von den Entwickler:innen erlebt wurden. Teilweise führt die gemeinschaftliche Reflexion über vergangene Konfliktsituationen auch zur Einführungen gemeinsamer Regeln oder geänderten Praktiken, die sich im Repertoire der Gemeinschaft niederschlagen.

Kommunikation von Kritik

Der Kommunikationsabbruch als Form der Konfliktbewältigung ist jedoch nur das letzte Mittel. Dies gilt es im Regelfall zu vermeiden. Ein wichtige Fähigkeit dafür ist die Art und Weise, wie Kritik formuliert wird:

> „[...] aber ich würde auch sagen/ [Ich habe] auch ein bisschen gelernt, wie man halt mit Leuten irgendwie online umgehen muss oder kann oder wie man das tun sollte. Dass man sich halt nicht/ (.) Dass die Leute nicht instantan sich denken: Boah, was für ein Arschloch. Das will man am Ende auch nicht. Das ist (..) ich bin jetzt auch nicht perfekt darin, also will ich mich auch nicht so darstellen, aber/ Ich bin auch garantiert schwierig mit umzugehen, wenn ich Leuten ihre Pull-Requests auseinandernehme und sage: Das ist Kacke, das ist Kacke. Also halt netter formuliert, aber das mag ja auch keiner. Aber wenn man das irgendwie freundlich verpackt, dann geht es dann meistens schon eher. Und so etwas, würde ich sagen, zu lernen, das ist auch/ (..) Ich bin zumindest schon einmal auf dem Weg dahin, ein bisschen dies zu lernen.“ (Dokument I-12; Abs. 57)

In diesem Beispiel beschreibt die Person ihren Kompetenzaufbau in Bezug auf das Formulieren von konstruktiver Kritik. Ein Pull-Request entspricht einem eingereichten Projektbeitrag zur Behebung eines Fehlers im Quellcode. Dieser wird durch den:die Projektmaintainer:in evaluiert und entsprechend eingepflegt oder abgewiesen. Die Projektmaintainenden bewegen sich dabei in einem Spannungsfeld: Sie vertreten den Anspruch an eine stabil laufende Software und der Integration von neuen Funktionen auf der einen Seite und sind, auf der anderen Seite, dabei aber auf die Mitarbeit von anderen Entwickler:innen angewiesen. Aus diesem Grund ist es in ihrem Interesse, qualitativ hochwertige Beiträge zum Projektfortschritt und idealerweise eine kontinuierliche Mitarbeit anderer Entwickler:innen zu forcieren.

Eigenen Standpunkt vertreten

Gewachsene kommunikative Fähigkeiten zeigen sich aber auch an anderen Stellen, die auch auf ein gestiegenes Selbstvertrauen hinweisen:

> „Was ich besser kann, ist auf jeden Fall Kommunizieren. Ich habe auch schon bei Konferenzen Talks gehalten, das hätte ich vorher nie gemacht. [...] Ja, also

> sowohl technisch als auch sozial schon (.) Einiges. Also sozial eher so diese Offenheit, auch mal über irgendein Thema, was man/ Wo man irgendwie sich auskennt, auch zu reden." (Dokument I-9; Abs. 155)

Diese Aussagen zeigen, dass sich eine Änderung der kommunikativen Fähigkeiten nicht nur auf den Bereich der internetgestützten Kommunikation auswirkt. Dies ist dem Aspekt geschuldet, dass gemeinschaftliche FLOSS-Entwicklung oftmals nicht auf das Medium Internet begrenzt ist. Die Entwicklungsarbeit ist eingebunden in soziale Prozesse, die an vielen unterschiedlichen Orten stattfinden.

Aufbau und Erhalt der Gemeinschaft

Ähnlich verhält es sich mit anderen Kommunikationsprozessen innerhalb der Entwicklungsgemeinschaft. Wie es bereits im Kapitel 4.1.3 *Lebenszyklus eines FLOSS-Projektes* schon angedeutet wurde, ist FLOSS, wie auch jegliche andere proprietäre Software, auf eine kontinuierliche Aktualisierung angewiesen. Obwohl der Quellcode im eigentlichen Sinn nicht altern kann, unterliegt Software einem Alterungsprozess. Die Rahmenbedingungen von Software ändern sich fortwährend, weswegen auch die Software kontinuierlich aktualisiert werden muss. Aus diesem Grund sind FLOSS-Projekte auf das stetige Engagement von Entwickler:innen angewiesen. Das folgende Zitat belegt die Bedeutung der Kommunikation beim Aufbau und Erhalt einer aktiven Entwicklungsgemeinschaft:

> „Also, 'na, wenn man eben in einem nicht-Einzelprojekt ist, also quasi nicht als einziger Mensch innerhalb von einem Softwareentwicklungsprojekt ist, sondern auch ein bisschen versucht, andere Menschen reinzuziehen, dann versucht man halt auch kommunikativ so mit ihnen umzugehen, dass es eine einladende Erfahrung ist und das die Leute Lust darauf haben, sich mehr einzubringen. Dabei entwickelt man halt, weiß ich nicht, Kommunikationstechniken oder besseres Verständnis für den Menschen, um besser zu verstehen, worum es ihm gerade geht, um sie (.) ja, reinzuziehen." (Dokument I-4; Abs. 220)

Ziel der Kommunikation ist es, potentielle Entwickler:innen in die Entwicklungsarbeit einzubinden und die bestehenden Mitglieder für die weitere Mitarbeit zu interessieren. Die Person beschreibt treffend, dass dazu Kenntnisse der unterschiedlichen Interessen potentieller und aktiver Entwickler:innen notwendig sind. Dies erfordert einen kommunikativen Austausch über Dinge die nicht Hauptgegenstandsbereich der Kommunikationsinhalte bei der Entwicklungsarbeit sind, wie dies bei der Beitragsanalyse der Mailinglisten vom qBittorrent gezeigt werden konnte. Dennoch belegen die Interviews, dass ein Austausch über Interessen, auch abseits vom jeweiligen FLOSS-Projekt, stattfindet und von den Entwickler:innen als positive Schlüsselerlebnisse beschrieben werden, die die Bindung an die Projektgemeinschaft verstärken. Oft wird sich über solche Themen im direkten Austausch am Rande von Konferenzen, auf Arbeitstreffen oder im Chat ausgetauscht.

Diese Form der Kommunikation, die auf die Verbesserung der zwischenmenschlichen Beziehungen innerhalb der Gemeinschaft ausgerichtet ist, ist ebenso essentiell wie der Austausch über die eigentliche Entwicklungsarbeit. Downes beschreibt sie als nicht segregierbaren Bestandteil der Kommunikation in Lernnetzwerken (vgl. Downes 2010, S. 15f). Ebensolche Kommunikation ist ein Teil der Sozialkompetenz, die im Rahmen von gemeinschaftlichen FLOSS-Projekten aufgebaut werden kann und für den Aufbau und den Erhalt einer Gemeinschaft unabdingbar ist.

9.2.2 Eigene Rolle in der Gruppe finden

Unter der Teamfähigkeit werden viele unterschiedliche Anforderungen an die Gruppenmitglieder zusammengefasst. Einige dieser Anforderungen wurden bereits bei der Kommunikation dargestellt. Es fehlt noch der Bereich, der sich nur indirekt mit der Kommunikation beschäftigt: Das Finden der eigenen Rolle innerhalb einer Gruppe. Diese ist natürlich auch durch die Möglichkeiten geprägt, die sich aufgrund der anderen Kompetenzbereiche ergeben. Doch auch die Gruppendynamik und die Wahrnehmung der eigenen Fähigkeiten im Vergleich mit anderen Gruppenmitgliedern beeinflussen die Suche nach einer eigenen Rolle innerhalb der Gruppe.

> „Ja, man lernt natürlich auch einfach viel über Menschen bei so einem Projekt. In dieser ganzen Kommunikation, also jetzt auch nicht unbedingt/ Also man macht auch negative Erfahrungen, das ist nicht alles positiv, was man da erlebt und man kriegt auch mit wie/ (.) Also es ist nicht unbedingt die Softwareentwicklung 'ne, aber/ Also es gibt da auch immer wieder Leute, die halt in Machtpositionen sind oder so und dann manchmal Entscheidungen durchdrücken, die man selber nicht so trägt oder nicht versteht, warum sie die unbedingt durchdrücken wollen. Das ist halt auch ein altes Projekt, da gibt es so was öfter mal, dass dann manche alten Teilnehmer dann auch mal aufgrund von ihrer Stellung irgendwelche Entscheidungen durchdrücken oder so. Da macht man auch mal negative Erfahrungen, 'ne. Und die sind dann auch entscheidend für das folgende Engagement, 'ne, oder man hält sich dann vielleicht auch irgendwann aus gewissen Dingen raus oder engagiert sich dann noch viel mehr.“ (Dokument I-2; Abs. 88)

In diesem Zitat werden wichtige Einflussgrößen auf das eigene Handeln in Bezug auf die Gruppe geschildert. Diese sind der Einfluss von relevanten Anderen, die historisch gewachsenen Praktiken in der Gemeinschaft und der Abgleich zwischen eigenen Erwartungen und der tatsächlichen Gruppendynamik. In hierarchiearmen Gruppen haben relevante Andere aufgrund ihrer Reputation einen größeren Einfluss auf Gruppenentscheidungen (vgl. Taubert 2006, S. 163f). Neue Gruppenmitglieder müssen sich zunächst erst einmal einen Überblick darüber verschaffen, wer die anderen Personen in der Gruppe sind und welche Rollen und Kompetenzen sie in der Gruppe verkörpern.

Diese Strukturen sind historisch gewachsen. Sie bestimmen das geteilte Repertoire der Gruppe, welches durch frühere Konflikte, Entscheidungen und Entwicklungsetappen geprägt ist. Die Werkzeuge und Praktiken der Gruppe sind nicht immer explizit fixiert und deswegen für neue Gruppenmitglieder nur bedingt nachvollziehbar. Der letzte Punkt der Interviewpassage betrifft die individuell empfundene Diskrepanz zwischen eigenen Erwartungen und der vorgefundenen Realität in der Gruppendynamik. Dies ist von diversen Faktoren wie zum Beispiel eigenen Werten, Interessen, früheren Erfahrungen und der aktuellen Lebenssituation abhängig.

In diesem komplexen Gefüge müssen neue Mitglieder einer Gemeinschaft eine eigene Position einnehmen und sie innerhalb und außerhalb der Gemeinschaft vertreten. Diese Position ist nicht statisch und muss bei sich ändernden Bedingungen angepasst werden. Neue Mitglieder sind dabei aber nicht nur passiv und ordnen sich den vorgefundenen Strukturen unter. Sie können selbst durch gesteigerte Partizipation an Gruppenprozessen auf diese einwirken und zu relevanten Anderen werden:

> „[...] also in vielen Projekten merkt man es, glaube ich, dass die Projekte typischerweise enorm durch kleine Schlüsselerlebnisse, also sei es, weiß nicht, dass jemand da irgendwie halt vielleicht nur punktuell oder gerade zum richtigen Zeitpunkt intensives Engagement reinwirft und Sachen anschiebt, dass dann plötzlich die anderen Menschen mitmachen und sich mitreißen lassen und die Sache dann Fahrt aufnimmt. Also der/ Den Unterschied, den ein Einzelner machen kann, ist typischerweise halt irgendwie enorm in kleineren Projekten mit ein paar Leuten, die sich halt irgendwie alle in verschiedenen Aktivitätsphasen befinden. Da lässt sich dann durch Engagement viel bewegen und viel multiplizieren.“ (Dokument I-4; Abs. 172)

Dies wird von Entwickler:innen als Stärkung der individuellen Rolle erlebt, was sich wiederum in einer höheren Identifikation mit der Gruppe niederschlägt. Dieser Prozess wurde von vielen Entwickler:innen als sehr positiv erlebt. Im Sinne der Theorie der *Legitimen Peripheren Partizipation* von Lave und Wenger (1991) entspricht dies dem Wandel von peripherer Partizipation durch ein neues Gruppenmitglied hin zu voller Partizipation, zum Beispiel als Kernentwickler:in.

9.2.3 Handlungsmacht als Gruppe

Die zunehmende Identifikation mit der Gruppe wird von vielen Entwickler:innen bei der Reflexion über das vergangene Engagement in einem FLOSS-Projekt als positiv erlebter Prozess wahrgenommen:

> „Ja (.) also, ich denke mal, ein wichtiger Punkt war halt, als wir gesehen haben: So, wir wollen jetzt so ein Netz bauen, wir haben aber die Software nicht oder wir haben irgendeine Software, die irgendwer entwickelt. Da waren wir ja quasi neu, wo wir einfach gesehen haben: Okay, wir verstehen nicht, was

der da macht, es ist alles undurchsichtig. Es ist zwar cool, dass der das macht, aber so kann das im Grunde nicht weitergehen, wenn hier das Netz wachsen soll oder so. Das muss transparenter werden, stabiler, mehr Leuten die Möglichkeit geben, da teilzunehmen und es/ Ich meine, vielleicht schon schön zu sehen oder ein positives Erlebnis, dass wir dann, sagen wir mal drei, vier Leute gesagt haben: Okay, wir nehmen uns des Problems an. Und jetzt zu sehen: Okay, diese Software gibt es immer noch. Das hat funktioniert, die Software zu entwickeln. Die wird jetzt von 1000, 2000 Routern genutzt oder auf denen läuft sie. Das ist natürlich schon irgendwie schön, interessant, ja, motiviert einen natürlich auch, da weiter daran zu arbeiten. Also, wenn man weiß: Okay, das wird genutzt. (..) Genau, das einfach vielleicht dann auch zu sehen: Okay, zusammen können wir halt auch größere Projekte umsetzen. (.) Das ist natürlich positiv.“ (Dokument I-2; Abs. 88)

Der Entwickler schätzt die gemeinschaftlich erbrachte Gruppenleistung. Trotz frustrierender Erlebnisse bei verschiedenen Gruppenprozessen mit anderen Entwickler:innen blickt er mit einem gewissen Stolz auf die gemeinschaftliche Unternehmung. Erst durch die Kooperation und Kollaboration mit anderen Entwickler:innen wurde dieses FLOSS-Projekt ermöglicht. Die persönlich erlebte Erfahrung der gesteigerten Handlungsmacht als Gruppe im Vergleich zu der Handlungsmacht als einzelne Person wird von mehreren Entwickler:innen berichtet. Sie ist eine Form des sozialeinsichtigen Handelns als ein Aspekt der Sozialkompetenz, wie sie Roth definiert hat.

9.2.4 Grundelemente der Kommunikation in FLOSS-Projekten

Insgesamt lassen sich bei der Kommunikation innerhalb der Gemeinschaft drei zentrale Elemente identifizieren: Empathie für die Bedürfnisse anderer Entwickler:innen der Gemeinschaft, Umgang mit kommunikativer Aggression und das Bewusstsein über die eigene und fremde Machtausübung.

Empathie für die Bedürfnisse Anderer

Das Strukturmodell der *Communities of Practice* besteht aus drei notwendigen Bedingungen, die für die Gründung und den Fortbestand einer solchen Praxisgemeinschaft notwendig sind: Das ist eine gemeinsame Wissensdomäne, die Community in Form des sozialen Netzwerks und die wechselseitige Praxis innerhalb dieses Netzwerks (vgl. Wenger et al. 2002, S. 27). Die geschilderten Sozialkompetenzen sind für die Resilienz der Community unabdingbar. Dies gilt erstens in Bezug auf Konflikte, die ein inhärentes Merkmal von Kommunikationsprozessen sind, und zweitens für die Partizipation der einzelnen Personen an der gemeinsamen Praxis. Der Aufbau von Sozialkompetenz erfolgt mit zunehmender Partizipation an den Austauschprozessen.

„The *community* creates the social fabric of learning. A strong community fosters interactions and relationships based on mutual respect and trust. It

encourages a willingness to share ideas, expose one's ignorance, ask difficult questions, and listen carefully." (ebd., S. 28; kursiv im Original)

Wenger et al. betonen die Bedeutung von Respekt und Vertrauen in gegenseitigen Austauschprozessen. Mit Bezug auf den gegenseitigen Umgang in der Telekommunikation definiert die Netiquette als normativer Handlungskodex einen geschützten Raum, um lernwirksame Kommunikation zu ermöglichen. Da dieser Kodex mittlerweile auch außerhalb internetaffiner Milieus bekannt ist, ist davon auszugehen, dass dessen Inhalt auch den FLOSS-Entwickler:innen bekannt ist. Neben dieser explizit formulierten Handlungsnorm gibt es jedoch auch implizite Komponenten für eine lernförderliche Fehlertoleranz gegenüber Fragenden: Alle FLOSS-Entwickler:innen befanden sich selbst einmal in der Position, sich als neue Person mit einem Wissensdefizit an der Praxis einer bereits etablierten FLOSS-Gemeinschaft erstmalig zu beteiligen. Dies ermöglicht ihnen die Perspektivenübernahme aufgrund ihrer eigenen Lernbiographie. Weiterhin ist ihnen die Notwendigkeit einer kontinuierlichen Integration von neuen Entwickler:innen für die Erreichung der gemeinsamen Ziele bekannt. Ansonsten droht die FLOSS-Entwicklung zukünftig zu stagnieren, da Entwickler:innen wegen anderer Prioritätensetzung ihr Engagement einstellen.

Beide Faktoren ermöglichen die notwendige Empathie für ein stabiles soziales Netzwerk. Es ist, zusammen mit der gemeinsamen Wissensdomäne, die Grundlage für die kollektive Praxis. Diese kann auch als Resonanz im Sinne von Hartmut Rosa (2016) bezeichnet werden. Die neurologische Basis hierfür ist das Konzept der Spiegelneuronen, deren Resonanzerregung durch Hemmung und Blockierung soziokulturell überformt werden und die kognitive, affektive und leibliche Weltbeziehung prägen (vgl. Rosa 2016, S. 262ff).

Umgang mit Aggression

Um Aggressionen in FLOSS-Projekten zu verstehen, braucht es eine Betrachtung der Ursachen. Missverständnisse sind ein Teil von Kommunikation. Dies trifft insbesondere auf die Telekommunikation zu, da hier wichtige Signale der Körpersprache (Stimme, Mimik, Gestik) zur Interpretation der Kommunikationsinhalte fehlen oder eingeschränkt sind. Die Nutzung von Emoticons kann allenfalls als eingeschränktes Surrogat angesichts des breiten Spektrums der leiblichen Kommunikation gesehen werden. Einschränkend kommt hinzu, dass Kommunikation doppelt situiert ist: Einerseits bei der kommunizierenden Person und andererseits bei der interpretierenden Person. Als zusätzliche Fehlerquelle soll noch auf die Einschränkung aufgrund der Latenz von internetbasierter Kommunikation hingewiesen werden. Dies gilt in erster Linie für die asynchrone Kommunikation in Foren, E-Mails oder den Kommunikationskanälen der FLOSS-Portale (zum Beispiel dem Bug-Tracker) einschließlich der Versionsverwaltung[158]. Ursachen für die Verzögerung

[158] Die Kommentare der Versionsverwaltung dienen primär der Dokumentation und nicht dem Austausch von Meinungen. Dennoch reflektieren die Kommentare implementierte Entscheidungen und haben damit auch als Handlung einen kommunikativen Charakter.

liegen zum einen im technischen Design der Kommunikationskanäle und zum anderen in der möglichen territorialen Verteilung der Entwickler:innen über verschiedene Zeitzonen hinweg. Rücksprachen bei Verständnisschwierigkeiten können also nur verzögert erfolgen. Die hier aufgeführten Besonderheiten der Telekommunikation bieten demnach an unterschiedlichen Stellen Raum für Missverständnisse. Darüber hinaus existieren natürlich auch inhaltliche Zielkonflikte und unterschiedliche Auffassungen über die Strategien zur Zielerreichung. Auch wenn das gruppendefinierende Hauptziel in der Entwicklung einer bestimmten FLOSS liegt, können sich unter diesem Deckmantel ganz unterschiedliche Interessen verbergen, wie ich es bei den Auslösern und Beweggründen im sechsten Kapitel gezeigt habe.

Im Folgenden werden die Formen von Aggression in FLOSS-Projekten aufgezeigt. Die Datengrundlage hierfür sind die Interviews. Sie stellen also subjektiv eingeschätzte Kommunikationsakte aus Sicht der jeweiligen Person dar. Dieser Unterschied ist wichtig, da auf Grundlage der Situiertheit von Kommunikationsprozessen die Definition darüber, was als aggressiv eingeschätzt wird, subjektiv verschieden ist.

> „Es gab auch so vor zwei Wochen, glaube ich, jemand im IRC, der dann irgendwie/ Er hat sich über die schlechte Dokumentation aufgeregt, was auch ein valider Punkt ist, zugegeben. Er meinte auch irgendwie sofort, ohne dass ich irgendwie zu Wort kam: Fuck right off and be a prick elsewhere. Wo ich dann auch nur dachte: Was soll das?“ (Interview I-10, Abs. 215)

Die Interpretation als kommunikative Aggression ist bei diesem Kommentar aufgrund des beleidigenden Inhaltes noch recht klar. Die hier anschließend skizzierte folgende Situation dagegen fällt in einen Graubereich, der nur von der involvierten Person als aggressiv beziehungsweise übergriffig eingeschätzt wurde:

> „Na, es gab halt zum Beispiel früher das Problem, dass wir doch relativ lange gebraucht haben/ Sozusagen das Veröffentlichen von Kalendern zu implementieren. Sozusagen [dass du] den öffentlichen Link teilen kannst und den sich jeder anschauen kann. Und dann kamen dann halt wirklich irgendwie auf GitHub jeden Tag Menschen, die dann so: Hey, plus one. I need this feature implemented. Und das war dann wirklich irgendwann nervig.
>
> [Nachfrage vom Interviewer] ’Plus one’ heißt?
>
> [Antwort] Also, sozusagen Daumen hoch quasi. Genau, es ist denen wichtig. (.) Wo man sich dann auch irgendwann denkt: Hey, es ist Freie Software. Wenn du es brauchst, dann hilf doch mit, es zu implementieren. Aber dann ’spam halt nicht das Forum voll oder ’spam halt nicht GitHub voll.“ (Interview I-10, Abs. 207 - 211)

Der Befragte war für die Implementation und Dokumentation der gewünschten Funktion verantwortlich. Erst der wiederholt formulierte Wunsch von verschiedenen Nutzer:innen

führte zur gereizten Reaktion von ihm. An diesen Beispielen wird deutlich, wie groß die Spannweite von aggressiver Kommunikation sein kann und wie wichtig der situative Kontext ist. Dennoch war das Problem der Aggression in FLOSS-Projekten mit wenigen Ausnahmen kein Thema bei den Interviews. Die absolute Mehrheit der Kommunikationsakte wurde als konstruktiv und respektvoll empfunden.

Die Reaktionen der Betroffenen führten zu einer Neugestaltung der Beziehung zur aggressiv kommunizierenden Person. Laut Interviewaussagen wurde entweder die Kommunikation über bestimmte Themen anschließend verweigert oder es wurde sämtliche Kommunikation einschließlich anderer Formen der Zusammenarbeit eingestellt. Die individuelle Interaktionsverweigerung als Ultima Ratio steht jeder Person der Entwicklungsgemeinschaft zur Verfügung. Ein Ausschluss einer Person von der Nutzung der gruppenweiten Entwicklungsinfrastruktur erfordert im Regelfall die Intervention des:der Projektmaintainer:in aufgrund der Administrationshoheit über die Infrastruktur wie zum Beispiel die Versionsverwaltung oder die Mailingliste. Im Beispiel kam die Aggression in Form der Beleidigung von außen, weswegen eine Ausgrenzung dieser Person für die Entwicklungsgemeinschaft keine gravierende Störung darstellt. Hinweise auf Konflikte zwischen Entwickler:innen, die bis zu Aggressionen und dem anschließenden Ausschluss der Person eskalierten, wurden weder in der untersuchten Mailingliste noch in den Interviews gefunden. Hier greifen die Formen der Konfliktbewältigung, wie sie im vorhergehenden Kapitel aufgezeigt wurden: Konflikte und Missverständnisse innerhalb der FLOSS-Gemeinschaft haben ein hohes Bedrohungspotential für die Zusammenarbeit. In diesem Fall kommen abgestufte Formen der Konfliktlösung oder notfalls der Einschränkung der Kommunikation zur Anwendung. Viele Entwickler:innen betonen deshalb die Bedeutung eines synchronen Kommunikationskanals für die beschleunigte Klärung von Missverständnissen. Vor diesem Hintergrund muss auch die Bedeutung von Chat, Telefon oder direkten Treffen für das soziale Netzwerk der Entwicklungsgemeinschaft gesehen werden. Sie stärken die sozialen Beziehungen, indem sie Vertrauen stärken und die Menschen und ihre Lebenswelt hinter den anonymen Profilen erfahrbar machen.

Bewusstsein über eigene und fremde Machtausübung

Nach der klassischen Machttheorie bedeutet Macht „[...] jede Chance, innerhalb einer sozialen Beziehung den eigenen Willen auch gegen Widerstreben durchzusetzen, gleichviel worauf diese Chance beruht“ (Weber 1980, S. 28). Webers Machtbegriff beschreibt die Machtausübung über andere. Da das Engagement in FLOSS-Projekten bei den meisten Entwickler:innen auf Freiwilligkeit beruht, können diese sich jederzeit einem Machteinfluss entziehen.[159] Die Flucht aus einem Machtzugriff ist noch nicht einmal mit dem Verlust der geleisteten Arbeit verbunden. Da die FLOSS und damit auch

[159] Dies ist nicht für Entwickler:innen gegeben, die FLOSS-Entwicklung als Teil ihrer Lohnarbeit leisten. Sie unterliegen in diesem Fall einer Betriebshierarchie, da sie weisungsgebunden handeln und sich nur durch Kündigung der vertraglich und gesetzlich begrenzten Machtausübung durch Vorgesetzte entziehen können.

alle Entwicklungsbeiträge unter einer Freien Lizenz stehen, kann jede Person die Codebasis kopieren und die Entwicklung nach den eigenen Vorstellungen fortsetzen. Dennoch ist unbestreitbar, dass manche Entwickler:innen bei Entscheidungsprozessen mehr Macht als andere haben, um ihren Willen durchzusetzen. Aus diesem Grund hat sich mittlerweile eine Unterscheidung zwischen *power over* als Macht über andere und *power to* als Handlungsmacht durchgesetzt. Macht über andere setzt aber immer Handlungsmacht voraus (vgl. Schuck 2012, S. 44f). Handlungsmacht wiederum ist die „*Möglichkeit zum wirksamen Handeln*“ (ebd., S.46; kursiv im Original).

In FLOSS-Entwicklungsgemeinschaften gibt es ein Machtgefälle in Hinblick auf die Handlungsmacht. Relevante Andere haben eine größere Chance, ihre Vorstellungen bei der gemeinschaftlichen Entscheidungsfindung durchzusetzen. Ein mögliches Analyseraster ist die unterschiedliche Kapitalausstattung in Anlehnung an Bourdieu (1983). Für die Betrachtung von Einflussmöglichkeiten in Sozialen Bewegungen haben Nepstad und Bob (2006) das Konzept vom *leadership capital* entwickelt, dass sich aus kulturellem, sozialem und symbolischem Kapital ableitet.[160] Entwickler:innen, die reich an *leadership capital* sind, können mehr Einfluss auf die Entscheidungen in der FLOSS-Gemeinschaft nehmen, als andere. Unter den speziellen Bedingungen von FLOSS-Projekten sind folgende Kapitalarten für das Führungskapital ausschlaggebend (vgl. Nepstad & Bob 2006, S. 4f):

1. Kulturelles Kapital: Kenntnisse von Werten und kulturellen Prinzipien der FLOSS-Bewegung, rhetorische Fähigkeiten, strategisches Gespür für Chancen und Hindernisse bei der Projektentwicklung, angepasster Umgang in der Kommunikation mit den Entwickler:innen und ihren jeweiligen Interessen sowie Sachwissen

2. Soziales Kapital: Starke soziale Beziehungen zu anderen Entwickler:innen (insbesondere zu den Kernentwickler:innen) der Gemeinschaft, Beziehungen zu anderen FLOSS-Projekten oder anderen Netzwerken, die für das jeweilige FLOSS-Projekt von Bedeutung sein können

3. Symbolisches Kapital: Reputation, soziale Anerkennung, biographische Erfahrungen

In Bezug auf Führungskapital haben die Projektmaintainer:innen und Kernentwickler:innen in den meisten Projekten die größten Einflussmöglichkeiten. Dies zeigt sich auch durch die Kommunikationshäufigkeit und die Kommunikationsinhalte, wie sie im Kapitel 6.5 anhand der Beitragsanalyse der E-Mails vom qBittorrent-Projekt gezeigt wurde. Sie sind diejenigen, die am Häufigsten auf Kommunikationsakte reagieren und zugleich auch selbst Diskussionen auf der Mailingliste auslösen. Damit bestimmen sie maßgeblich, über

[160] Ökonomisches Kapital an sich wurde im Vergleich zu den anderen Kapitalarten bei Sozialen Bewegungen als weniger bedeutend eingestuft. Nur in seiner Möglichkeit zur Umwandlung in die anderen Kapitalarten erhält es eine Bedeutung (vgl. Nepstad & Bob 2006, S. 4).

welche Themen in der Entwicklungsgemeinschaft sich ausgetauscht wird. Zusätzlich haben Projektmaintainende die Handlungsmacht in Form der Entscheidung darüber, welche Projektbeiträge von anderen Entwickler:innen in die Codebasis aufgenommen und welche zurückgewiesen werden. Die Machtfülle ist aber nur relativ, da sie auf die Zuarbeit anderer Entwickler:innen angewiesen sind. Die Ausführungen des:der Maintainer:in vom qBittorrent-Projekt belegen jedoch, dass sie:er sich dieser Machtfülle durchaus bewusst ist. Nicht zuletzt deshalb, weil auch seine:ihre Laufbahn bei dem Projekt als Entwickler:in begann und sie:er erst später zum:zur Projektmaintainer:in avancierte.

Die Folgen von Missbrauch von Führungsmacht zeigte sich in den Interviews meist in der Einschränkung der individuellen Zusammenarbeit bis hin zur Einstellung des Engagements. In einigen Fällen führte dies aber auch zur Abwanderung mehrerer Entwickler:innen, die das Projekt dann unter anderem Namen nach eigenen Vorstellungen weiterentwickelten.

9.3 Selbstkompetenz

Bei der Definition der Selbstkompetenz nach Roth handelt es sich auf den ersten Blick um den Antagonisten zur Sozialkompetenz. Während die Sozialkompetenz dazu befähigen soll, das individuelle Handeln und Urteilen auf die Strukturen und Bedürfnisse von anderen auszurichten, zielt die Selbstkompetenz auf soziale Abgrenzung:

> „In der moralischen Entwicklungsförderung des Einzelnen geht es um den Aufbau einer persönlichen moralischen Einstellung, die sich in einem Verhalten äußert, das sich einem gegen die eigene Meinung und Erkenntnis gerichteten Druck der Gruppe nicht kampflos ausliefert [...] bzw. nicht ohne das Bewußtsein eines Konflikts der Gruppenmeinung anschließt.“ (Roth 1971, S. 551)

Für Roth steht bei der Selbstkompetenz das moralische Handeln im Mittelpunkt. Dies betrifft besonders ethische Konfliktlagen, in denen selbstbestimmte Handlungen oder Urteile gefordert sind, die unter Umständen gegen die eigene Interessen oder die der eigenen Gruppe gerichtet sind (vgl. ebd., S. 539). Im Kern ist die Selbstkompetenz also ein Korrektiv gegenüber dem Konformitätsdruck einer Gruppe oder der Gesellschaft, das sich als moralische Selbstbestimmung, Selbstbehauptung und Ich-Stärke manifestiert. Ziel ist die Befähigung des Individuums zur Selbstständigkeit und moralischer Selbsteinsicht in Bezug auf soziale Konfliktsituationen (vgl. ebd., S. 548). Dementsprechend handelt es sich beim Aufbau von Selbstkompetenz um all die Prozesse, die das Individuum als autonom agierende und urteilende Person gegenüber der Gruppe stärken.

Beim Vier-Säulen-Modell des Deutschen Qualitätsrahmens entspricht die Selbstkompetenz am ehesten der Kompetenzkategorie Selbstständigkeit. Sie bezeichnet „[...] die Fähigkeit und Bereitschaft, eigenständig und verantwortlich zu handeln, eigenes und das Handeln anderer zu reflektieren und die eigene Handlungsfähigkeit weiterzuentwickeln“

(Arbeitskreis Deutscher Qualitätsrahmen 2011, S. 9). Die Definition, wie auch die Operationalisierung in den Niveaustufen, fokussiert auf das selbstständige Arbeiten und Lernen. Unabhängig von anderen sollen Ziele definiert, notwendige Ressourcen akquiriert und die Folgen des eigenen Handelns reflektiert werden (vgl. ebd., S. 6f). Der Schwerpunkt liegt aber auf einer Selbstständigkeit im Sinne verinnerlichter Gruppenziele, die dadurch ohne äußere Anleitung auskommt und dennoch zu eigenen Handlungen in Konformität zur Gruppe befähigt. Bei Roth hingegen zielt die Selbstkompetenz auf die Entwicklung einer moralischen Urteils- und Handlungsfähigkeit, die ein kritisches Hinterfragen gesellschaftlichen Strukturen und Prozessen ermöglicht.

Nieke bringt in seiner Definition beide Pole zusammen. Sozialkompetenz hat dabei zwei Funktionen: Zum einen die Orientierung in der Welt und zum anderen die Selbststeuerung. Die Selbststeuerung soll zu einer selbstbestimmten Lebensgestaltung und Lebensführung befähigen. Teilweise wird sie auch als Selbstdisziplin oder Affektkontrolle beschrieben. In dieser Funktion hat sie mit der Sozialkompetenz eine gemeinsame Schnittmenge (vgl. Nieke 2012, S. 5f).

Der Aufbau von Selbstkompetenz konnte in den Interviews an verschiedenen Stellen identifiziert werden und erfolgt anhand der Kategorien Selbstvertrauen, Urteils- und Handlungsfähigkeit und Solidarität.

9.3.1 Selbstvertrauen in die eigenen Fähigkeiten

Viele Entwickler:innen verspüren im Verlauf der Auseinandersetzung mit FLOSS ein gestiegenes Selbstvertrauen in die eigenen Fähigkeiten. Dies beginnt schon früh, beispielsweise bei dem Prozess, auf das freie Betriebssystem GNU/Linux zu wechseln. Je mühsamer dieser Prozess erlebt wurde, desto stärker war danach das Vertrauen, auch andere technische Herausforderungen erfolgreich zu meistern:

> „Und man ist dann wirklich so nah dran, dass man den kompletten Rechner so für sich einrichten und arbeiten kann und verändern kann. Und ab dem Punkt habe ich auch dann das Gefühl gehabt, dass ich wirklich all das auch irgendwie so kontrollieren kann und wenn ich irgendwas anderes will, dann kann man es schon irgendwie machen und fummeln weil man so richtig nah dran ist. Und (.) ja, das war für mich auch so ein schönes Gefühl, wo ich sage: Ja, ich benutze Linux auch wirklich dauerhaft und nicht nur als zweites System oder zusammen mit Windows. Sondern ich habe dann seit vielen Jahren auch ausschließlich Linux benutzt und habe es für mich als System gefunden, wo ich, wenn ich ein Problem habe, kann ich es fixen und wenn ich was anders haben will, kann ich mir es selber reinbauen. Ja, es gibt ein sehr schönes Gefühl.“ (Dokument I-8; Abs. 125)

Von solchen Erfahrungen berichteten mehrere Entwickler:innen. Sie führten zu einem gestärkten Selbstvertrauen in die eigenen Kompetenzen. Dies zeigt sich beispielsweise

in der wachsenden Bedeutung von Freier Software für die eigene Nutzung oder das anhaltende Engagement in FLOSS-Projekten. Auch zeigten sich ähnliche Effekte bei Entwickler:innen in FLOSS-Projekten, die sich nicht direkt im Bereich der Programmierung engagierten:

> „Keine Ahnung, also, vielleicht bin ich ein bisschen selbstständiger geworden, einfach dass ich mich mehr traue, Sachen einfach selber auszuprobieren am Computer. Aber das ist ja jetzt irgendwie nicht so überraschend. Ja (..) und klar, technisch habe ich sicherlich so einiges gelernt.“ (Dokument I-3; Abs. 241)

Ebenso wurde die erste Übernahme der Rolle als Maintainer:in von vielen als wichtiges Schlüsselerlebnis angesehen, bei dem sich die jeweiligen Personen bewähren mussten. Es exponiert die betreffende Person gegenüber der Öffentlichkeit, da ihr dadurch die Koordinierung des Projekts zugeschrieben wird. Gegenüber der Projektgemeinschaft wird Verantwortung übernommen und die Verpflichtung eingegangen, sich nicht nur kurzfristig um das Projekt zu kümmern. Die Befragten gaben an, dass die erfolgreiche Bewältigung von anspruchsvollen Anforderungssituationen sich positiv auf das Selbstvertrauen auswirkt.

9.3.2 Transfer kritischer Urteils- und Handlungsfähigkeit

Einen Hinweis auf eine individuelle Handlungsfähigkeit, die sich aus einer Kritik gesellschaftlicher Rahmenbedingungen ableitet, liefert folgendes Zitat:

> „Na ja, was es auf jeden Fall gebracht hat, ist/ Ich habe gesehen/ Also, vielleicht fange ich so an: Man kommt in die Computerwelt rein und hat das Gefühl, man kommt nicht an Microsoft vorbei, heute vielleicht noch in Klammern: oder an Apple. Aber wenn man dann Energie reinsetzt und sich die Open-Source-Welt anguckt, merkt man: Oh doch, es geht sehr wohl, man muss es halt nur tun. Genau, und das ist eigentlich ganz spannend. Da habe ich das Gefühl, dass ist in vielen Bereichen so. Man hat das Gefühl, der vorgezeichnete Weg ist klar und es gibt überall klare Incentives dazu, schlimme Dinge zu machen, zum Beispiel [Microsoft] Word zu benutzen. Aber man muss halt nur ein bisschen Energie reinstecken und merkt schon, es geht auch ohne. Genau, das ist, glaube ich, in ganz vielen gesellschaftlichen Bereichen so, dass man das Gefühl hat, die Welt sagt dir es geht nur so, aber das stimmt überhaupt nicht, das geht anders. Ja, na das/ Jetzt haben wir einen interessanten Twist. Genau, das hat mir möglicherweise auch Vorschub gegeben, überhaupt politisch aktiv zu werden bei den Piraten. Zu sehen, doch, die Welt geht anders, wenn man nur genügend Anfangsenergie reinsteckt und dann ist die Welt auch ganz schön. Und man kann sich auch ganz anders daran beteiligen.“ (Dokument I-13; Abs. 198)

Die zitierte Person beschreibt einen gesellschaftlichen Konformitätsdruck, der die Nutzung von proprietärer Software zur Norm erhebt. Der de-facto-Standard äußert sich beispielsweise durch die breite Akzeptanz von kommerzieller, proprietärer Software und den Informationsaustausch auf Grundlage von proprietären Dateiformaten. Das Anreizsystem für die Nutzung solcher Software verläuft über den ausbleibenden Rechtfertigungsdruck und eine gut ausgebaute Versorgungsinfrastruktur für solche Software.[161] Nicht unterschätzt werden sollte auch der Punkt, dass sich bei der Nutzung weit verbreiteter Software auch leichter Unterstützung bei Problemen im eigenen sozialen Umfeld finden lässt.

Nutzer:innen von Freier Software, die sich an offenen Standards orientiert, profitieren nicht von diesen Vorteilen und werden dadurch diskriminiert. Um gleichermaßen an den gesellschaftlichen Austauschprozessen partizipieren zu können, müssen sie erheblich mehr an sozialem und kulturellem Kapital aufwenden. Dieser zusätzliche Aufwand schlägt sich aber nicht nur in einem Aufbau von Sachkompetenz nieder, sondern wirkt sich auch auf die Selbstkompetenz aus. Dem Konformitätsdruck zu widerstehen, also gemäß den eigenen Idealen zu handeln, ohne auf die Nutzung von Computersystemen zu verzichten, stärkt die individuelle empfundene Handlungsfähigkeit von vielen Entwickler:innen. Bei dieser Person führte es darüber hinaus zu einer Reflexion über anderen gesellschaftliche Bereiche, die in einem politischen Aktivismus mündeten. Dies zeigt einen Transfer von einem gesellschaftlichen Bereich auf einen anderen, der sich durch kritisches Urteilsvermögen und persönlich empfundener Handlungsfähigkeit auszeichnet.

9.3.3 Solidarität als moralische Handlungsnorm

Bereits im Abschnitt 7.2.3 dieser Arbeit wurde gezeigt, dass die reziproke Hilfsbereitschaft für viele FLOSS-Entwickler:innen einen Auslöser für ihr Engagement in einem FLOSS-Projekt darstellt. Dies impliziert gleichzeitig, dass die Hilfsbereitschaft bereits vor dem FLOSS-Engagement als Wertorientierung vorherrschte. Ob sich die moralische Selbsteinsicht durch das Engagement in FLOSS-Projekten grundlegend änderte, konnte anhand der Interviews nicht direkt belegt werden. Was gezeigt werden kann, ist die Verbindung von moralischen Urteilen und Engagement in FLOSS-Projekten:

> „Das Mitmachen in Freien Softwareprojekten gibt mir halt ein großes Gefühl von kooperativer Schaffenskraft und der, sagen wir, moralischer Fragwürdigkeit von Dingen, die man nicht der Gemeinschaft zur Verfügung stellt.“ (Dokument I-4; Abs. 208)

Der Entwickler bezieht sich auf die Solidarität dergestalt, dass die Veröffentlichung von Verbesserungen am Quellcode für ihn als moralische Handlungsnorm gilt. Dies schlägt sich auch in der Wahl der FLOSS-Projekte nieder die dieser Entwickler unterstützt. Die

[161] Viele Computersysteme werden bereits mit vorinstallierten proprietären Betriebssystemen verkauft, während beispielsweise vorinstallierte GNU/Linux-Systeme nur Nischenprodukte darstellen. Ebenso sind kommerzielle Dienstleistungsunternehmen bei Coumputerproblemen mit dem Betriebssytemen macOS (Apple) oder Windows (Microsoft) viel weiter verbreitet.

Bereitschaft, sich in einem Projekt zu engagieren, steigt für diesen Entwickler, je größer die Bedeutung der FLOSS für die jeweiligen Nutzer:innen ist und wie viele Personen diese Software nutzen. Noch präziser drückt es das folgende Zitat aus, indem es die Gründe für einen einzelnen Beitrag zum Projektfortschritt differenziert:

> „Und wenn man dann halt merkt: Okay, hier funktioniert was nicht, denn entweder es stört einen selber, so dass man davon betroffen ist, dass es nicht funktioniert. Dann ist man davon getrieben, im Sinne dessen, dass man da halt das Problem fixen will, daran mitzuwirken. Manchmal sind es aber auch so Sachen wie: Ach, oh ja, das ist ja ein Problem. Okay, das kann ich mal eben fixen. Dann tue ich das halt einfach mal für das Gemeinwohl. Und, naja, es ist auch so ein bisschen das schöne Gefühl: Okay, ich commite[162] jetzt auch wieder für die Gemeinschaft. (.) Ja es ist viel/ Ich glaube, wenn man da länger darüber nachdenkt, ist das oftmals auch so ein sozialer Aspekt dieses: Ich möchte eigentlich, dass Sachen für die Leute funktionieren, was einem in Teilen da auch immer so antreibt.“ (Dokument I-7; Abs. 32)

Das Engagement erfolgt zwar oft aufgrund eigener Betroffenheit, wodurch die Person an der Behebung des Fehlers ein direktes Interesse hat. Davon unabhängig kann es aber auch aufgrund der Bedeutung passieren, die es für andere Nutzer:innen hat. Dadurch ist das jeweilige Engagement uneigennützig und kann als solidarische Handlung bezeichnet werden.

9.4 Sprachkompetenz

In dem Kompetenzmodell von Roth ist die Sprachkompetenz nicht als eigener Kompetenzbereich beschrieben. Stattdessen wird sie als Teil des sacheinsichtigen Handelns gewertet. Sprache wird als eine Form des symbolischen Umgangs mit Dingen verstanden, die sowohl in der menschlichen Entwicklung wie auch in der Individualentwicklung die Voraussetzung für das Denken darstellt. Erst durch die Sprache wird eine neue Form der Sacherkenntnis ermöglicht, da sie im sprachlichen Austausch neue Einsichten über die Eigenschaften und Beziehungen zwischen den Dingen ermöglicht. Durch die Sprache können abstrakte Zusammenhänge gelernt werden, die sonst nur durch die direkte sinnlichen Erfahrung zugänglich waren (vgl. Roth 1971, S. 459f).

Nieke kritisiert zurecht an dem Kompetenzmodell von Roth, dass die Sprachkompetenz nicht nur ein eigenständiger Kompetenzbereich ist, sondern den Stellenwert einer Kompetenzdimension besitzt. Denn Sprachkompetenz ist die Voraussetzung für den Aufbau von Sach-, Sozial- und Selbstkompetenz (vgl. Nieke 2012, S. 4). Dementsprechend definiert er Sprachkompetenz als „[...] grammatisches und lexikalisches Vermögen zu Sprachverständnis und zum Umgang mit der Sprache.“ (ebd., S. 17).

[162] In der hier genutzten Form bezieht sich das Wort *commite* auf das englische *to commit*, also etwas übergeben oder einchecken. In verschiedenen Versionsverwaltungssystemen wird damit eine einzelne Änderung am Datensatz bezeichnet.

9.4.1 Fremdsprachenerwerb

Ein:e Projektmaintainer:in vom qBittorrent-Projekt berichtete von einer Verbesserung der Fremdsprachenkenntnisse. Die Person bezieht sich dabei auf Englisch, welche als Bezugssprache für den Austausch mit den anderen Entwickler:innen der Projektgemeinschaft dient. Über die genauen Teilbereiche der Sprachkompetenz, die sie durch ihr Engagement in FLOSS-Projekten vertiefte, liegen keine Informationen vor.

Im Rahmen der Interviews erwähnte ansonsten keine Person einen Kompetenzzuwachs bei der Sprachkompetenz. Dies kann damit zusammenhängen, dass dieser Bereich von den befragten Entwickler:innen, im Gegensatz zu der Sachkompetenz, als weniger relevant erachtet wurde. Dennoch muss davon ausgegangen werden, dass es durch inzidentelles Lernen in informellen Lernumgebungen zu einem Aufbau von Sprachkompetenz gekommen ist. Einen Überblick zum Forschungsstand zu inzidentellem Fremdspracherwerb durch internetbasierte Medien liefert Uhl (vgl. 2019, S. 30 - 35).

Aufgrund verschiedener Praktiken bei der Entwicklungsarbeit sind die Entwickler:innen gezwungen, englischsprachige Inhalte zu verstehen beziehungsweise selbst produzieren zu können. Die befragten Personen, die mutmaßlich allesamt Englisch nicht als Erstsprache gelernt haben, nutzten englischsprachige Medieninhalte bei der Informationsrecherche in technischen Dokumentationen, Tutorials und archivierten Forenbeiträgen. Das von fast allen befragten Personen konsultierte Online-Forum Stackoverflow ist fast ausschließlich[163] englischsprachig. Es ist anzunehmen, dass dies auch für die Mehrheit der technischen Dokumentationen gilt. Ebenso ist Englisch die Verkehrssprache der großen FLOSS-Portale wie GitHub, GitLab oder SourceForge. Dies hat zur Folge, dass auch die Fehlerberichte und die Kommunikation mit den Entwickler:innen auf Englisch erfolgen. Neben der Informationsrecherche und einfachen Koordinationsaufgaben, gibt es auch Aufgabengebiete mit erhöhten Anforderungen an die Sprachkompetenz der dafür zuständigen Entwickler:innen. Dies sind insbesondere die Lokalisierung[164] der FLOSS bezüglich der Dokumentation und der Mensch-Maschine-Schnittstelle. Die Forschungslage auf diesem Gebiet ist noch sehr dürftig. Hinweise für den inzidentellen Fremdsprachenerwerb bei der Nutzung digitaler Medien liefert beispielsweise Rogers (2016).

9.4.2 Schriftkommunikation zur gemeinschaftlichen Problemlösung

Telekommunikation zeichnet sich durch bestimmte Merkmale aus, die für das spezifische Themengebiet von Problemlösungssituationen relevante Vor- und Nachteile beinhalten. Der hauptsächliche Vorteil liegt in der Persistenz von diversen Kommunikationskanälen. Dadurch stellen vergangene Problemlösungssituationen in öffentlich zugänglichen Medien eine relevante Informationsquelle für Entwickler:innen bei ähnlichen Problemen

[163] Die absolute Mehrheit der Forenthemen sind in Englisch (ca. 20,6 Millionen) - daneben existieren *Stackoverflow.com*-Foren in Russisch (ca. 355.000 Themen), Portugiesisch (ca. 145.000 Themen) und Spanisch mit ca. 142.000 Themen (Angaben von Stackoverflow.com mit Stand vom 28.12.2020).

[164] In der Softwareentwicklung wird mit Lokalisierung die Anpassung von Software an sprachliche und kulturelle Gegebenheiten für spezifische Gruppen von Nutzer:innen bezeichnet.

dar. Dieses Vorgehen bei der Problemlösung wurde ausführlich in Kapitel 8.4 dargestellt. Dennoch hat Telekommunikation in Form der Schriftkommunikation auch gravierende Nachteile: Wichtige Informationen aus der Körpersprache wie Mimik, Gestik und Stimme fehlen. Diese Kommunikationsbestandteile stehen bei der Interpretation der Kommunikationsinhalte auf der Seite des:der Empfänger:in nicht zur Verfügung und können damit leichter zu Missverständnissen führen. Ebenso fehlt bei asynchronen Kommunikationskanälen ein zeitnaher Rückkanal. Dadurch können im Fall von Interpretationsschwierigkeiten Rückfragen zum Verständnis nur zeitversetzt erfolgen.

Dieser Nachteil kommt besonders in den Situationen erschwerend zum Tragen, wo es nicht um triviale Koordinationsaufgaben oder klare technische Fragen handelt. Die folgenden zwei Beispiele demonstrieren Situationen, bei denen eine besondere sprachliche Kompetenz gefordert ist:

> „Ja, es kommt halt immer auf die Größe und, was auch immer, auf das Projekt an, ob es irgendwie mehr ein Ein-Mensch-Projekt oder ein Viele-Menschen-Projekt ist und da sind Communities irgendwie, also abstrakt als Menschen, die da eine gewisse Emotionalität zu aufgebaut haben und sich ein bisschen Kümmern-Gefühle aufgebaut haben, das ist halt wichtig, um es langfristig zu erhalten. Und das ist sicherlich immer das Schwierige, da irgendwie da Leute reinzuziehen und da interessiert dran zu machen. Typischerweise halt über den technischen Weg wahrscheinlich. Manche Projekte machen, glaube ich, auch viel über den sozialen Weg, da habe ich aber kein Gefühl zu.“ (Dokument I-4; Abs. 148)

In diesem Fall handelt es sich um den sensiblen Akt der Gestaltung zwischenmenschlicher Beziehungen. Es geht zum einen um das fragende Deuten der Beweggründe potentieller Entwickler:innen und zum anderen um den Aufbau einer identitätsstiftenden, emotionalen Beziehung zu dem FLOSS-Projekt und der Entwicklungsgemeinschaft. Das zweite Beispiel betrifft nur indirekt die zwischenmenschliche Beziehung. Hier liegt der Fokus auf die Bearbeitung von Problemen, die in ihrer Komplexität gegenüber den sonstigen Problemen in der regulären Entwicklungstätigkeit hervortreten.

> „Die Idee ist halt nicht, dass einfach nur einer eine Entscheidung trifft und dann wird die umgesetzt so, sondern dass du halt schon, irgendwie konsensbasiert vielleicht, Entscheidungen treffen und ja, da ist es natürlich wichtig, dass man irgendwie zu diesem Konsens kommt oder zumindest mal eine Diskussion über die Themen anregt, führt. Ja und/ Also bei der Softwareentwicklung jetzt ist es halt so, dass/ Also wir haben halt so einen Issue-Tracker, da kann man dann, wenn man jetzt irgendwie eine Idee hat, kann man die dort vorschlagen. Dort erreicht sie dann erst einmal nur sozusagen die Entwickler ’ne, also die kriegen dann mit: Okay, da hat jemand was eingestellt, gibt es eine Idee, dann kann es da zu einer Diskussion kommen. Im Idealfall ist

> es halt so, dass es halt für alle überzeugende Argumente gibt, die dann dazu führen, dass man so die Richtung einschlägt. Kann aber auch mal sein, dass man dort zu keiner Lösung kommt oder einfach das Thema als so wichtig erachtet, weil das irgendwie eine grundlegende Änderung ist, dass man das mit einem größeren Kreis besprechen will. Da gibt es/ Skalieren wir es dann quasi an unsere Mailingliste, was so ein bisschen das Kommunikationsorgan für unsere Community ist.“ (Dokument I-2; Abs. 56)

In diesem Zitat wird das Vorgehen bei unterschiedlichen Komplexitätsstufen der Problemlösung geschildert. Die Projektgemeinschaft ist arbeitsteilig organisiert und widmet sich den Problemen innerhalb ihres Aufgabengebietes. Das Selbstverständnis der Gruppe sieht bei bestimmten Problemen jedoch eine Entscheidung in der Gesamtgruppe vor. Die betreffenden Entwickler:innen müssen bei der Problemlösung auch die Auswirkungen auf die impliziten oder expliziten Regelungen im Selbstverständnis der Gruppe mit berücksichtigen und dies innerhalb ihrer Teilgruppe validieren. Erst dann können sie entscheiden, ob die Gesamtgruppe zur Problemlösung hinzugezogen werden muss. Alle diese Kommunikationsakte geschehen unter den formulierten Einschränkungen der asynchronen Schriftkommunikation.

Die Anwendung der Schriftkommunikation als Teilkompetenz der Sprachkompetenz ist nicht zu verwechseln mit dem verschriftlichten Ausdruck der verbalen Anteile von sozialkompetenten Austauschprozessen von Angesicht zu Angesicht innerhalb der Entwicklungsgemeinschaft. Sie stellt eine eigene Teilkompetenz dar: Sozialkompetenz zusammen mit Sprachkompetenz befähigt zu gemeinschaftlicher Problemlösung unter den Bedingungen von raum- und zeitdistanter Kommunikation. Im Gegensatz zu verbalen Austauschprozessen unter Nutzung des gesamten Informationsspektrums der Körpersprache erfordert diese Form der Schriftkommunikation eine besondere Berücksichtigung der folgenden Bereiche:

- Präzise und unmissverständliche Kommunikation: Die schriftlichen Kommunikationsakte müssen in Bezug auf den Wortschatz auf die Rezipierenden angepasst sein. Formulierung und Satzbau müssen den Rezipierenden eine einfache und unmissverständliche Interpretation ermöglichen.

- Vollständige, strukturierte Argumentationskette: Das gewünschte Vorgehen bei der Problemlösung muss so begründet werden, dass sie in ihrer Logik nachvollziehbar ist.

- Referenz auf notwendiges Wissen, Standpunkte und Vorannahmen: Für die Rezipierenden muss der situierte Kontext des Kommunikationsaktes erkennbar sein.

- Antizipation von Interpretationsschwierigkeiten oder abweichender Position: Mögliche Missverständnisse und Gegenargumente bei den Rezipierenden sollten bereits beim Kommunikationsakt antizipiert und entsprechend thematisiert werden.

Diese Anforderungen an asynchrone Schriftkommunikation erfordert fundierte Kenntnisse der situierten Kontexte der Rezipierenden. Dazu gehören auch die Bedingungen der Praxisgemeinschaft mit ihren gemeinsamen Zielen, die Praxis des wechselseitigen Engagements und das gemeinsame Repertoire. Die Relevanz dieser Fähigkeit der gemeinschaftlichen Problemlösung mittels asynchroner Schriftkommunikation zeigt sich in den vielen Missverständnissen, die die Interviewten thematisierten. Insbesondere bei sozialen Konflikten innerhalb der Entwicklungsgemeinschaft ist diese Fähigkeit zur Konfliktlösung wichtig. Viele Entwickler:innen betonen deshalb die Bedeutung eines synchronen Kommunikationskanals zwischen den Entwickler:innen. Dieser Wunsch entspringt der Erfahrung von der Fehleranfälligkeit solcher raum- und zeitdistanten Kommunikation.

9.5 Leibkompetenz

Beim Drei-Säulen-Modell von Roth findet die Einflussgröße des Körpers als konstituierendes Element der Leiblichkeit keine Betrachtung. Möglicherweise ist dies auf einen von Roth gedachten Leib-Seele-Dualismus zurückzuführen. Die Phänomenologie von Merleau-Ponty versucht diese Dichotomie zu überwinden. In seiner Philosophie versucht er durch den Begriff des Leibes, so wie er ihn benutzt, den Gegensatz von Körper und Geist zu umgehen. Es gibt kein rein personelles autonomes Bewusstsein, sondern Sinnkonstituierung erfolgt anhand von sozial vorstrukturierten Erfahrungsfeldern, die erst durch die reflexive Erfahrung mit anderen ein gegenseitiges Verstehen ermöglichen. Dies zeigt sich beispielsweise in der Bedeutung, die das Körperschema auf unsere sozialen Interaktionen hat. Der Körper entspricht der physischen Seite unserer Existenz, während der Leib die Verbindung von Körper und unserem reflexiven Bewusstsein von uns selbst und unserer sozialen Umwelt ist, mit der er in einer wechselseitigen Beziehung steht (vgl. Meyer-Drawe 2001, S. 140 - 145). Konkret zeigt sich die Bedeutung des Leibes für soziale Austauschprozesse an folgendem Beispiel:

> „Die Offenheit unseres leiblichen Selbst ist durch das Faktum unserer Geburt gegeben. Unsere leibliche Verbundenheit mit dem Anderen zeigt sich darüberhinaus in unserer Verwundbarkeit durch den Anderen. Dabei ist nicht an extreme Formen körperlicher Gewalt gedacht, sondern etwa an den Blick des Anderen, der mir ein Gefühl des Ausgeliefertseins, der Gewalt eröffnen kann. [...] Der Blick des Anderen kann mich verlegen machen, kann mich belästigen, kann mich ermuntern und bestätigen, kann mir aber auch Angst machen und radikale Unsicherheit verursachen. Mein Blick kann die Aufmerksamkeit des Anderen auf mich nötigen, kann die inter-subjektive, aber anonyme Beziehung zu einem Kontakt verdichten; wir spüren den Blick des Anderen, ohne ihn zu sehen, usw.“ (ebd., S. 150)

Anhand dieses Beispiels wird deutlich, wie der gestaltbare Teil von Körperfunktionen Einfluss auf die Sozialität haben kann. Aus diesem Grund ist es folgerichtig, diese in ei-

nem eigenen Kompetenzbereich zusammenzufassen. Nieke erweitert deswegen die Trias von Roth zusätzlich um die Leibkompetenz:

> „Hierher gehören die bewusste Gestaltung von symbolischen Handlungen in Gestik und Mimik, die Stimmmodulation zur Hervorbringung der gesprochenen Sprache, der gestaltbare Anteil der Körperhaltung (daneben gibt es einen unbewussten, habitualisierten Anteil), die landläufig bekannte Körperbeherrschung, die in der 'Leibeserziehung' Thema einer leistungssteigernden Amplifikation ist. Dies wird zwar von Selbst- und Sozialkompetenz gesteuert und moduliert, nicht aber direkt umgesetzt.“ (Nieke 2012, S. 4)

Die von Nieke und Meyer-Drawe aufgeführten Körperfunktionen verdeutlichen bereits die Komplexität für eine Kompetenzmessung. Leibkompetenz erstreckt sich über ein breites Spektrum: vom Erlernen komplexer Bewegungsabläufe, wie zum Beispiel Fahrradfahren bis hin zu feinsten Handlungen in Form von Körpersprache, die großen Anteil an der Gestaltung zwischenmenschlicher Beziehungen haben.

Leibkompetenz bei der FLOSS-Entwicklung

Einen konkret identifizierbaren Aufbau von Leibkompetenz wurde weder im Rahmen der Beitragsanalyse der Mailinglisten noch anhand der Befragung gefunden. Dies kann verschiedene Ursachen haben:

- Zwischenmenschliche Austauschprozesse finden überwiegend internetgestützt statt. Aufgrund der technisch vermittelten Kommunikation, die überwiegend in Schriftform, wie zum Beispiel E-Mail, Chat oder schriftliche Kommentare in FLOSS-Portalen, stattfindet, werden nonverbale Kommunikationsinhalte gar nicht oder nur äußerst reduziert übertragen. Austauschprozesse durch physische Begegnungen der Entwickler:innen stellten für die Mehrheit der befragten Personen nur einen kleinen Teil der Kommunikation mit anderen Entwickler:innen dar.

- Der Reflexionsprozess der Entwickler:innen bei den Interviews fokussierte sich auf die Sach- und Sozialkompetenz. Dies kann damit zusammenhängen, dass zum einen die Explikation von Kompetenzen in diesen Bereichen von den befragten Personen einfacher zu leisten war oder zum anderen, dass die gewählten Leitfragen des Erhebungsinstrumentes nicht dazu imstande waren, einen solchen Reflexionsprozess zu initiieren.

9.6 Zusammenfassung

Die Befragung der Entwickler:innen lieferte einen tiefergehenden Einblick in den Kompetenzaufbau beim Engagement in FLOSS-Projekten. Die Untersuchung zielte auf eine Darstellung der Breite des Spektrums von unterschiedlichen Teilkompetenzen ab. Als Analyseraster diente das Kompetenzmodell aus Sach-, Sozial- und Selbstkompetenz von

Roth und deren Erweiterung durch Nieke mit der Sprach- und Leibkompetenz. Mit Ausnahme der Leibkompetenz konnten in allen anderen Kompetenzbereichen beziehungsweise Kompetenzdimensionen der Aufbau von Teilkompetenzen identifiziert werden. Den Schwerpunkt bildete dabei die Sachkompetenz und die Sozialkompetenz.

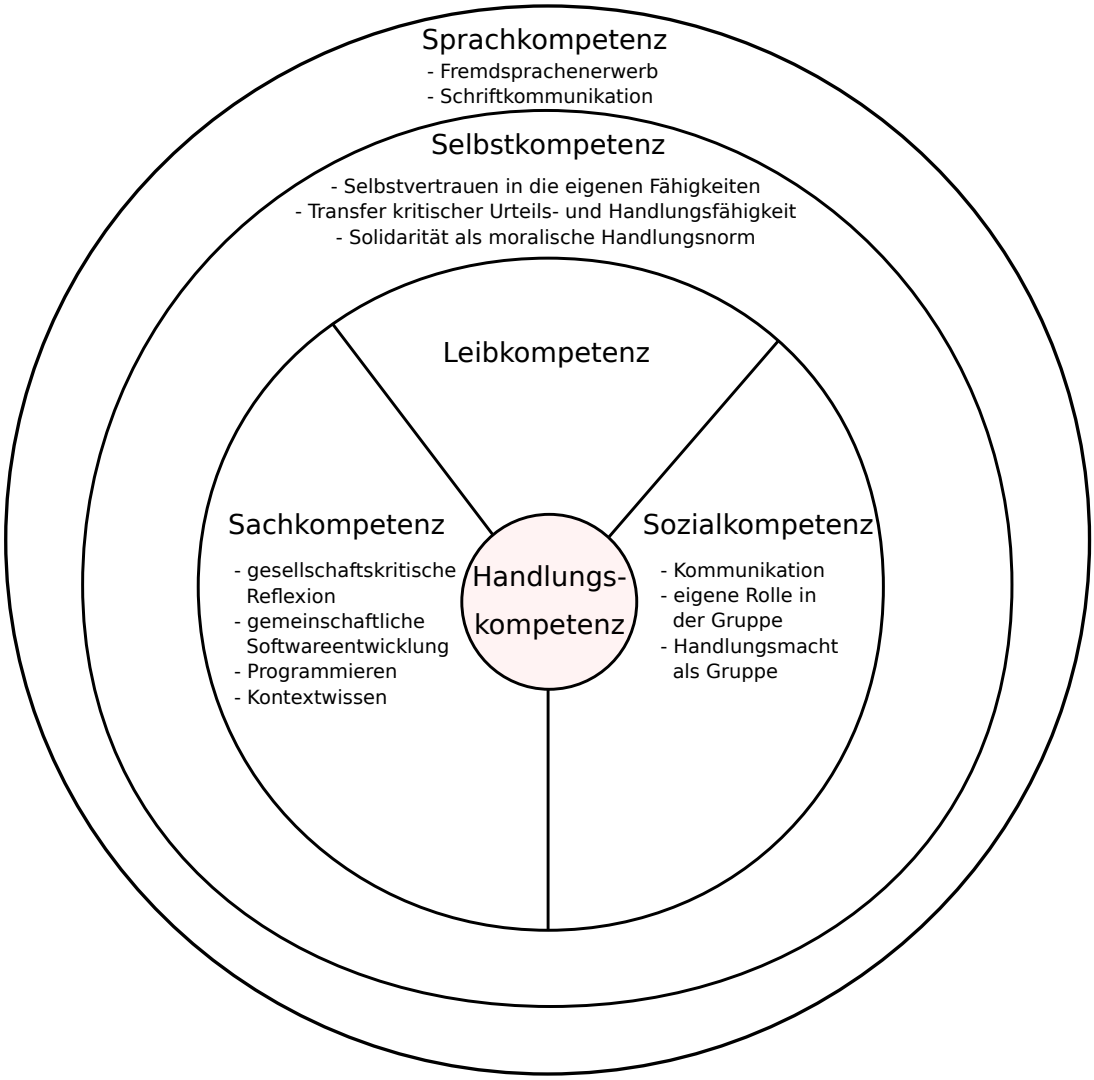

Abbildung 9: Teilkompetenzen bei der FLOSS-Entwicklung (Darstellung nach Nieke 2012, S. 5; Anpassung: K.M.)

Im Zentrum des Modells steht die Handlungskompetenz. Auf Grundlage aller anderen Kompetenzebereiche ermöglicht sie die Aufgabenbearbeitung und Werterealisierung (vgl. Nieke 2012, S. 5). Die äußerste Schale in diesem Kompetenzmodell ist die Sprachkompetenz. Bei den befragten FLOSS-Entwickler:innen sind dies in erster Linie Verständnis- und Ausdrucksfähigkeiten im Englischen, da alle Personen mutmaßlich Englisch nicht als Erstsprache erlernten:[165] Auf Englisch erfolgt oft die Kommunikation innerhalb der Entwicklungsgemeinschaft, in FLOSS-Portalen und in den wichtigsten Internetforen zur

[165] Alle Interviews erfolgten auf Deutsch und das Sprachniveau aller befragten Entwickler:innen war durchweg hoch und frei von einer Akzentfärbung durch eine Fremdsprache.

Softwareentwicklung. Auch sind viele technische Dokumentationen nur auf Englisch verfügbar.

Die nächste Schale ist die Selbstkompetenz. Sie dient der Orientierung in der Welt als moralische Urteils- und Handlungsfähigkeit. Gleichzeitig ist sie auch für die Selbststeuerung verantwortlich. Die befragten Entwickler:innen beschrieben einen Aufbau von Selbstkompetenz im Rahmen eines gestiegenen Selbstvertrauens in die eigene Fähigkeiten.

Auf Grundlage der Sprach- und Selbstkompetenz realisieren sich die Sach-, Sozial- und Leibkompetenz. Bezüglich der Leibkompetenz konnte kein Komptenzaufbau in dieser Arbeit festgestellt werden. Im Gegensatz dazu wurde von den Entwickler:innen ausführlich über Teilkompetenzen in den Bereichen der Sach- und Sozialkompetenz berichtet. Bei der Sachkompetenz wurden Teilkompetenzen in den folgenden Gebieten identifiziert: gesellschaftliche Auswirkungen von proprietärer und Freier Software, Programmierfähigkeiten in verschiedenen Programmiersprachen, Kontextwissen im Themenbereich der jeweiligen FLOSS-Projekte und Prozesse und Werkzeuge der gemeinschaftlichen Softwareentwicklung.

Bezüglich der Sozialkompetenz wurde ein Kompetenzaufbau bei der Kommunikation mit anderen Entwickler:innen identifiziert. Die Orientierung innerhalb der Gruppe und darin das Finden der eigenen Rolle in der Entwicklungsgemeinschaft wurde thematisiert. Mit zunehmender Erfahrung bei der gemeinschaftlichen FLOSS-Entwicklung berichteten die Entwickler:innen von einem positiven Bezug zur Handlungsmacht von Gruppen bei der Problemlösung. Auch dies ist ein Aspekt der Sozialkompetenz, den Roth als sozialeinsichtiges Handeln beschrieben hat.

Im nächsten Kapitel wird sich der Verteilung der Kompetenzen unter den Entwickler:innen gewidmet. Anhand einer Typologie werden individuelle Unterschiede beim Kompetenzaufbau herausgearbeitet.

10 Typologie der FLOSS-Gemeinschaft

Die Typenbildung in diesem Kapitel stellt die letzte Phase der Analyse dar. Ausgehend von den Interviews werden typische Merkmalskonstellationen bei Entwickler:innen beschrieben. Dadurch ist es möglich, von den Einzelfällen zu abstrahieren und den Fokus auf verschiedene Personengruppen innerhalb des Datenkorpus zu legen. Es ermöglicht die Untersuchung von Gemeinsamkeiten und Unterschiede zwischen den Einzelfällen selbst wie auch zwischen unterschiedlichen Formen des Engagements.

Dieses Kapitel gliedert sich in drei Teile: Im ersten Teil wird das Vorgehen der Typenbildung beschrieben. Daran schließt sich im zweiten Teil die Vorstellung der Typologie anhand der charakteristischen Merkmalskonstellationen an. Abschließend erfolgt im letzten Teil die Vorstellung der Typen anhand einer repräsentativen Fallinterpretation.

10.1 Typenbildung

Für die Typisierung der FLOSS-Entwicklungsgemeinschaft wurde sich für die fallorientierte Vorgehensweise entschieden. Im Gegensatz zur variablenorientierten Vorgehensweise geht es nicht um das Auffinden von kausalen Beziehungen zwischen den Variablen, sondern um die Beziehungen zwischen den Fällen. Das Ziel ist die empirische Klassifikation der Einzelfälle durch das Finden von Ähnlichkeitsstrukturen (vgl. Kuckartz 2010, S. 237).

Definition des Merkmalsraums

Für die Konstruktion der Typen wird auf die drei Kategoriekomplexe zurückgegriffen, die für die Beantwortung der Forschungsfragen zentral waren: die Beweggründe für ein Engagement bei FLOSS-Projekten, den Bedingungen für das Lernen bei der Entwicklungsarbeit und dem daraus resultierende Kompetenzaufbau. Zusätzlich wurden sekundäre Informationen über die FLOSS-Projekte selbst und den Umfang des individuellen Engagements für die Typenbildung mit einbezogen. Sie geben Aufschluss darüber, wie sich bestimmte Rollen und Strukturen in der Entwicklungsgemeinschaft auf das Lernen auswirken. Die Variablen wurden dichotomisiert.[166]

Konstruktion der Typologie

Zu Beginn der Typenbildung wurden Codematrizen für alle drei Kategoriekomplexe erstellt. Die Zeilen bestehen aus den Kategorien und die Spalten repräsentieren die Einzelfälle. Auf dieser Grundlage wurden die Ähnlichkeiten zwischen den Einzelfällen berechnet. Dieses Verfahren wurde ebenso auf eine Codematrix angewendet, die alle drei Kategoriekomplexe enthielt. Durch Reduktion wurden die Kategorien entfernt, die entweder nur bei einer oder zwei Personen oder bei fast allen befragten Entwickler:innen gefunden wurden.

[166] Dichotom bedeutet, dass eine Variable nur zwei Zustände einnehmen kann: Merkmal vorhanden oder Merkmal nicht vorhanden.

Einschränkung durch das Antwortverhalten

Das Verfahren war forschungspragmatisch sinnvoll. Es zeigte sich aber, dass interne Homogenität und externe Heterogenität nicht bei jedem Typus mit derselben Stärke erreicht werden konnte. Ursache hierfür sind die Unterschiede bei den Interviewantworten. Einige Entwickler:innen gaben sehr ausführliche und differenzierte Antworten zu allen drei Kategoriekomplexen. Dies führte zu einer Profilmatrix, in der viele Merkmale als vorhanden kodiert werden konnten. Demgegenüber gab es viele Interviews, die nur zu einem oder zwei Kategoriekomplexen differenzierte Antworten gaben. Eine auf diesen Daten beruhende Typenbildung hätte zu einer Prävalenz der Einzelfälle geführt, die bei allen Komplexen differenzierte Antworten gaben beziehungsweise eine hohe Anzahl an kodierten Merkmalen aufweisen. Bei der näheren Betrachtung der einzelnen Komplexe hätten diese Einzelfälle mit den Einzelfällen zu einem Typus gruppiert werden können, die auch in diesem Bereich viele Merkmale aufweisen. Dies hätte zu einer Typenbildung geführt, die keine externe Heterogenität aufweist, oder allenfalls zu einem Typus aus zwei bis drei Einzelfällen hoher interner Homogenität und den verbleibenden Einzelfällen, die aber untereinander in keinen weiteren Typus zusammengefasst werden konnten. Eine derartige Typenbildung hätte das differenzierte Antwortverhalten wiedergespiegelt, aber nicht bestimmte Merkmalskonfigurationen im Merkmalsraum.

Um diesen Effekt abzumildern, hätte es eine größere Fallzahl erfordert. Alternativ wäre auch eine Anpassung der Leitfragen möglich gewesen, die das Antwortverhalten der befragten Entwickler:innen auf bestimmte Kategorien festlegt. Ein solches Vorgehen hätte allerdings den narrativen Charakter des Interviews gestört und wäre auch nicht mehr mit dem Ziel einer theorieentdeckenden Forschung vereinbar gewesen. Stattdessen wäre ein solches Vorgehen für eine auf dieser Arbeit aufbauenden Forschung denkbar, die die Prüfung von Hypothesen zum Ziel hat.

Vorgehen bei der Typenbildung

Um eine Typenbildung trotz der Einschränkungen durch das Antwortverhalten vorzunehmen, wurde sich für die Einbeziehung von sekundären Informationen zu den Einzelfällen entschieden. Dies sind die Rahmenbedingungen der FLOSS-Projekte, in denen sich die jeweiligen Entwickler:innen engagieren und die jeweilige Bedeutung der einzelnen Projekte für die jeweils befragte Entwickler:in.

Zusätzlich wurde sich für eine intuitiven Gewichtung der Kodierung entschieden. Da die Befragten in der Regel in mehr als einem Projekt aktiv sind oder waren, wurde entsprechend mit Bezug auf das thematisierte FLOSS-Projekt die Antworten differenziert. Teilweise ergab sich daraus eine eineindeutige Zuordnung der Entwickler:innen zu den Typen. Je nach Auswahl des Projektengagements oder den Rahmenbedingungen konnte eine Person unterschiedlichen Typen zugeordnet werden.

10.2 Typologie der Entwickler:innen

Für die Identifikation von charakteristischen Merkmalsausprägungen wurde sich bei der Typenbildung auf die Zusammenstellung von möglichst gleichstarken Gruppen konzentriert. Die Typologie besteht aus vier Typen. Von den 14 befragten Entwickler:innen wurden jeweils vier Personen den *Maintainer:innen* und den *professionellen Programmier:innen* und jeweils drei Personen den *Gemeinschaftsorientierten* und den *sozialen Unterstützer:innen* zugeordnet. Die typspezifischen Merkmalsausprägungen werden im Folgenden vorgestellt.

10.2.1 Die Maintainer:innen

Dieser Typus beinhaltet Entwickler:innen, die in einem FLOSS-Projekt die Rolle des:der Maintainer:in ausfüllen. Sie engagieren sich in diesem FLOSS-Projekt überdurchschnittlich stark und zeigen mit dieser Rolle auch die Bereitschaft nach außen, dies auch zukünftig zu tun. Teilweise haben diese Entwickler:innen ein FLOSS-Projekt selbst initiiert, teilweise sind sie erst später dazugekommen und ihnen wurde aufgrund ihres Engagements diese Rolle angeboten. Vielfach sind sie von dem Wunsch getrieben, eine Freie Alternative zu bereits vorhandener, proprietärer Software zu entwickeln.

Mit dieser Rolle ist die Verwaltung der eingereichten Beiträge verbunden. Aus diesem Grund ist diese Gruppe gut vertraut im Umgang mit Versionsverwaltungssystemen. Dies ist in erster Linie *Git*, aber auch der Funktionsumfang der diversen FLOSS-Portale, die neben einer Versionsverwaltung noch zahlreiche andere Werkzeuge für die Kommunikation und Planung der Entwicklungsarbeit zur Verfügung stellen. Durch diese Rollen haben sich die Entwickler:innen Sachkompetenz mit Bezug auf die Methoden der Softwareentwicklung angeeignet. Doch auch die Sozialkompetenz ist wichtig, da sie auf die Unterstützung anderer Entwickler:innen angewiesen sind. Diesbezüglich wurde der Umgang mit Menschen, vor allem in Bezug auf die Einreichung von Bug-Reports und Projektbeiträgen, als Kompetenz beschrieben, die durch diese Rolle weiter ausgebaut wurde.

Die Maintainer:innen lernen viel durch die Praxis in einem FLOSS-Projekt. Dies ist aber nicht nur das Programmieren selbst, wie dies bei dem Typus der professionellen Programmierer:innen der Fall ist, sie betonen dabei auch die sozialen Prozesse, die für die Zusammenarbeit in einem FLOSS-Projekt notwendig sind. Allgemein nutzen sie mehr Kommunikationskanäle, als die anderen Typen. Dies gilt sowohl für die eigene Informationsrecherche als auch für den Austausch mit anderen Entwickler:innen.

Die Personen dieses Typus identifizieren sich stark mit ihrem FLOSS-Projekt. Sie beziehen viel Anerkennung aufgrund ihrer Arbeit als Maintainer:in, was sie in der weiteren Ausübung dieser Rolle bestärkt. Dies ist aber nicht nur die Anerkennung ihrer Arbeit als Maintainer:in im Rahmen der Entwicklungsgemeinschaft. Schon die Zahl der Nutzer:innen ihrer FLOSS vermittelt ihnen das Gefühl, dass ihr Engagement sinnstiftend und gesellschaftlich nützlich ist.

10.2.2 Die professionellen Programmierer:innen

Bei dem Typus der professionellen Programmierer:innen handelt es sich um Personen, welche die FLOSS-Entwicklung nicht nur in ihrer Freizeit betreiben. Sie werden direkt oder indirekt für ihre Arbeit an einem FLOSS-Projekt bezahlt. Entweder sind sie Mitarbeiter:in einer Firma, die diese FLOSS selbst einsetzt oder die Entwickler:innen installieren die FLOSS im Kundenauftrag beziehungsweise passen diese an. Obwohl deren Engagement einen wichtigen Teil ihrer Erwerbsarbeit darstellt, investieren diese Personen auch einen großen Teil ihrer Freizeit für die FLOSS-Projekte. Dabei wird sich oft in verschiedenen Projekten gleichzeitig engagiert und die Grenze zwischen FLOSS-Engagement als Lohnarbeit und als Freizeitbetätigung wird individuell verschieden gezogen. Manche Entwickler:innen halten diese Sphären strikt auseinander, für andere ist der Übergang fließend. Viele gaben explizit an, dass ihnen die FLOSS-Entwicklung Freude bereitet.

Verbunden mit der Lohnarbeit ist auch die Bedeutung der Reputation. Ein FLOSS-Engagement repräsentiert auch nach Außen die Sachkompetenz der Entwickler:innen. Die Entwickler:innen sind sich dieser Außenwirkung bewusst und sie benennen diese auch als einen von mehreren Beweggründen für ihr Engagement.

Bei den professionellen Programmierer:innen gibt es einen hohen Anspruch an den Programmier- und Dokumentationsstil. Hier liegt der Stellenwert darauf, dass auch der eigene Quellcode dazu geeignet ist, dass andere diesen leicht verstehen und daran anknüpfen können. Dies liegt darin begründet, dass diese Entwickler:innen selbst fremden Quellcode für die Lösung ihrer Probleme nutzen beziehungsweise dadurch ihre Programmierfähigkeiten ausbauen. Ebenso ist für sie die Rückmeldung von anderen Programmierer:innen zu ihrem eigenen Quellcode eine wichtige Ressource, um die eigenen Programmierfähigkeiten zu verbessern.

Alle Personen dieses Typs zeichnen sich durch eine hohe Sachkompetenz auf dem Gebiet der Softwareentwicklung aus. Das Programmieren stellt oft den Schwerpunkt ihres FLOSS-Engagements dar. Sie verfügen über viel Spezialwissen in Bezug auf den Aufbau und die Funktionsweise der Software und widmen sich dadurch auch den Problemen, die von den meisten anderen FLOSS-Mitgliedern nicht bearbeitet werden können. Oft gehören sie zu den Kernentwickler:innen in unmittelbarer Nähe zu den Maintainer:innen.

10.2.3 Die Gemeinschaftsorientierten

Die Gemeinschaftsorientierten legen besonderen Wert auf die Entwicklungsgemeinschaft und den Austausch mit anderen Entwickler:innen. Dementsprechend engagieren sie sich in großen FLOSS-Projekten mit vielschichtigen Austauschprozessen zwischen Entwickler:innen und Nutzer:innen. Ihr Engagement ist viel mehr prozeßorientiert als bei den anderen Typen. Während bei den Maintainer:innen und den professionellen Programmierer:innen eine effizient agierende Entwicklungsgemeinschaft ein Werkzeug ist, um ein bestimmtes Ziel zu erreichen, sind die Geimeinschaftsorientierten vor allem an den zwischenmenschlichen Beziehungen und Themen, abseits der eigentlichen Entwicklungsar-

beit, interessiert.

Bezogen auf die Entwicklungsgemeinschaft sind sie in stärkerem Maße auf die Menschen und ihrer Denk- und Lebensweise fokussiert. Dies äußert sich in der Bereitschaft, auch Möglichkeiten des direkten Austauschs mit anderen Entwickler:innen wahrzunehmen, sofern dies im Rahmen des FLOSS-Projektes möglich ist. Dies sind zum Beispiel Konferenzen, Arbeitstreffen oder regelmäßige Zusammenkünfte in lokalen Strukturen.

Personen dieser Gruppen beziehen Selbstvertrauen aus der Teilnahme an der gemeinsamen Praxis der Entwicklungsgemeinschaft. Die Zusammenarbeit als Gemeinschaft bekommt einen Wert an sich. Dies geht mit der Einsicht einher, dass das koordinierte Agieren als Gemeinschaft der Entwicklung als Einzelperson vorzuziehen ist, um langfristig FLOSS entwickeln zu können. Alle Entwickler:innen, die diesem Typus zugeordnet wurden, thematisierten ihren Aufbau von Sozialkompetenz in Bezug auf den Umgang mit anderen Entwickler:innen der Entwicklungsgemeinschaft. Ebenso heben sie die gesellschaftliche Bedeutung von FLOSS als öffentliches Gut stärker hervor. Bei den Beweggründen für ihr FLOSS-Engagement stehen gesellschaftskritische Überzeugungen wie zum Beispiel soziale Teilhabe stärker im Vordergrund.

10.2.4 Die soziale Unterstützer:innen

Der Typus der sozialen Unterstützer:innen ist, ähnlich wie die Gemeinschaftsorientierten, an den Austauschprozessen in einer Gemeinschaft interessiert. Allerdings haben sie einen anderen Bezugsrahmen: Sie fühlen sich nicht einem bestimmten FLOSS-Projekt verbunden. Stattdessen fühlen sie sich einer sozialen Entität verbunden, die aus einer oder mehreren Personen bestehen kann. Die soziale Entität oder Teile von ihr engagieren sich in einem FLOSS-Projekt. Die Beziehung zwischen sozialen Unterstützer:innen und einem FLOSS-Projekt ist also nur indirekt. Soziale Unterstützer:innen sind hauptsächlich an der gemeinsamen Praxis mit der sozialen Entität interessiert. Das Engagement in einem FLOSS-Projekt erfolgt aus Unterstützung des Engagements der sozialen Entität. In den Interviews bestanden die sozialen Entitäten aus wichtigen Personen im sozialen Umfeld wie beispielsweise enge Freund:innen oder Lebenspartner:innen. Soziale Entitäten können aber auch aus informellen Lernumgebungen bestehen, wie sie bereits im Kapitel 8.4.4 in Form von Hackerspaces und ähnlichen Strukturen beschrieben wurden.

Soziale Unterstützer:innen engagieren sich dort in FLOSS-Projekten, wo sie ein inhaltliches Interesse haben. Sie orientieren sich dabei an den Interessen von Gleichgesinnten. Sie engagieren sich mit ihnen zusammen und das verbindende Element ist die gemeinsame Auseinandersetzung mit einem Thema. Dementsprechend bilden sie ein lokales Lernnetzwerk, in welchem Lernprozesse in direktem Austausch mit anderen Lernenden ihres sozialen Umfelds erfolgen. Ähnlich wie die Maintainer:innen, lernen sie viel durch die Praxis in FLOSS-Projekten. Der Unterschied ist, dass sie dabei auch auf lokale Strukturen ihres Lernnetzwerkes zurückgreifen. Daraus ergibt sich ein hoher Anteil von synchroner Kommunikation im direkten Austausch mit anderen Entwickler:innen, während bei den

anderen Typen die internetgestützte Kommunikation überwiegt.

10.3 Repräsentative Fallinterpretation

Die folgenden Einzelfallinterpretationen beschreiben je einen repräsentativen Einzelfall der vorgestellten Typologie. Daran werden die externe Heterogenität zwischen den Typen anhand eines Einzelfalles herausgearbeitet.

Die Einzelfälle können nicht immer eindeutig einem Typus zugeordnet werden. Da fast alle Entwickler:innen in mehr als einem FLOSS-Projekt aktiv sind oder waren, kann sich der situative Kontext des Engagements in diesen FLOSS-Projekten stark unterscheiden. So kann es sein, dass eine Person in Abhängigkeit vom jeweiligen FLOSS-Projekt gleichzeitig unterschiedlichen Typen zugeordnet werden kann. In der folgenden Fallinterpretation werden die Merkmale herausgearbeitet, die für die Zuordnung zu einem bestimmten Typus ausschlaggebend waren.

10.3.1 Sascha: der Maintainer

Die folgende Fallinterpretation illustriert am Beispiel des Entwicklers Sascha den Typus der Maintainer:innen. Er gründete ein eigenes FLOSS-Projekt und entwickelt es mit Hilfe anderer Entwickler:innen beständig weiter. Neben diesem Hauptprojekt, was einen Großteil seines Engagements ausmacht, steuert er immer wieder auch zu anderen Projekten Fehlerkorrekturen bei, wenn er kleinere Probleme bei von ihm genutzter Software entdeckt. Dies setzt voraus, dass die betreffenden Programme in einer Programmiersprache geschrieben sind, die er bereits beherrscht. Diese Form des Engagements ist jedoch nur auf einmalige Beiträge begrenzt und es erfolgt keine längerfristige Interaktion mit den betreffenden Entwicklungsgemeinschaften. Im Gegensatz dazu gilt sein hauptsächliches Interesse dem FLOSS-Projekt, welches er selbst initiiert hat. Dabei fungierte er von Anfang an als Maintainer des FLOSS-Projektes. Die Gründungsidee beschreibt Sascha folgendermaßen:

> „Ich hatte irgendwie keinen Bildbetrachter, der mich komplett befriedigt. Und dann habe ich mir gedacht: Ja, wie witzig wäre es denn, das mal so selber zu schreiben. Und dann habe ich es natürlich im ganz kleinen Rahmen selber mal probiert. Dann ging das einigermaßen und dann habe ich es auf GitHub geschoben und irgendwann kamen dann tatsächlich Leute, die das auch verwendet haben. Und dann ist es auch einfach gewachsen. Ja und seit dem/ Die Motivation war einfach, es gibt nichts, was mich komplett befriedigt und dann ist mir aufgefallen, es gibt tatsächlich mehr Leute denen es genauso geht und die jetzt irgendwie mein Projekte auch cool finden.“ (Dokument I-11; Abs. 16)

Das Interesse an einem eigenen FLOSS-Projekt bestand aber nicht nur aus dem empfundenen Mangel an einer verfügbaren Software. Ihm war es von Anfang an wichtig, seine

Software auch anderen zur Verfügung zu stellen. Zu diesem Zweck war Sascha in verschiedenen Internetforen aktiv, in denen sich Programmierer:innen über Softwareideen austauschen und auch die ersten Umsetzungsversuche miteinander teilen.

Die Reaktionen von anderen Entwickler:innen sind ein wichtiges Schlüsselelement. Je nach Art und Weise können sie eine Quelle für Anerkennung sein und damit den qualitativen Sprung von einer Idee mit einem einfachen Prototypen bis hin zu einem eigenständigen FLOSS-Projekt bewirken:

> „Im Idealfall gibt es halt konkrete Ziele, die von irgendwelchen Usern nachgefragt wurden und dann ist es immer recht schön, wenn ich dann nachfragen kann und dann sehe ich: Ah, die wollen das wirklich und da ist Interesse da! Und dann ist halt die Motivation entsprechend viel größer, als wenn man jetzt sich denkt: Ach, dieses Feature will ich ja nicht unbedingt und das braucht ja doch keiner und dann macht man halt irgendwie nichts.“ (Dokument I-11; Abs. 92)

Rückmeldungen anderer Entwickler:innen waren aber nicht nur als positive Bestärkung für den Maintainer relevant, auch für den eigentlichen Lernprozess waren sie essentiell. Die Veröffentlichung des Quellcodes als FLOSS hat auch eine ganz pragmatische Funktion:

> „Für mich selbst dann auch noch aus Eigennutz, weil ich habe damit Programmieren mehr oder weniger gelernt und ich bin sehr viel besser geworden dadurch, dass es Freie Software war, andere Leute einen Patch geschrieben oder mir Feedback geben. Das wäre einfach nicht möglich, wenn ich dies als kleines eigenes Projektchen (..) versuche Closed-Source zu/ Ja, nicht einmal wirklich zu veröffentlichen oder nur Binary Codes herzugeben, weil dann würde ich nie Feedback über Code-Qualität bekommen.“ (Dokument I-11; Abs. 20)

Dies unterstreicht die wichtige Funktion von interessierten Programmierer:innen beziehungsweise im späteren Verlauf die wachsende Entwicklungsgemeinschaft, die sich um ein FLOSS-Projekt herum aufbaut. So wurde von diesem Entwickler auch das erste Unterstützugsangebot eines anderen Entwicklers als wichtiges Schlüsselerlebnis beschrieben:

> „Ich habe es halt irgendwie in einem Foren-Post erst einmal veröffentlicht. Ganz klein, die Leute haben es ausprobiert und dann vielleicht/ Am Anfang ist das natürlich immer: Es werden massiv neue Features requested und ganz viele Ideen kommen rein und dann werden die erst einmal eingebaut und wenn sie merken, das Projekt wird tatsächlich einigermaßen unterstützt, kamen halt/ Die erste Person war jemand, der das Ganze tatsächlich packa-

> gen[167] wollte für NixOS[168] und dann gemerkt hat, es geht deswegen, deswegen und deswegen nicht. Hat alles eingebaut und entsprechend einen Pull-Request aufgemacht, passte dann weiterhin darauf auf, dass es irgendwie halbwegs mit dem kompatibel ist.“ (Dokument I-11; Abs. 56)

Sascha beschreibt, wie Stück für Stück die Zahl der an seinem FLOSS-Projekt interessierten Entwickler:innen stieg. Die ersten langfristig aktiven Mitglieder der Entwicklungsgemeinschaft waren Paket-Maintainer:innen. Diese achten hauptsächlich darauf, dass die FLOSS für eine konkrete GNU/Linux-Distribution kompatibel bleibt. Entweder sie leiten die Probleme, die im Kontext einer bestimmten GNU/Linux-Distribution auftreten, an den Maintainer weiter, oder sie beheben selbst die Programmfehler im Quellcode des Ursprungsprojektes. Im letzteren Fall würden die Paket-Maintainer:innen in dem FLOSS-Projekt die Rolle von Kernentwickler:innen einnehmen.

Bei kleinen FLOSS-Projekten ohne eigene Kommunikationskanäle, wie in diesem Beispiel, werden solche Austauschprozesse innerhalb der Entwicklungsgemeinschaft von den Werkzeugen des FLOSS-Portals *GitHub* mit entsprechenden Werkzeugen und Prozessen standardisiert verarbeitet oder per direkter E-Mail an einzelne Entwickler:innen kommuniziert:

> „Das hat eigentlich immer damit angefangen, dass die entweder ein Issue oder einen Pull-Request aufgemacht haben und das Ganze ist dann in irgendeine Art E-Mail-Kommunikation übergegangen, weil es doch recht/ Also es muss ja nicht jeder mitlesen, was wir da durchdiskutieren. Und jetzt, wenn ich eine neue Version irgendwie vorbereite oder weiß, es wird irgendwas passieren, bekommen die natürlich eine E-Mail oder ich flag[169] das entsprechend und erkläre, was los ist. Aber das ganze läuft über die offiziellen Issue-Tracker, über E-Mail oder über GitHub. Also jetzt nichts/ Kein IRC oder keine persönlichen Gespräche. Die Personen sind auch soweit verstreut, dass ich sie nicht persönlich treffen könnte.“ (Dokument I-11; Abs. 84)

Aufgrund seiner Rolle hat er sich den Umgang mit dem Versionsverwaltungssystem und weiteren Werkzeugen für die gemeinschaftliche Softwareentwicklung angeeignet. Ebenso die Planung aller notwendigen Schritte bei der Softwareentwicklung.

Die Einbindung neuer Entwickler:innen ist eine wichtige Hilfe, da dadurch die Arbeitslast aufgeteilt werden kann. Dementsprechend ist die Kommunikation ein wichtiger Faktor, um neue Entwickler:innen in die Entwicklungsgemeinschaft einzubinden:

---

[167] Der Prozess beschreibt die Integration einer Software für eine bestimmte Paketverwaltung. Dadurch wird sie für alle Nutzer:innen von GNU/Linux-Distributionen mit der jeweiligen Paketverwaltung einfach verfügbar. Die Installation erfolgt dann im Rahmen der Softwareverwaltung des Systems. Manuelles Kompilieren oder installieren der Software entfällt dadurch.

[168] NixOS ist eine GNU/Linux-Distribution.

[169] Aus dem Englischen übernommener Begriff (’to flag’ = kennzeichnen), um die Veröffentlichung einer neuen Softwareversion im FLOSS-Portal anzuzeigen.

„[...] man lernt natürlich, wie man auf Bug-Reports reagiert im Laufe der Zeit (..) und vielleicht lernt man auch, welche Kanäle gut für so etwas geeignet sind.“ (Dokument I-11; Abs. 156)

Hinter jedem Bug-Report steht eine Person, die mehr oder weniger gut das gefundene Problem beschrieben hat. Im Zweifelsfall braucht es Rückfragen zur Klärung der Ursache. Jeder Bug-Report ist auf dem FLOSS-Portal öffentlich einsehbar und dokumentiert auch den Umgang mit dem Fehler. Der Entwickler betont seinen Anspruch, möglichst schnell die Programmfehler zu beheben.

Die Kommunikation mit andere Nutzer:innen und Entwickler:innen beschreibt er als durchweg positiv:

„Was mich immer sehr beeindruckt hat ist das konstruktive Feedback, das man bei so einer Art Projekt bekommt, was meiner Meinung nach generell in der Gesellschaft nicht mehr so häufig der Fall ist, dass man, wenn man etwas sagt, konstruktiv das Feedback bekommt, mit dem Ziel, das eigene oder dieses Projekt oder das, was auch immer man vertritt, zu verbessern, anstatt irgendwie eine Gegenposition darzustellen oder das Eigene zu verkaufen. Und das finde ich in der gesamten Open-Source-Community sehr beeindruckend gelöst und überraschend regelmäßig mit positiven [unverständlich], man liest sehr wenig Negatives, es gibt sicherlich auch Communities, die recht vergiftet wurden, aber doch im Großen und Ganzen sehr positiv die Grundstimmung.“ (Dokument I-11; Abs. 132)

Die Austauschprozesse waren für ihn die Voraussetzung für sein selbstgesteuertes Lernen. Sascha hatte schon vor dem FLOSS-Projekt basale Programmierkenntnisse. Nach seiner Beschreibung hat er das eigentliche Programmieren und die Methoden der Softwaretechnik erst durch die Anwendung in seinem FLOSS-Projekt gelernt und das wäre ohne die Austauschprozesse mit anderen Entwickler:innen nicht möglich gewesen.

10.3.2 Alexis: der professionelle Programmierer

Alexis verfügt über eine formal erworbene Qualifikation als Programmierer. Als Student hat er sich in seinem ersten FLOSS-Projekt in einer studentischen Arbeitsgruppe engagiert. Ein weiteres FLOSS-Projekte, in dem er auch heute noch an unterschiedlichen Stellen mitwirkt, hat sein Interesse aber schon vor dem Beginn seines Informatikstudiums geweckt. Mittlerweile finanziert er über Dienstleistungen, die im Bezug zu dem FLOSS-Projekt stehen, sein Studium. Das im Folgenden näher beschriebene FLOSS-Projekt, führte zu Einordnung in den Typus der professionellen Programmierer:innen:

„Also ich biete im Grunde Dienstleistung an. Ich baue, schließe halt teilweise Menschen ans [FLOSS-Name]-Netz an, die das nicht selber machen wollen. Das ist so ein bisschen heraus entstanden aus der Flüchtlingskrise. Also im

> Rahmen der Flüchtlingskrise haben wir halt viele Unterkünfte für Geflüchtete an unser Netz angebunden. Eine Zeit lang haben wir das halt komplett ehrenamtlich gemacht. Irgendwann hat dann einfach die Arbeit/ Da war das dann einfach zu viel Arbeit oder Arbeiten, wo wir gesagt haben: Das wollen wir als Ehrenamtliche nicht machen. Und ich habe mich dann halt sozusagen ein bisschen selbstständig gemacht und gesagt: Okay, ich biete das dem Unterkunftsbetreiber an, das ich zum Beispiel innerhalb der Unterkunft dann Router installiere, die streng genommen nicht unbedingt was mit dem [FLOSS-Projekt]-Netz zu tun haben. Genau, wir bieten sozusagen/ Also ich baue im Grunde ganz traditionell Netzwerke in Büros, Unterkünften. Schließe die aber dann halt immer noch auch an das [FLOSS-Projekt]-Netz an und kriege dafür im Grunde Geld. Genau, das ist jetzt nicht so/ Genau das ist im Grunde, ich sehe das so als Dienstleistung.“ (Dokument I-2; Abs. 24)

Aus einem Hobby heraus entwickelte sich für Alexis die Möglichkeit, mit Dienstleistungen die mit der FLOSS in Verbindung stehen, Geld zu verdienen. Die Beweggründe des initialen Engagements waren soziale Teilhabe und die Möglichkeit der eigenen Partizipation. Zusätzlich war bei ihm auch ein thematisches Interesse an den zugrunde liegenden Technologien:

> „Aber, also für viele und auch für mich teilweise ist es halt ja auch einfach ein Hobby und du willst wie gesagt ja auch Spaß haben, du willst dir diese Technik erkunden/ Also, (.) gibt’s so ganz verschiedene Gründe, warum man das jetzt macht.“ (Dokument I-2; Abs. 20)

Die Möglichkeit, auf Grundlage der FLOSS kommerzielle Dienstleistungen anbieten zu können, war eher ein pragmatischer Entschluss von ihm und nicht von Anfang an intendiert. Es überwiegen die von der Lohnarbeit unabhängigen Beweggründe für das Engagement. Dennoch merkt er an, dass die Reputation aufgrund seines FLOSS-Engagements für sein späteres Berufsleben von Bedeutung sein kann:

> „Das ist auch einfach wertvoll zu wissen, wie du jetzt eine Änderung einbringst in so ein Projekt. Mit der Zeit wirst du dann vielleicht da auch erfolgreicher. Ja, das denke ich, ist auch ein Tool, was man dann nutzen kann irgendwie, ’ne. Und einem dann vielleicht auch, sagen wir mal, in irgendeinem Beruf hilft. Wenn jetzt die Firma da mit Freier Software arbeitet. Ja, also ich denke, es kann einfach auch auf das Berufsleben vorbereiten.“ (Dokument I-2; Abs. 130)

Als wichtig für ein späteres Berufsleben empfindet er seine Erfahrungen in der gemeinschaftlichen Softwareentwicklung. Dies impliziert auch das Projektmanagement, Kenntnisse im Umgang mit verschiedenen Entwicklungswerkzeugen und die praktische Anwendung von diversen Programmiersprachen. Hierbei unterscheidet er klar zwischen theo-

retischen Kenntnissen von Programmiersprachen und der Anwendung in konkreten Einsatzszenarien:

> „[Du] Kannst natürlich Informatik studieren und dann programmierst du auch ein bisschen im Studium, aber wenn du darüber hinaus nichts machst, dann hast du im Endeffekt dieses Handwerk vielleicht nicht gelernt oder sage ich mal (.) ja, also du kannst die Grundlagen ein bisschen, aber hast einfach keine Übung und ich glaube, zu so einem Handwerk gehört einfach die Übung. Programmieren ist das dann. Da braucht man einfach Übung für, muss viel Code gelesen haben, gesehen haben, selbst geschrieben haben und ja, Freie Software hilft dabei natürlich.“ (Dokument I-2; Abs. 130)

Charakteristisch für die Mitglieder dieses Typus ist die Bedeutung des Programmier- und Dokukumentationsstils. Unter anderem resultiert dies daraus, dass Alexis sich seine Programmierfähigkeiten viel durch das Lesen und Verstehen von Quellcodes anderer Entwickler:innen erarbeitet hat. Dies führte zu einer Sensibilisierung für einen sauberen und nachvollziehbaren Programmierstil. Der ist eine wichtige Grundlage für die gemeinschaftliche Softwareentwicklung. Andere Entwickler:innen müssen den Quellcode verstehen können, um darauf aufzubauen oder seine Funktion beurteilen zu können.

Dies ist auch der zweite wichtige Punkt, wie das Programmieren bei diesem Typus gelernt wird: Gemeinschaftliche Softwareentwicklung in einem FLOSS-Projekt beruht auch auf einer Feedback-Kultur. Entwickler:innen fordern sich Rückmeldungen zu ihrem Quellcode ein.

> „Also, gerade wenn man viel macht oder ein tieferes Verständnis davon hat, dann ist es eigentlich so, dass man sehr lange einfach vor dem Problem sitzt, viele Informationen versucht zu sammeln und meistens das Problem eher alleine löst und sich dann noch einmal zur Lösung vielleicht ein Feedback holt.“ (Dokument I-2; Abs. 44)

Je nachdem, wie komplex das Problem oder die gewünschte Funktionalität ist, finden auch vor der Implementation Absprachen über den Lösungsweg statt. Teilweise ist das Problem aber auch so spezifisch, dass nur wenige Entwickler:innen in dem Projekt über die nötige Sachkompetenz zum Problemverständnis verfügen. In diesem Fall greift die solitäre Lösungsstrategie und erst nach der Implementation wird um ein konstruktives Feedback durch andere Entwickler:innen gebeten.

10.3.3 Mika: die Gemeinschaftsorientierte

Beim ersten FLOSS-Projekt der Entwicklerin unterstützte sie ihren Lebensgefährten bei der Erstellung einer komplexen Webseite und bei der Übersetzung in mehrere Sprachen. Bei dem darauffolgenden Projektengagement erlebte sie das erste Mal die intensiven Austauschprozesse einer differenzierten Entwicklungsgemeinschaft mit einer zwei- bis drei-

stelligen Anzahl an Mitgliedern. Der Anlass war kein funktionaler Fehler bei der eigenen Nutzung, was sonst für viele Entwickler:innen der Einstieg ist, sondern die schlechte Übersetzung der Software in der von ihr genutzten Sprache. Anfangs engagierte sich Mika als Übersetzerin, später wurde sie auch in den Bereichen Softwaretest, Dokumentation, Öffentlichkeitsarbeit und der Unterstützung von Nutzer:innen aktiv. Die Erfahrung dieser Entwicklungsgemeinschaft sind Gegenstand der folgenden Zitate und waren ausschlaggebend für die Zuordnung zu dem Typus der Gemeinschaftsorientierten. Den Stellenwert der Entwicklungsgemeinschaft für ihr persönliches Engagement beschreibt die Entwicklerin wie folgt:

> „Also das [die Entwicklungsgemeinschaft] ist ja schon das, was das [unverständlich] ausmacht, sonst wäre es echt öde. Manchmal macht es echt richtig Spaß, mit den Leuten zusammenzuarbeiten. Was weiß ich. Es gibt zum Beispiel einen Mitübersetzer, mit dem führe ich dann detaillierte Diskussionen über irgendwelche winzigen Formulierungsabstufungen und das macht/ Ich glaube für 90% aller Leute wäre es total nervig, aber wir beide haben Spaß dabei. Oder, (..) ja, alleine möchte ich daran nicht arbeiten, das ist einfach öde.“ (Dokument I-3; Abs. 161)

Hier wird ihr Fokus auf die Austauschprozesse mit den anderen Entwickler:innen deutlich. Während bei den beiden Typen der Maintainer:innen und Programmierer:innen eine defekte oder fehlende Funktionalität für das Engagement entscheidend ist, sind es bei den Gemeinschaftsorientierten der Austausch in der Gemeinschaft. Das wechselseitige Engagement innerhalb der Entwicklungsgemeinschaft ist vorrangig an den Zielen des FLOSS-Projektes ausgerichtet, beinhaltet jedoch auch Raum für themenfremde Inhalte. So kommt es auch zum Aufbau von zwischenmenschlichen Beziehungen zwischen den Mitgliedern der Gemeinschaft, in der die Entwickler:innen auch andere Perspektiven ihres Lebensalltags miteinander teilen. Auf die Frage nach den Inhalten der Kommunikation mit anderen Mitgliedern der Entwicklungsgemeinschaft antwortete Mika:

> „Zu 98% ist das [FLOSS-Projekt]-zentriert, denke ich. Es gibt irgendwie ein paar Leute, die, also mit denen unterhalte ich mich dann auch nochmal darüber, wie, was weiß ich, wie die jetzt auf Linux umziehen oder (.) keine Ahnung (.). Na gut, einer zum Beispiel spricht mit mir darüber, wie gerade sein Vater/ Quatsch, seine Mutter stirbt oder so, aber jetzt/ Oder wenn es irgendwelche/ Ja obwohl, wenn es Konflikte im Projekt gibt, dann ist es ja auch mit [FLOSS-Projekt] verbunden.“ (Dokument I-3; Abs. 165)

Im Gegensatz zu den anderen Typen nutzen die Gemeinschaftsorientierten auch mehr Kommunikationskanäle. Ein besonderen Stellenwert haben direkte Treffen der Entwicklungsgemeinschaft. Dies kann in Form von Arbeitstreffen geschehen, an denen bestimmte Entwicklungsziele erreicht werden sollen oder Treffen, die der Planung und Zielbestimmung dienen.

> „Also es hat sich mittlerweile etabliert, denke ich. Ich glaube, wir haben jetzt das dritte oder vierte Hackfest[170] gehabt, wo sich die Entwickler treffen und ja, es ist halt schon irgendwie ganz schön, wenn die Wege nicht so weit sind. Wenn man da halt einfach direkt miteinander diskutieren kann über das, was man da gerade macht und die Leute sind oft auch ein bisschen motivierter. Also nach dem diesjährigen Hackfest habe ich/ Meine ich zumindest einen deutlichen Anstieg der Aktivität vermerkt zu haben.“ (Dokument I-3; Abs. 257)

Der Austausch zwischen den Entwickler:innen ist entweder einer der Beweggründe oder zumindest ein wichtiger Faktor, um die Fortsetzung des Engagements zu gewährleisten. Da Austauschprozesse auch immer zu Konflikten führen können, ist die Lösung von sozialen Konflikten auch eine Kompetenz, die Gemeinschaftsorientierte aufbauen:

> „Ja, ich glaube schon. Ich denke, ich habe ein bisschen besser den Umgang miteinander in internationalen Gemeinschaften gelernt. Das ja/ (.) Ich war noch schlimmer früher als heute.“ (Dokument I-3; Abs. 237)

Da Mika neben der eigenen Übersetzungsarbeit auch die Koordinierung des gesamten Übersetzungsteams für das FLOSS-Projekt übernimmt, liegen auch die Absprachen der Beiträge mit allen Übersetzer:innen bei ihr. Die Bedienoberfläche der Software ist mittlerweile in über ein Dutzend Sprachen übersetzt worden und wird stetig erweitert. Dazu kommen noch die Übersetzungen der programminternen Hilfe, Tutoriale und der Webseite des Projektes. Neben der Sozialkompetenz im Umgang mit anderen Mitgliedern der Entwicklungsgemeinschaft berichtet sie insbesondere von einem gestiegenen Selbstvertrauen in die eigenen Fähigkeiten:

> „Keine Ahnung, also, vielleicht bin ich ein bisschen selbstständiger geworden, einfach dass ich mich mehr traue, Sachen einfach selber auszuprobieren am Computer. Aber das ist ja jetzt irgendwie nicht so überraschend. Ja (..) und klar, technisch habe ich sicherlich so Einiges gelernt.“ (Dokument I-3; Abs. 241)

Das gewachsene Vertrauen ist auch auf die gestiegene Sachkompetenz der Entwicklerin zurückzuführen. Sie nutzte verschiedene Online-Ressourcen, um sich selbst die Programmiersprache des Content-Management-Systems[171] anzueignen, mit welchem die Webseite des FLOSS-Projektes arbeitet. Aufgrund ihrer Sprachübersetzung der Webseite sind ihr Programmfehler auf der Webseite aufgefallen, die sie nun selbst beheben konnte. Als treibende Kraft des Kompetenzaufbaus war aber immer die Freude am Austausch mit den

[170] Ein Hackfest ist ein Veranstaltungsformat, bei dem die Beteiligten innerhalb der Dauer der Veranstaltung zusammen bestimmte Probleme bei der FLOSS-Entwicklung beheben oder einen festgelegten Entwicklungsstand erreichen wollen.

[171] Ein ’Content-Management-System’ ist eine Software, die die gemeinschaftliche Erstellung und Bearbeitung von Inhalten ermöglicht. Die Nutzer:innen können Inhalte damit intuitiv bearbeiten und veröffentlichen, ohne dass sie tiefere Kenntnisse von Programmier- oder Auszeichnungssprachen benötigen.

anderen Entwickler:innen ausschlaggebend. Dennoch benennt Mika auch andere Aspekte, die für sie eine Rolle spielen:

> „Ja, also man sagt das ja immer, so eine/ So Softwarefreiheit und so. Aber ich glaube das ist gar nicht der Hauptbeweggrund. (.) Klar ist es irgendwie cool, wenn man sieht, dass Leute auf der ganzen Welt, auch solche, die kein Geld haben oder so, sich das einfach herunterladen können und damit irgendwas Schönes, Tolles machen können und so. Aber für mich ist es hauptsächlich, ja, es macht halt Spaß, mit den Leuten zu arbeiten, es ist (.) es macht Spaß, das was ich da tue im Projekt. Ja, ich lerne halt auch viel. Ich habe viele Gelegenheiten Sachen auszuprobieren und Dinge zu testen, ja.“ (Dokument I-3; Abs. 25)

Sie benennt neben dem eigenen Spaß bei der Entwicklungsarbeit auch den normativen Wert der sozialen Partizipation für ihr Engagement. Durch ihre Aktivität als Übersetzerin und Koordinatorin hat sie einen maßgeblichen Anteil an der Ausweitung des Kreises der potentiellen Nutzer:innen der FLOSS. Daneben unterstützt Mika auch einzelne Nutzer:innen der FLOSS. In verschiedenen Foren melden sich mittlerweile Nutzer:innen, die bei konkreten Anwendungsszenarien mit der Software Probleme haben. Auch in diesem Bereich ist sie aktiv und nutzt ihre Kompetenzen, um Nutzer:innen zu helfen.

10.3.4 Kyle: der soziale Unterstützer

Diese Fallinterpretation bezieht sich auf einen Vertreter der sozialen Unterstützer:innen. Der Entwickler Kyle ist aktiv in einem Hackerspace. Dieser dient gleichzeitig auch als Treffpunkt einer Linux User Group und eines Chaostreffs[172]. Sein Engagement bei verschiedenen FLOSS-Projekten erfolgte aufgrund von Problemen, die die eigene Nutzung beeinträchtigten:

> „Bei mir ist oft so, die Motivation erst mal eher: Da ist irgendwas kaputt, was ich gerade brauche und dann habe ich bei Open-Source dann eben die Möglichkeit, da einfach das Projekt bei GitHub oder GitLab zu suchen, einen Fork davon zu machen, die Änderung zu machen und den Pull-Request zu machen. [...] Für mich ist es immer recht interessant, wenn ich mir den Code angucken kann, wenn ich sagen kann: Ich kann lernen, warum die Software irgendwas macht, warum die sich zum Beispiel auch gerade nicht so verhält, wie ich das wünsche. Ich kann das natürlich dann auch anpassen und eben gerade dieser Punkt auch, ein Fehler, den ich beheben kann, weil ich reingucke und sehe: Da ist bei zwei Zeilen etwas falsch, das kann ich fix beheben. Das ist halt das, was ich bei geschlossener Software nicht kann. Das ist so, ja, einfach diese

[172] Ein Chaostreff ist ein regelmäßiges informelles Zusammentreffen von Personen, welche sich dem *Chaos Computer Club* verbunden fühlen.

> Möglichkeit: Ich habe den Quellcode verfügbar, ich kann ihn anpassen oder ich kann auch einfach daraus lernen.“ (Dokument I-1; Abs. 5)

Der Entwickler benennt pragmatische Gründe, die die Nutzung von FLOSS für ihn attraktiv macht: Er hat die Möglichkeit, daran Veränderungen vorzunehmen beziehungsweise kann anhand des Quellcodes der Software diese Fähigkeiten erlernen. Sein Lernen steht auch in Verbindung mit dem Hackerspace, der die Funktion eines Lernnetzwerkes hat. Hier trifft Kyle auf Gleichgesinnte, die ähnliche Ambitionen verfolgen:

> „Und das ist dann auch ein bisschen motiviert eben durch den Hackspace bei uns im Ort, wo es dann halt auch/ Jemand hat es da halt gesehen, hat es dann auf der Mailingliste geschrieben und so und dann/ Hat man im Hackspace unteinander so: Ja, wollen wir uns mal treffen und/ (.) Das jeder so ein bisschen an seinen Sachen da arbeiten kann und da Pull-Requests machen kann.“ (Dokument I-1; Abs. 17)

In diesem Beispiel engagieren sich die Hackerspace-Mitglieder zwar nicht im gleichen FLOSS-Projekt, wie bei anderen Vertreter:innen dieses Typus, dennoch wird im Lernnetzwerk dazu aufgerufen, dies gemeinschaftlich zu tun. Die Mitglieder fokussieren ihre jeweiligen FLOSS-Projekte und nutzen dies als einen Anlass für ein Treffen. Durch den persönlichen Austausch gibt es auch die Möglichkeit der gegenseitigen Hilfestellung:

> „Ich hatte vor längerer Zeit mal versucht, für Arch-Linux einen Window-Manager zu paketieren und da scheiterte es dann beim Kompilieren an irgendeiner bestimmten Stelle und da hatte ich dann auch im Hackspace jemanden gefragt: Kannst du mal reingucken? Und mit dem haben wir es dann soweit aussortieren können, dass wir dann wussten: Okay, hier brauchen wir zum Beispiel nicht weitermachen, weil, da müsste man eben an dem Projekt wesentlich weiter entwickeln. Und das, (.) ja, (.) das ist durchaus wichtig eben, aber bei mir beschränkt sich das mehr so auf das Hackspace-Umfeld. Man kennt da die Leute und man kann auch in etwa einschätzen, wer zu welchem Thema ein bisschen was sagen kann und es sind auf alle Fälle wichtige Quellen.“ (Dokument I-1; Abs. 25)

Die Gemeinschaft des Lernnetzwerkes ist auch als Wissensressource bei der Problemlösung von Bedeutung. Ähnlich wie die Entwicklungsgemeinschaft in einem FLOSS-Projekt kennen die Mitglieder die Kompetenzen der anderen Personen ihres Netzwerkes. Auf diese können sie bei Bedarf zurückgreifen. Da die FLOSS-Entwicklung in diesem Lernnetzwerk ein gemeinsames Interesse darstellt, ist diese Art der Unterstützung Teil der gemeinschaftlichen Praxis dieses Lernnetzwerkes. Über die Unterstützung von Personen mit solchen Fragestellungen werden andere Mitglieder des Lernnetzwerkes in das Engagement bei deinem FLOSS-Netzwerk einbezogen. Teilweise führt eine solche Hilfestellung zu einem längerfristigen Engagement der hilfegebenden Person bei dem FLOSS-Projekt.

Das Lernnetzwerk ist aber nicht nur eine Unterstützungsstruktur für andere FLOSS-Projekte. Das Lernnetzwerk selbst zeigt Praktiken, die denen einer FLOSS-Entwicklungsgemeinschaft ähneln. Dies zeigt beispielsweise die Diskussion, die in dieser Form auch von einer Entwicklungsgemeinschaft einer spezifischen Software hätte stammen können:

> „[...] na, das Umfeld macht es auch so ein bisschen noch mit, wenn man eben sich damit näher beschäftigt und dann auch Kontakt zu Communities hat oder eben durch so etwas wie dem Hackspace. Und dort wird dann auch regelmäßig diskutiert: Ja welche Lizenz ist denn jetzt so das Ideale? Also bei uns gibt es dann so zwei Fraktionen: Eine Fraktion, die eher so den freiheitlichen Anspruch hat, lieber die MIT-Lizenz einsetzt und sagt, ich sehe das ganze freiheitlich, ich will halt nicht jemanden direkt vorschreiben, was er jetzt genau mit dem Quellcode danach noch macht. Während die Anderen halt sagen: Ja, GPL, ich will halt wenigstens, dass es nicht wieder geschlossen werden kann.[173]“ (Dokument I-1; Abs. 45)

Hier kommt die Ähnlichkeit zu dem Typus der Gemeinschaftsorientierten zum Tragen. Es gibt nicht nur gegenseitige Hilfestellung, sondern es finden auch multilateral Austauschprozesse zwischen den Mitgliedern statt, die sich an gemeinsamen Interessen orientieren und sich auch in gemeinschaftliche Unternehmungen niederschlagen können.

10.4 Zusammenfassung

Die typologische Inhaltsanalyse führte zu einer Typologie mit vier verschiedenen Typen. Charakteristische Muster der Merkmalskonstellationen zeigten einerseits unterschiedliche Interesse beim Engagement in FLOSS-Projekten, andererseits Unterschiede im Kompetenzaufbau. Die Abweichungen im Kompetenzaufbau lassen sich unter anderem auf die jeweiligen Rollen und Interessen beim Engagement zurückführen.

Zusammenfassend wird festgestellt, dass *Maintainer:innen* viel Sachkompetenz im Bereich der Softwaretechnik aufbauen. Der Aufbau von sozialen Kompetenzen bezieht sich in erster Linie auf zwei Aspekte: Kommunikation mit Personen, die Fehlerberichte einreichen und dem Bestreben, neue Entwickler:innen für das FLOSS-Projekt zu gewinnen. Bei kleinen FLOSS-Projekten sind sie die Universalist:innen: Sie beschäftigen sich unter anderem mit Programmierung, Projektmanagement und Öffentlichkeitsarbeit. Der Typus der *Professionellen Programmierer:innen* hingegen widmet sich vorrangig der Programmierung. Sie betreiben Softwareentwicklung nicht nur als Hobby, sondern auch als Lohnarbeit. Ihr Kompetenzaufbau liegt im Bereich der gemeinschaftlichen Softwareentwicklung. Dies betrifft die Besonderheiten in der Art und Weise des Programmierens und der Dokumentation. Die *Gemeinschaftsorientierten* sind vorrangig an den Austauschprozessen

[173] Die GPL-Lizenz verlangt, dass Änderung von Software, die mit dieser Lizenz geschützt ist, auch wieder unter den Bedingungen der GPL veröffentlicht werden müssen. Diese Praxis wird auch als *Copyleft* bezeichnet. Die MIT-Lizenz hingegen beinhaltet keinen solchen Passus. Abgewandelte Software, die ursprünglich unter den Bedingungen der MIT-Lizenz veröffentlicht wurde, könnte dementsprechend auch wieder mit einer proprietären Lizenz (*Copyright*) versehen werden.

mit Entwickler:innen und Nutzer:innen einer konkreten FLOSS interessiert. Sie schätzen die zwischenmenschlichen Beziehungen und sind auch an der Lebenswelt anderer interessiert. Ihr Interesse an der FLOSS-Entwicklung ist stärker normativ geprägt als bei den anderen Typen. Sie thematisieren vor allem den Aufbau von sozialen Kompetenzen durch ihr Engagement in der Entwicklungsgemeinschaft. Im Gegensatz zu den Gemeinschaftsorientierten haben die *Sozialen Unterstützer:innen* einen anderen Bezugspunkt zu ihrem FLOSS-Engagement. Sie unterstützen nicht ein einzelnes FLOSS-Projekt, sondern das Engagement von für sie wichtigen Bezugspersonen. Mit diesen engagieren sie sich in einem Lernnetzwerk. Sozusagen ist das eigene Lernnetzwerk aus Gleichgesinnten, wie zum Beispiel ein Hackerspace, das Zentrum ihres Engagements. Sie interessieren sich für die Themen der FLOSS-Projekte, mit denen sich die Bezugspersonen beschäftigen. Es erfolgt ein gemeinsames Engagement mit der präferierten sozialen Entität in einem FLOSS-Projekt. Sie lernen viel durch direkte Austauschbeziehungen mit Gleichgesinnten in ihrem lokalen Umfeld.

Durch die Typologisierung konnte gezeigt werden, dass es große Unterschiede zwischen den individuellen Interessen beim FLOSS-Engagement gibt, der sich wiederum auf den Kompetenzaufbau auswirkt. Die aufgezeigten Typen unterscheiden sich auch in ihren präferierten Lösungsstrategien und damit in ihren Lernprozessen, wie sie sich neue Fähigkeiten und Fertigkeiten aneignen.

11 Schlussfolgerungen und Fazit

Zu Beginn dieses Kapitels werden die wichtigsten Analyseschritte dieser Arbeit zusammengefasst. Es wird anhand früherer Forschungsarbeiten zu diesem Thema begründet, weshalb dieses Vorgehen dazu geeignet ist, fehlende Perspektiven in einem multidimensionalen Ansatz zu vereinen. Dadurch kann das informelle Lernen Erwachsener in internetgestützten Gemeinschaften in einer Weise beschrieben werden, wie es in vorherigen wissenschaftlichen Abhandlungen noch nicht durchgeführt wurde. Zu diesem Zweck wird insbesondere der Aufbau von Kompetenzen in unterschiedlichen Kompetenzbereichen und deren Rahmenbedingungen betrachtet.

Im Anschluss daran wird der wissenschaftliche Ertrag dieser Arbeit in den gegenwärtigen Diskurs zu informellem Lernen eingeordnet. Schwerpunktmäßig handelt es sich um selbstgesteuertes Lernen anhand von Problemen, deren Lösung für die Entwickler:innen sinnstiftend ist. Vielfach sind sie dafür auf die Unterstützung anderer Entwickler:innen angewiesen, weswegen das Lernen im sozialen Kontext für den individuellen Kompetenzaufbau nicht außer Acht gelassen werden kann. Dementsprechend wird die netzwerkorientierte Lerntheorie der *Communities of Practice* auf ihr Potential zu kooperativem und kollaborativem Lernen Erwachsener hin bewertet und fehlende Dimensionen dieser Theorie herausgearbeitet. Da die Lernprozesse der Mitglieder von dezentralen Praxisgemeinschaften unter den besonderen Bedingungen von zeit- und raumdistanter Telekommunikation stattfinden, werden diesbezügliche Probleme und Potentiale für das selbstgesteuerte Lernen aufgezeigt.

Darauf aufbauend werden sechs methodische und didaktische Gestaltungsprinzipien herausgearbeitet, wie informelles Lernen in internetbasierten Lernnetzwerken den Bedürfnissen der darin involvierten Lernenden gerecht werden kann. Anschließend werden die Erkenntnisse über das informelle Lernen in Gemeinschaften in einen größeren Zusammenhang eingeordnet. Hierfür wird auf dem im Theorieteil dieser Arbeit vorgestellten *Capability Approach* von Martha Nussbaum zurückgegriffen. Nussbaums Konzept des *Guten Lebens* wird auf die Anwendbarkeit für die Mitglieder der untersuchten Praxisgemeinschaften hin überprüft. Weiterhin soll an Diskurse angeknüpft werden, die den Fähigkeitenansatz als Perspektive zur Weiterentwicklung einer kritischen Bildungstheorie betrachten.

Zum Abschluss dieser Forschungsarbeit wird das Potential weitergehender Forschung an internetbasierten Praxisgemeinschaften thematisiert. Anhand der Erkenntnisse aus dem Forschungsdesign der vorliegenden Arbeit werden Möglichkeiten aufgezeigt, um einzelne Aspekte des informellen Lernens in zukünftigen Forschungsarbeiten gezielter erfassen zu können.

11.1 Erweiterung des Forschungsstandes zu FLOSS-Entwickler:innen

Die vorliegende Forschungsarbeit leistet einen wichtigen Beitrag zum Verständnis der Interessen, Tätigkeiten und des Kompetenzaufbaus beim Engagement in FLOSS-Projekten. Ein Großteil der bereits veröffentlichten Forschungsarbeiten über die Entwickler:innen von FLOSS sind dem quantitativen Paradigma zuzuordnen. Sie bieten wenig Einsicht über die Komplexität des informellen Lernens durch ein FLOSS-Engagement aus Sicht der Entwickler:innen, da sie nur in Ansätzen das individuelle und gemeinschaftliche Lernen und dessen Rahmenbedingungen thematisieren.

Die wenigen qualitativen Studien auf diesem Gebiet, wie beispielsweise Tepe und Hepp (2008b) oder Bolstad (2006) beschäftigen sich mit einigen Teilaspekten des Lernens. Tepe und Hepp konzentrieren auf die Beschreibung einiger Beweggründe für ein FLOSS-Engagement. Gleichzeitig schlagen sie eine Brücke zu wesentlichen Einflüssen durch die Entwicklungsgemeinschaft, welche für diese Forschungsarbeit ein zentrales Analysekriterium darstellte. Bolstad fokussierte sich im Vergleich zu Tepe und Hepp in seiner autoethnografischen Studie auf seine eigenen Lernprozesse als Entwickler in Interaktion mit einer FLOSS-Gemeinschaft. Dadurch war ihm nur die isolierte Perspektive auf eine Person möglich.

Die hier vorgelegte Forschungsarbeit verfolgt einen holistischen Ansatz zum Verständnis vom Kompetenzaufbau beim Engagement in FLOSS-Projekten. Dadurch lässt sich erstmals das informelle Lernen in internetgestützten Praxisgemeinschaften mit allen relevanten Einflussgrößen darstellen und in ihrer Bedeutung für die befragten Entwickler:innen einschätzen. Unterschiede im Aufbau von Teilkompetenzen in unterschiedlichen Kompetenzbereichen mit unterschiedlichen Interessen in situierten Kontexten der Entwickler:innen lassen sich durch diesen Ansatz in Beziehung setzen.

Eine oft vernachlässigte Einflussgröße ist der soziale Kontext von Lernprozessen: Netzwerkorientierte Lerntheorien, die sich am Sozialkonstruktivismus orientieren, berufen sich auf die sozial vermittelte Konstruktion von Wirklichkeit, die sich damit auch auf das individuelle Lernen auswirkt. Als Analyseraster für die vorliegende Forschungsarbeit fungiert hierbei die Theorie der *Communities of Practice* von Wenger (1998). Mittels dieser Theorie wurden die Austauschprozesse innerhalb einer Entwicklungsgemeinschaft betrachtet. Dabei wurde die Kommunikation auf der Mailingliste des Projektes qualitativ und quantitativ ausgewertet. Die Aktivität der Projektgemeinschaft bei der Softwareentwicklung anhand der eingereichten Beiträge bei der Versionsverwaltung wurde ebenso erfasst. Diese wurden mit wichtigen Schlüsselereignissen für die Entwicklungsgemeinschaften in Beziehung gesetzt, die aufgrund der inhaltlichen Beitragsanalyse auf der Mailingliste nachvollzogen werden konnten.

Mit diesem Vorgehen konnte die Wirkung der gemeinschaftlichen Praktiken der Entwicklungsgemeinschaft auf das Engagement und den Kompetenzaufbau einzelner Entwickler:innen nur indirekt gedeutet werden. Aus diesem Grund wurden zusätzlich leitfa-

dengestützte Interviews mit Entwickler:innen von verschiedenen FLOSS-Projekten durchgeführt. Außerdem bilden sie den Datenkorpus, um Interessen, Situiertheit, die Wirkung gemeinschaftlicher Austauschprozesse und den individuellen Kompetenzaufbau aufgrund eines Engagements in FLOSS-Entwicklungsgesellschaften zu beschreiben. Weiterhin lieferten sie die Grundlagen für eine Typologisierung von FLOSS-Entwickler:innen. Die Typologie aus vier Typen zeigt charakteristische Merkmalskombinationen, an den denen die komplexe Zusammenhänge dieser Rahmenbedingungen des informellen Lernens demonstriert werden können. Frühere Typologien von FLOSS-Entwickler:innen wie Ghosh et al. (2002) arbeiteten dagegen nur mit Informationen zu Beweggründen für eine FLOSS-Engagement, die dem quantitativen Paradigma zuzuordnen sind. Die in dieser Arbeit vorgestellten Typen betrachten dagegen die individuelle Entwicklungsarbeit als ein multidimensionales Zusammenspiel von unterschiedlichen Einflüssen des situierten, selbstgesteurten Lernens in kooperativ und kollaborativ agierenden Praxisgemeinschaften. Dies ermöglicht neue Sichtweisen auf informelles Lernen in internetbasierten Lernnetzwerken.

11.2 Kompetenzaufbau durch ein FLOSS-Engagement

Wie bereits im Kapitel 8 beim Lernen durch die Entwicklungsarbeit gezeigt wurde, ist das Engagement in der FLOSS-Entwicklung hauptsächlich im Bereich des informellen Lernens zu verorten. In Einzelfällen wurden auch selbstgesteuerte Lernprozesse in non-formalen Kontexten gefunden. Damit bestätigt die vorliegende Arbeit die These, die Barcomb et al. (2015) in ihrer quantitativ angelegten Studie überprüft haben und ein Kombination aus informellen und non-formalen Lernstrategien bei der Entwicklungsarbeit beschrieben. Trotz solcher Studien über FLOSS-Entwickler:innen war der Aufbau von Kompetenzen in diesem Bereich des informellen Lernens bislang nur oberflächlich erforscht.

Dies gilt nicht nur für den Bereich der FLOSS-Entwicklung: Der gesamten Bereich des informellen Lernens durch gemeinnützige Tätigkeit und dessen Wirkung auf Akteur:innen in diesem Bereich ist wissenschaftlich kaum erforscht. Reinders weist darauf hin, dass sich der Großteil der Studien auf diesem Gebiet auf Kinder und Jugendliche konzentrieren. Zusätzlich hat die empirische Ehrenamtsforschung einen starken Einfluss vom *service learning* der USA, wo ehrenamtliches Engagement Bestandteil des Schulunterrichtes ist und deswegen allenfalls als non-formales Lernen eingestuft werden muss (vgl. Reinders 2018, S. 440f).

Der wissenschaftliche Ertrag dieser Arbeit füllt diese Leerstelle. Erstmals lassen sich die Prozesse des informellen Lernens von Erwachsenen und deren Wirkung mit Blick auf den Kompetenzaufbau einschließlich ihrer Interdependenzen beschreiben. In vorherigen Studien wurden diese Bereiche getrennt voneinander betrachtet und es wurde hypothesenprüfend vorgegangen. Mit dem hier gewählten Ansatz einer qualitativen Studie als theorieentdeckende Forschung, konnte der Kompetenzaufbau mit der notwendigen Komplexität aus individuellem Lernen, eingebettet in einem sozialen Kontext, dargestellt wer-

den. Zum einen wurde gezeigt, dass der Kompetenzaufbau in den Bereichen Sach- und Sozialkompetenz sehr viel differenzierter ist als zuvor bekannt war und zum anderen, dass es auch in den Kompetenzbereichen Selbst- und Sprachkompetenz zu einem umfassenden Kompetenzaufbau kommen kann. Dabei zeigte sich, dass der Kompetenzaufbau zwischen den Entwickler:innen durchaus stark voneinander abweichen kann. Die bisherigen Studien auf diesem Gebiet konnten für dieses Phänomen keine befriedigenden Antworten finden. Aus diesem Grund wurde das Forschungsdesign vorliegender Arbeit auf die Beantwortung dieser Fragen ausgerichtet. Mittels der narrativen Elemente der Interviews konnten die Entwickler:innen die Rahmenbedingungen ihres Kompetenzaufbaus explizieren. Auf dieser Grundlage sind vier verschiedene Typen von Entwickler:innen identifiziert worden: Maintainer:innen, professionelle Programmierer:innen, Gemeinschaftsorientierte und soziale Unterstützer:innen. Sie unterscheiden sich in Bezug auf ihre Interessen und ihre Rollen in der Praxisgemeinschaft. Dies schlägt sich in einem typspezifischen Aufbau unterschiedlicher Kompetenzen nieder. Auch die die bevorzugten Lösungsstrategien beim problemorientierten Lernen variieren innerhalb der Typologie.

11.3 Beitrag zu einer Theorie des informelles Lernens von Erwachsenen in internetbasierten Lernnetzwerken

Angesichts der langen Tradition von netzwerkbasierten FLOSS-Praxisgemeinschaften, die bis in die 1970er[174] Jahre zurückreicht, ist es verwunderlich, dass diese Form der ältesten partizipativen Medienkultur im Internet weder im deutschsprachigen Raum noch darüber hinaus bildungswissenschaftlich ausgiebig beforscht wurde. Das Strohfeuer um *Massive Open Online Courses* (MOOCs), dass als digitales Pendant zum universitären Lehr-Lern-Alltag betrachtet werden kann (vgl. Iske 2018, S. 523), weicht zwar mittlerweile einer realistischen Erwartung an Lernmanagementsysteme, jedoch hat das informelle Lernen als E-Learning weiterhin einen schweren Stand. Das erwachende Forschungsinteresse an internetgestützten, informellen Lernnetzwerken mit hohem Grad an Selbststeuerung wie beispielsweise innerhalb sozialer Netzwerke, Wikipedia oder anderen themenspezifischen Community-Plattformen weist zwar schon in die richtige Richtung, jedoch scheint die Forschung über informelles Lernen Erwachsener in selbstorganisierten Lernnetzwerken noch in den Kinderschuhen zu stecken.[175]

Die vorliegende Forschungsarbeit gibt erstmals einen tiefen Einblick über selbstgesteuertes, informelles Lernen Erwachsener innerhalb informeller, internetbasierter Lernnetzwerke. Gemeinschaftliche FLOSS-Entwicklung kann mit Bezug auf das didaktische

[174] Bereits im ARPANET, dem Vorläufer des Internets, herrschte schon eine Schenkkultur von Software in Form von Quellcodes (vgl. Levy 2010, S. 138). Den Begriff von proprietärer Software gab es zu diesem Zeitpunkt noch nicht. Spätestens mit dem Beginn des GNU-Projekts rief Stallman andere Programmierer:innen dazu auf, sich an der gemeinschaftlichen Softwareentwicklung zu beteiligen (vgl. Stallman 2015, S. 27).

[175] Oft werden solche Lernnetzwerke auch unter dem Begriff *Social Media* oder *Web 2.0* zusammengefasst. Ein Überblick liefert Rohs (2013).

Modell als *connectivist MOOC*[176] gesehen werden. Hier liegt der Schwerpunkt auf der Autonomie der Lernenden: Sie setzen sich ihre eigenen Lernziele, entscheiden über den zeitlichen Umfang und bestimmten selbst, welche Informationsressourcen und Werkzeuge sie nutzen (vgl. Haug & Weidekind 2013, S. 165). Wie der Name bereits zeigt, orientiert sich das Konzept an der Lerntheorie des Konnektivismus. Für sich genommen kann der Konnektivismus noch nicht als eigenständige Lerntheorie betrachtet werden, da er sich im Kern auf den Sozialkonstruktivismus beruft und auch innerhalb seiner Theorie Widersprüche aufweist. Auch Downes zeigt in seinen Ausführungen, dass der Schwerpunkt auf dem methodischen und didaktischen Konzept des Lernens durch soziale Interaktion liegt (vgl. Downes 2013, Slide 8). Dagegen ist die Theorie der *Communities of Practice* ein sehr viel elaborierterer Ansatz zur Beschreibung von kooperativem und kollaborativem Lernen. Was ihm jedoch fehlt ist einerseits die Übertragung in die spezifische Sphäre des E-Learnings mit seinen Chancen und Einschränkungen für gemeinschaftliches Lernen in internetbasierten Lernnetzwerken und andererseits die individuellen Einflussfaktoren auf Lernen in sozialen Kontexten.

Formen selbstgesteuerten Lernens

Dohmen hat zu Beginn des Forschungsinteresses am informellen Lernen auf dessen Zweckorientierung hingewiesen. Informelles Lernen ist ein Mittel, um einen bestimmten Zweck zu erreichen: Beispielsweise als Form der Lebensbewältigung im Kontext des lebenslangen Lernens oder einer spezifischen Situationsanforderung (vgl. Dohmen 2001, S. 19). In dieser Forschungsarbeit wurde das informelle Lernen mit Fokus auf seine Problemorientierung untersucht. Das selbstgesteuerte Lernen der Entwickler:innen zielte auf eine Problemlösung. Es handelt sich dabei um intentionales Lernen, also ein zielgerichtetes, bewusstes Lernen. Nach der Problemwahrnehmung wurde sich Kontextwissen angeeignet, um das Problem zu verstehen und anschließend eine Lösungsstrategie gewählt. Durch Versuch und Irrtum wurde die Problemlösung getestet und gegebenenfalls nach einer nicht zufriedenstellenden Problemlösung der Problemlösungszyklus in veränderter Form erneut angewendet. Dennoch kommt es zu inzidentellem Lernen, also beiläufigen Lernprozessen. Trotz der maßgebenden Selbststeuerung sind nicht intendierte Lernprozesse Teil des Lernens. Das sind beispielsweise der Aufbau verschiedener Sozialkompetenzen durch die gemeinschaftlichen Austauschprozesse, weil die solitäre Problemlösung nicht den gewünschten Erfolg hatte. Kommunikationsakte werden mit dem Ziel der Informationsrecherche initiiert, dennoch bauen die Entwickler:innen durch Erfahrungslernen Kompetenzen in der Pflege sozialer Beziehungen auf. Inzidentelles Lernen kann demnach nicht von intendiertem Lernen durch Selbststeuerung getrennt betrachtet werden. Beide können zusammen auftreten.

[176] Bei MOOC's wird zwischen zwei Formen unterschieden: extended MOOC's entsprechen dem methodischen Format einer Online-Vorlesung, die um weitere Elemente nach den technischen Möglichkeiten der MOOC-Software ergänzt werden kann. Oftmals wird bei der allgemeinen Bezeichung MOOC von einem *extended* MOOC beziehungsweise xMOOC ausgegangen, da diese zahlenmäßig dominieren (vgl. Iske 2018, S. 523).

Marsick und Watkins weisen darauf hin, dass inzidentelles Lernen ein fester Bestandteil vom Lernen durch Versuch und Irrtum ist. Ein unerwarteter Fehler oder ein nicht antizipiertes Ereignis stellen eine Lernmöglichkeit dar. Es muss aber nicht zwangsläufig zum Lernen kommen: Erst die Reflexion über den Prozess entscheidet darüber, ob gelernt wird oder ob vorherige Einstellungen und Wissensbestände unverändert beibehalten werden (vgl. Marsick & Watkins 2015, S. 7ff). In dieser Forschungsarbeit wurde gezeigt, dass selbstgesteuertes Lernen nach einer Problemwahrnehmung sowohl zu intentionalen wie auch inzidentellen Lernprozessen führt, da Lösungsstrategien durch Versuch und Irrtum häufig anzutreffen sind. Beides trägt zu einem Kompetenzaufbau bei, auch wenn dieser teilweise von den Entwickler:innen nicht explizit wahrgenommen wird.

Die Bedeutung von Sinn und Interesse

Eine der leitenden Forschungsfragen dieser Arbeit ist die Frage nach den Ursachen für das Engagement in FLOSS-Praxisgemeinschaften. Es zeigte sich, dass die Entwickler:innen sehr ausführlich ihre Beweggründe explizieren können. Sie beschreiben verschiedene Interessen, die sie durch ihr Engagement verfolgen. Auch deren Veränderung im Zuge eines mehrjährigen Engagements in Praxisgemeinschaften können dargestellt werden. Dies zeigt, dass die Bewertung des Engagements anhand der gemachten Erfahrungen elementar für die individuellen Sinnstrukturen sind. Von ihnen hängt ab, ob die eigene Tätigkeit als sinnvoll erlebt wird. Diese Einschätzung wiederum beeinflusst, ob und wie in Zukunft das Engagement fortgesetzt wird. Diesbezüglich konnte gezeigt werden, dass individuelle Interessen und gemeinschaftsorientierte Interesse eng miteinander verbunden sind.

Bei formal organisierten Lernarrangements erfolgt die Sinnfindung auch über die Validierung von Lernerfahrungen. Diese erfolgt über die Verleihung von Bildungsabschlüssen oder Qualifikationen. Dies ist beim informellen Lernen nicht der Fall, dennoch sind die Entwickler:innen bestrebt, ihr Handeln wert- und zweckrational zu begründen. Die Anerkennung von informell aufgebauten Kompetenzen ist allerdings in diesem Bereich eine Grauzone, weil ein öffentliches FLOSS-Engagement als Reputation durchaus Einfluss auf die Employability in einschlägigen Branchen hat.

Anerkennung ist aber auch in Bezug für informelles Lernen innerhalb von betrieblichen Kontexten relevant. Hier nimmt die Bedeutung von internetgestützten Lernnetzwerken durch einen Wandel von Arbeits- und Organisationsformen parallel zu formalen Weiterbildungen stetig zu (vgl. Dehnbostel 2018, S. 435). Sinn und Interesse sind fundamental für das Verständnis von selbstgesteuertem Lernen. Dies gilt sowohl für das Engagement in informellen Lernnetzwerken als auch für das individuelle Lernen bei der Anwendung von solitären Problemlösungsstrategien.

Notwendige Erweiterung der *Communities of Practice*

Für das gemeinschaftliche Lernen auf der Grundlage von sozialkonstruktivistischen Lerntheorien hat sich die Theorie der *Communities of Practice* zur Beschreibung von ko-

operativem und kollaborativem Lernen großteils als geeignetes Analyseraster erwiesen: Die Theorie hat ihre Stärke bei der Beschreibung der gemeinschaftlichen Praktiken und deren Resultate. Dies gilt aber nur für die Außenbeobachtung des gemeinsamen Repertoires. Die individuellen Auswirkungen der Austauschprozesse auf einzelne Mitglieder der Praxisgemeinschaft werden in dieser Theorie nur am Rande betrachtet:

> „In addition, these elements [domain, community, and practice] provide a means to understand the different ways in which participation is meaningful to members — some may be more interested in the community than in the practice aspect, for instance." (Wenger et al. 2002, S. 40f)

Was genau eine bedeutungsvolle Partizipation für ein einzelnes Mitglied ausmacht, wird nicht weiter ausgeführt. Die Lerntheorie der Praxisgemeinschaft basiert auf der Teilhabe an soziokulturellen Austauschprozessen. Lernen ist dahingehend eine Änderung der sozialen Praxis (vgl. Lave & Wenger 1991, S. 35). In diesem Sinne wird Lernen nicht als individueller Prozess verstanden, sondern als Änderung von Praktiken in einer Gemeinschaft. Es ist damit eine relationale Änderung des Denkens und Handeln aller Mitglieder der Gemeinschaft.

Dieser Argumentation folgend fehlt der Theorie der Communities of Practice ein Konzept des individuellen Lernens. Der individuelle Anteil am Lernprozess ist die Art und Weise, wie eine Person an den Austauschprozessen teilnimmt. Lernen setzt also die Teilhabe an einer sozialen Praxis voraus. Ein Lernen ohne Praxis kann es demnach nicht geben. Unbestreitbar ist, das der Mensch als soziales Wesen auf den Austausch mit anderen Menschen für die physische und psychische Entwicklung angewiesen ist. Diese Ausklammerung individueller kognitiver Prozesse war Lave und Wenger bewusst: Sie wollten gezielt einen Gegenpol zu Lerntheorien setzen, die sich rein auf ein Individuum stützen (vgl. ebd. S. 52). Dennoch haftet dadurch dieser Lerntheorie ein Makel an. Die Frage, was Individuen zu einer Partizipation anregt, ist für eine Lerntheorie durch soziale Praxis ebenso wichtig wie die Praxis selbst. Aus diesem Grund wird der Bedeutung von Interesse und Sinn in dieser Arbeit ein besonderer Stellenwert zugewiesen. Neben weiteren Faktoren habe sie einen entscheidenden Einfluss darauf, wie und ob an einer Praxisgemeinschaft partizipiert wird.

Die Lerntheorie ist blind für individuelle Handlungen einer Person, die gegen die Strukturelemente einer Praxisgemeinschaft gerichtet sind. Als lernhemmende Strukturen und Prozesse werde lediglich mangelnder Zugang und fehlende Transparenz zu den Strukturen und Interaktionen in der Praxisgemeinschaft beschrieben (vgl. ebd. S. 101). Es scheint also nur zwei Modi von Lernkontexten zu geben: Die Zunahme an partizipativen Handlungen der Praxisgemeinschaft in Form der *legitimen peripheren Partizipation* und die Stagnation der Austauschprozesse zwischen Individuum und Gemeinschaft. Ein Lernen, was sich im Aufbau einer kritischen Selbstkompetenz gegenüber dem Druck der eigenen Gruppe manifestiert (vgl. Roth 1971, S. 551), scheint es nicht zu geben. Eine kritische Urteils- und Handlungsfähigkeit braucht jedoch ein Fundament internalisierter

Werte, Normen und Moralvorstellungen, um unter bestimmten Bedingungen auch gegen die Interessen der Gruppe handeln zu können.

Um das Lernen in Praxisgemeinschaften beschreiben zu können, bedarf es der Dimensionen des individuellen Lernens wie Situiertheit und der Betrachtung der jeweiligen Interessen und Sinnstrukturen. Ebenfalls kommt eine Lerntheorie, die sich nur auf das Individuum stützt, nicht ohne ein Konzept des Lernens in seinem sozialen Kontext aus.

Besonderheiten raum- und zeitdistanter Telekommunikation

Die geographische Verteilung der Mitglieder einer Praxisgemeinschaft birgt zusätzliche Schwierigkeiten für internetbasierte Lernnetzwerke. Neben der größeren kulturellen Heterogenität der Mitglieder tritt zusätzlich die Einschränkung von unterschiedlichen Zeitzonen aufgrund der potenziell globalen Ausdehnung solcher FLOSS-Gemeinschaften hervor. Dadurch sind die Möglichkeiten zum kommunikativen Austausch eingeschränkt oder bedürfen hohen Anstrengungen seitens der einzelnen Mitglieder. Die räumliche Ausdehnung wirkt sich hauptsächlich auf die synchrone Kommunikation aus. Dies hat nachteilige Auswirkung auf die gemeinschaftliche Praxis. Langfristig konstruktiv interagierende Praxisgemeinschaften entwickeln eine *craft intimacy*, ein Konzept, das Wenger et al. wie folgt beschreiben:

> „Community members get to know each other's style and approach to technical problems. In conversation and joint projects, they discover their strengths and weaknesses and come to appreciate others' contributions, energy, interest, perspectives, and individual styles." (Wenger et al. 2002, S. 98)

Craft intimacy ist das dass zentrale Konzept, um eine Vielfalt an Austauschprozessen innerhalb einer Praxisgemeinschaft zu stimulieren. Ansonsten besteht die Gefahr, dass sich ein sternförmiges Netzwerkmodell der Praxisgemeinschaften herausbildet: Im Zentrum steht die Person, die am meisten kommunikativ in Erscheinung tritt. Im Regelfall ist dies der:die Projektmaintainer:in, mit einer Vielzahl an bilateralen Kommunikationsbeziehungen mit einzelnen Entwickler:innen. Daraus folgt eine problemorientierte, lernwirksame Kommunikation als dyadische Beziehung zwischen Projektmaintainer:in und Entwickler:in. Informelles Lernen erfolgt in solch einem Kontext als *peer education*[177], welches oftmals unter Formen des kooperativen und kollaborativen Lernens subsumiert wird (vgl. Konrad 2014, S. 80).

Idealerweise sollte die lernwirksame Kommunikation auch zwischen einzelnen Entwickler:innen erfolgen, um eine kommunikative Überlastung des Netzwerkzentrums zu vermeiden. Dies lässt sich durch die Vorteile asynchroner Kommunikation in offenen,

[177] Lernen durch *peers* wird in der Forschung oft mit dem Bereich des Kinder- und Jugendalters in Verbindung gebracht. *Peer-education* ist aber auch im Bereich der Erwachsenenbildung ein fester Bestandteil informeller Lernprozesse. Die Definition von *peers* über das ähnliche Alter verschwimmt hier und die Zugehörigkeit zu bestimmten Status- oder Interessengruppen tritt in den Vordergrund. Etabliert hat sich das Konzept beispielsweise in Form von Selbsthilfegruppen.

persistenten Kommunikationskanälen erreichen. Gruppenweite persistente Kommunikationskanäle bieten dahingehend zwei Vorteile: Erstens bieten sie die Möglichkeiten zur Stimulation der *craft intimacy* und zweitens dienen sie dem Aufbau einer gemeinschaftlichen Informationsressource. Lernwirksame Kommunikation konnte anhand des persistenten Kanals der öffentlichen Projekt-Mailingliste nur in begrenztem Maß gefunden werden (vgl. Kapitel 6.5). Daraus lässt sich jedoch nicht schließen, dass es diese nicht gab. Anhand der solitären Problemlösungsstrategie wurde demonstriert, wie asynchrone Kommunikation über Problemlösungen in diversen Praxisgemeinschaften von Entwickler:innen als Informationsquelle für die Lösung eigener Probleme genutzt wird. Dieses Phänomen ist auch als *lurking* im Netzjargon bekannt. Als Lurker werden Personen bezeichnet, die passiv die Beiträge in Kommunikationskanälen verfolgen, jedoch selbst nicht aktiv mit eigenen Beiträgen in Erscheinung treten. Diese Beschreibung kann eine negative Konnotation aufweisen, wenn dies mit dem Phänomen des Trittbrettfahrens assoziiert wird.

Mit dem Aspekt der Problemlösung und des Wissenstransfers beschreibt Kahnwald eine andere Perspektive dieses Phänomens: *Lurking* schont die Ressourcen einer Praxisgemeinschaft, da Fragen nicht mehrfach gestellt werden müssen (vgl. Kahnwald 2013, S. 38). Weiterhin gilt *Lurking* in begrenztem Umfang auch als akzeptierte Praxis, um sich vor der eigenen aktiven Partizipation über die gruppenweiten kulturellen Normen zu informieren (vgl. Stegbauer & Rausch 2001, S. 49). Demnach ist es auch ein vorbereitender Prozess für neue Mitglieder zum Kennenlernen der Strukturen und Mitglieder einer bereits etablierten Praxisgemeinschaft. Archivierte E-Mail- oder Forenbeiträge stellen somit eine Form zeit- und raumdistanter lernwirksamer Kommunikation dar. Darüber hinaus erleichtert und stimuliert sie auch die legitime periphere Partizipation durch Hinweise über die *craft intimacy* von existierenden Praxisgemeinschaften.

11.4 Konsequenzen für die Gestaltung internetbasierter Lernnetzwerke

Aus den praktischen Erfahrungen der befragten Entwickler:innen in den FLOSS-Gemeinschaften lassen sich grundlegende Strukturelemente für die Gestaltung dezentraler Praxisgemeinschaften ableiten. Sie zielen auf zwei wichtige Komponenten ab: Erstens die Rahmenbedingungen zu schaffen, die lernförderliche Kommunikation innerhalb der Gemeinschaft ermöglichen und zweitens, den Mitgliedern Zugang zu den Informationen über Praktiken, Werkzeuge und Zielbestimmungen der Gemeinschaft gewähren, um ein erstes Engagement neuer Mitglieder zu ermöglichen.

Verschiedene Interessen der Mitglieder berücksichtigen

Die Betrachtung der Interessen von Entwickler:innen zeigt in dieser Arbeit ein breites Spektrum von unterschiedlichen Beweggründen für ein FLOSS-Engagement auf. Die Typisierung zeigt zwei Pole: Entweder die Entwickler:innen sind hauptsächlich an dem sozialen Beziehungsgeflecht der Entwicklungsgemeinschaft interessiert oder sie verfol-

gen in erster Linie ihre Interessen bezogen auf eine bestimmte Technologie oder Anwendungsfeld. Auslöser für ein Engagement ist jedoch meistens die eigene Betroffenheit durch eine fehlerhafte funktionierende oder nicht existente FLOSS.

Auf informelle Lernnetzwerke bezogen lässt sich folgendes feststellen: Die Partizipation an informellen Lernnetzwerken erfolgt problemorientiert mit einem konkreten Bezug zur eigenen Lebenswelt. Das Engagement ist, zumindest zu Beginn, auf Lebensbewältigung in klar definierten Anwendungsbereichen hin ausgerichtet. Mit zunehmender Erfahrung in der Praxisgemeinschaft treten auch andere Beweggründe hervor, die mit dem ursprünglichen Auslöser nur noch lose in Verbindung stehen. Nichtsdestotrotz ist die Möglichkeit, eigene Interessen zu verfolgen, für die Sinnfindung essentiell und dadurch eine wichtige Voraussetzung für das individuelle Lernen. Praxisgemeinschaften mit kontinuierlichen Austauschprozessen und dem Vermögen, auch neue Personen zur Partizipation im Netzwerk zu ermutigen, adressieren ein breites Spektrum der möglichen Interessen ihrer Mitglieder.

Kommunikation fördern

Kommunikation ist die Grundvoraussetzung eines Lernnetzwerkes. Sie ermöglicht nicht nur den Transfer von Wissen zwischen den Mitgliedern, sie stellt auch selbst in ihrer öffentlichen, persistenten Form eine Informationsressource für Nicht-Mitglieder dar und dokumentiert die Praktiken der Gemeinschaft. Dafür braucht es einen gemeinsamen Kommunikationskanal, der allen Teilnehmenden zugänglich ist und sich an alle anderen Teilnehmenden in ihrer Gesamtheit richtet. Daneben sollten auch Kommunikationskanäle für den vertraulichen Austausch zwischen einzelnen Teilnehmenden existieren.

Ein wertschätzender und respektvoller Umgang unter den Mitgliedern ist für den Beziehungsaufbau und die Gruppenidentität unabdingbar. Auch ein Austausch darüber, wer im Lernnetzwerk welche Kenntnisse hat und was einzelne Mitglieder leisten können, erleichtert die Koordination des kooperativen und kollaborativen Handelns. Insbesondere sollten Konflikte im Lernnetzwerk kommunikativ im respektvollen Austausch gelöst werden. Ungelöste Konflikte können die gesamte Gemeinschaft lähmen oder gar zur Abwanderung Einzelner oder ganzer Teile der Gemeinschaft führen.

Strukturen für den Beziehungsaufbau schaffen

Physische Treffen mit Entwickler:innen, mit denen zuvor nur internetbasierte Kommunikation erfolgte, stellten für viele Entwickler:innen prägende Schlüsselerlebnisse dar, die stark zu einer gemeinsamen Gruppenidentität beitrugen. Bei räumlich weit verzweigten Netzwerken könnten dies Treffen von einzelnen, regional benachbarten Entwickler:innengruppen sein. Möglich sind Formate wie Konferenzen, Arbeitstreffen oder rein informelle Begegnungen, die hauptsächlich dem gegenseitigen Kennenlernen dienen. Sind physische Treffen nicht möglich, sind auch internetbasierte Kommunikationskanäle denkbar, die anderen Entwickler:innen eine Vorstellung über die Personen hinter den Profilnamen oder E-Mail-Adressen ermöglichen, wie beispielsweise Videokonferenzformate.

Niedrigschwelliger arbeiten Praxisgemeinschaften, die auf der Projektwebseite engagierte Personen mit ihren Tätigkeitsbereichen und Kontaktmöglichkeiten vorstellen. Dies ermöglicht neuen Mitgliedern, gezielt Fragen an die Personen zu richten, in deren Verantwortungsbereich die betreffenden Frage fällt.

Synchrone Kommunikationskanäle ermöglichen

Sowohl bei der selektiven als auch bei der kollaborativen Lösungsstrategie und auch bei der Entscheidungsfindung in Bezug auf Entwicklungsziele wurde von vielen Entwickler:innen wiederholt die Bedeutung eines synchronen Kommunikationskanals zu einzelnen Entwickler:innen betont. Dieser dient der Klärung von Missverständnissen, die aufgrund der eingeschränkten Ausdrucksmöglichkeiten durch raum- und zeitdistanter Telekommunikation ein latentes Problem der Schriftkommunikation darstellt. Außerdem ermöglicht er eine schnelle Rückmeldung bei fehlenden Information. Die befragten Entwickler:innen nutzten zu diesem Zweck verschiedenste Chat-Dienste, Instant-Messenger oder Telefonate.

Periphere Partizipation erleichtern

Für den Fortbestand einer Praxisgemeinschaft ist die Integration neuer Mitglieder überlebenswichtig. Ausscheidende Mitglieder müssen ersetzt werden und der Prozess vom ersten Engagement bis zur vollen Partizipationfähigkeit benötigt Zeit. Interessierte müssen sich erst in die Praktiken der Praxisgemeinschaft einfühlen, sich das notwendige Wissen für eigene Problemlösungen aneignen und das Vertrauen haben, dass ihre Arbeit von der Gemeinschaft geschätzt und respektiert wird. Verschiedene Maßnahmen sind dafür geeignet, potentiellen Mitgliedern die ersten Austauschprozesse mit der Entwicklungsgemeinschaft zu erleichtern:

- Aufgabensammlung von benötigten Maßnahmen für den Entwicklungsfortschritt, die auch von neuen oder unerfahrenen Mitgliedern erfolgreich bearbeitet werden können - hier bietet sich eine Kategorisierung der zu lösenden Probleme nach Grad der Schwierigkeit an

- Dokumentation der Diskussions- und Arbeitsweise der Entwicklungsgemeinschaft: Anhand öffentlich einsehbarer Dokumente können neue Mitglieder eine Anleitung, welche Anforderungen die Entwicklungsgemeinschaft an einen Beitrag stellt und wo benötigte Informationen gefunden werden können beziehungsweise auf welchen Kommunikationskanälen Änderungswünsche kommuniziert werden können oder nach Hilfe gefragt werden kann

- Funktionierendes Beitragsmanagement: Anfragen oder eingereichte Beiträge zum Projektfortschritt müssen zeitnah beantwortet, bearbeitet oder in das gemeinsame Repertoire aufgenommen werden können

Transparenz über das gemeinsame Repertoire schaffen

Das gemeinsame Repertoire umfasst weitaus mehr als die Sammlung der Beiträge zur Annäherung an das Ziels, zu dessen Erreichung sich die Praxisgemeinschaft ursprünglich gegründet hat. Auch hinter der Art und Weise, wie dieses Ziel erreicht werden soll, verbergen sich Wertorientierungen und politische Vorstellungen. Darüber sollte sich eine Praxisgemeinschaft bewusst sein und diese Grundlagen der Zusammenarbeit im gemeinsamen Austausch erstellen und für alle interessierten Personen zugänglich machen. Dies beinhaltet auch das konkrete Vorgehen bei der Entscheidungsfindung und den Arbeitsabläufen.

Darüber hinaus ist auch die Transparenz bezüglich des Projektfortschritts wichtig. Potentielle Mitglieder benötigen Informationen darüber, wie neue Beiträge in das Gesamtprojekt integriert werden und ob dieser Prozess von außen nachvollzogen werden kann. Einige Entwickler:innen legen auch großen Wert auf die Zuordnung individueller Anteile einzelner Personen am gesamten Projektfortschritt. Insbesondere der Typus der professionellen Programmierer:innen nutzen dies für ihre Reputation.

11.5 Capabilities und Lebensqualität bei der FLOSS-Entwicklung aus bildungstheoretischer Sicht

Im Theorieteil der Arbeit wurde im Kapitel 3.1.4 das Potential des Fähigkeitenansatz von Nussbaum beschrieben. Als Zielbestimmung pädagogischen Handelns bietet dieser Ansatz eine Perspektive zur Weiterentwicklung einer kritischen Bildungstheorie. Bildung wird dabei als eine Befähigung verstanden, die ein *gutes Leben* ermöglicht. Dafür braucht es allerdings eine klare Vorstellung, was als gutes Leben gilt und was es zu dessen Erreichung bedarf.

Subjektives Wohlbefinden ist als Evaluationskriterium dafür nicht geeignet, denn subjektive Wertmaßstäbe sind formbar. Sie werden durch Sozialisation und Enkulturation erworben und passen sich den individuellen Lebensbedingungen an (vgl. Otto et al. 2010, S. 152f). Lernen ist in diesem Sinne eine Bewältigungsstrategie, um sich im Alltag anzupassen und dadurch eine subjektive Verbesserung an aktuelle Lebensbedingungen zu gewährleisten. Das subjektive Wohlbefinden verbessert sich dann, ohne dass die begrenzenden Faktoren für den eigenen Möglichkeitsraum kritisch hinterfragt werden können.

Dagegen setzt Nussbaum beim Fähigkeitenansatz auf eine Kombination aus externen Bedingungen und internen Fähigkeiten. Ziegler fasst dies mit Blick auf die Capabilities wie folgt zusammen: Die Befähigung zu einem guten Leben bedeutet demnach, „[...] das spezifische Zusammenspiel der Eigenschaften, Fähigkeiten und Bedürfnisse von Subjekten mit objektiven (sozialen und politischen) Gegebenheiten und Möglichkeitsräumen gegenüber den institutionellen und materiellen Bedingungen“ (Ziegler 2018, S. 80) zu berücksichtigen. Capabilities als Entfaltungsmöglichkeiten erweitern damit die subjektiv verzerrte Dimension des Wohlbefindens mit den objektiven Gegebenheiten der Lebensbedingungen. Befähigung lässt sich nicht auf individuelle Eigenschaften oder Kompeten-

zen reduzieren, sondern betrifft die Entfaltungsmöglichkeiten und Lebensaussichten, die Menschen in realen gesellschaftlichen Verhältnissen realisieren können (vgl. ebd., S. 80). Dafür sind materielle Voraussetzungen eine notwendige, aber keine hinreichende Bedingung. Aus diesem Grund laufen Hilfssysteme, die auf die ausschließliche Verteilung von materiellen Ressourcen setzen, ins Leere. Wie genau der Fähigkeitenansatz von Nussbaum ein Beitrag zur Weiterentwicklung einer kritischen Bildungstheorie in Bezug auf informelles Lernen darstellen kann, beschreibt Ziegler wie folgt:

> „Die Stärke des Ansatzes besteht darin, ein klassisches Motiv der emanzipatorischen Bildungstheorie in das Zentrum einer allgemeinen gerechtigkeitstheoretischen Perspektive zu rücken: die Ermöglichung von Autonomie der Lebenspraxis und die Erweiterung der Entfaltungsmöglichkeiten der Subjekte. Sofern es bei dem, was als »informelles Lernen« erforscht wird, nicht nur um die Beschreibung familien- oder peerbezogener Sozialisations- und Wissenserwerbsprozesse geht, sondern um eine bildungtheoretische Fokussierung dieser Prozesse im Verbund mit der gerechtigkeitstheoretischen Idee »to create a community in which people stand in relation of equality ot others« (Anderson 1999, S. 289), scheint der Capabilities Ansatz eine vielversprechende evaluative Heuristik für die Analyse dieser Prozesse zu eröffnen.“ (ebd., S. 83)

Zieglers Kritik zielt auf bildungswissenschaftliche Diskurse, die auf Evaluation von Lernprozessen im Sinne von Vergleichbarkeit setzen, aber die Ursachen von Ungerechtigkeit entweder marginalisieren oder als vermeintlich unveränderbare Konstante nicht thematisieren.

Befähigung durch die Partizipation in FLOSS-Entwicklungsgemeinschaften

Der folgende Abschnitt zeigt, wie aus Sicht einer befähigungsorientierten, evaluativen Heuristik der Ertrag dieser Forschungsarbeit eingeschätzt werden kann. Diesbezüglich ist ein ausschließlicher Fokus auf den Kompetenzaufbau, wie er im Kapitel 9 vorgestellt wurde, nicht ausreichend.

Nussbaum stellt für ihre *starke vage Konzeption des Guten* einen Katalog von Grundfähigkeiten auf, der bereits im Theorieteil der Arbeit in Kapitel 3.1.4 vorgestellt wurde. Bei einigen dieser Grundfähigkeiten kann ein Bezug zu dem Engagement in FLOSS-Entwicklungsgemeinschaften hergestellt werden:

1. Aufbau von Bindungen zu Dingen und Personen:

   Mit Bezug auf die FLOSS-Entwicklung soll das Augenmerk auf die FLOSS selbst als immaterielles Gut und den Praktiken der Gemeinschaft gelenkt werden. FLOSS-Entwickler:innen berichten von einer großen Zufriedenheit mit Blick auf die eigene oder die von anderen Mitgliedern der Entwicklungsgemeinschaft erbrachten

Leistungen. Dies wurde sowohl in den E-Mail-Diskussionen durch die verschiedenen Formen der geäußerten Anerkennung, wie auch in den Ausführungen innerhalb der Interviews ausgedrückt. Anerkennung spielt auch indirekt eine Rolle: Beispielsweise durch die Verbreitung der FLOSS als Indikator für die wachsende Zahl an Nutzer:innen. Anerkennung, die insbesondere von erfahrenen FLOSS-Entwickler:innen als Beweggrund für die Fortsetzung des Engagements genannt wird, ist eine Form des positiven Bezugs zu der FLOSS selbst und den gemeinschaftlichen Praktiken der Entwicklungsgemeinschaft, die die FLOSS-Entwicklung ermöglicht hat. Die Bedeutung einer Bindung zu den Praktiken der Entwicklungsgemeinschaft wurde als Teil der Sozialkompetenz in Kapitel 9.2.3 in Bezug auf die Handlungsmacht als Gruppe identifiziert. Aspekte, die diese Grundfähigkeit beeinflussen, zeigen sich demnach durch die Änderung von Interessen und Sinnstrukturen, wie auch durch den Aufbau von Teilkompetenzen im Bereich der Sozialkompetenz.

2. Vorstellung vom Guten haben und Befähigung zur kritischen Reflexion der eigenen Lebensplanung:

   Was genau für jeden Menschen ein gutes Leben ist, ist individuell verschieden. Auch muss zwischen situativen Handlungen und generellen Zielvorstellungen unterschieden werden. Die Idealtypen des sozialen Handelns von Weber unterscheiden zweckrationales, wertrationales, affektuelles und traditionales Handeln (vgl. Weber 1980, S. 12f). Dieser Typologie folgend, wurden indirekt Zielvorstellungen anhand von Wertorientierungen nachgewiesen. Wertrationales Handeln in FLOSS-Projekten spielt für viele Entwickler:innen als Beweggrund für ein FLOSS-Engagement eine wichtige Rolle. Dahinter verbergen sich implizite Vorstellungen über ein gutes Leben beziehungsweise welche Handlungsweisen damit konsistent sind. Dies zeigt sich konkret als sachkompetente Urteilsfähigkeit bezüglich gesellschaftlicher Verhältnisse. Dennoch sollte nicht vernachlässigt werden, dass zweckrationales Handeln den wichtigsten Auslöser für das erste Engagement in einem FLOSS-Projekt darstellt. Es erfolgt oftmals aufgrund einer persönlichen Betroffenheit durch Programmfehler in der genutzten Software. Bezüglich der Reflexion über die eigene Lebensplanung wurde gezeigt, dass die interviewten Entwickler:innen die Grenze zwischen FLOSS-Entwicklung als reines Hobby oder als Lohnarbeit bewusst ziehen. Soll die Entwicklungsarbeit vom Hobby zum Beruf gemacht werden, wird stärker auf die Außenwirkung des eigenen Engagements geachtet. Die Signalwirkung eines Engagements in FLOSS-Projekten als Reputation für potentielle Arbeitgeber:innen ist diesen Entwickler:innen durchaus bewusst.

3. Verschiedene Formen von sozialen Bindungen eingehen können:

   Soziale Bindungen aufbauen und unterhalten zu können ist die Schlüsselkompetenz bei der gemeinschaftlichen FLOSS-Entwicklung. Als Grundvoraussetzung braucht

es Sprachkompetenz bezüglich der gemeinsam genutzten Sprache. Die Stärke der sozialen Beziehung kann verschieden sein. Von kurzweiligen, anonymen Einzelkontakten über asynchrone Kommunikationskanäle der FLOSS-Portale bis hin zu langjährigen freundschaftlichen Beziehungen zu Gleichgesinnten mit regelmäßigen physischen Treffen sind alle Formen von sozialen Bindungen in FLOSS-Projekten anzutreffen. Dementsprechend ist auch die Sozialkompetenz der Kompetenzbereich, dem in dieser Forschungsarbeit aufgrund seines Stellenwertes bei den Interviews eine besondere Beachtung geschenkt wurde. Dies zeigt sich aber nicht nur im Aufbau sozialer Kompetenzen. Im Rahmen der Typologisierung der Entwickler:innen zeigt sich, dass soziale Bindungen ein wichtiger Beweggrund für ein FLOSS-Engagement ist. Dies war einer der Gründe, weshalb dieses Interessenkonstrukt auch als Unterscheidungsmerkmal für die Typenbildung genutzt wurde. Zwei der vier konstruierten Typen leiten in unterschiedlichen Weisen ihre Sinnstrukturen aus den sozialen Bindungen und den daraus resultierenden Austauschprozessen ab.

4. Freude an erholsamen Tätigkeiten:

   Die Freude am FLOSS-Engagement ist wohl eines der wichtigsten Beweggründe. Diese zeigt sich in unterschiedlichen Facetten. Für einige Entwickler:innen ist es die Freude am entdeckenden beziehungsweise am problemzentrierten Lernen in der Auseinandersetzung mit einer bestimmten Technologie oder ein thematisches Interesse. Andere hingegen heben die Bedeutung der sozialen Austauschprozesse mit Nutzer:innen und anderen Entwickler:innen hervor. Da die Entwicklungsarbeit in einem FLOSS-Projekt für die Mehrheit der Entwickler:innen ein gemeinschaftsorientiertes Ehrenamt darstellt, ist die Fähigkeit, dies als freudvolle oder erholsame Tätigkeit zu erleben, elementar. Dadurch ist sie Bestandteil der eigenen Sinnkonstruktion. Auch wenn FLOSS-Entwicklung für viele als Hobby beschrieben wird, so ist es auch ein forderndes Hobby. Es erfordert Zeit, die als knappe Ressource gegen anderen Anforderungen im Lebensalltag der Entwickler:innen verteidigt werden muss. Mitunter ist die FLOSS-Entwicklung auch nicht erholsam und kann auch frustrierende Aspekte haben, wie dies in Kapitel 8.1.2 anhand von Problemen bei der gemeinschaftlichen FLOSS-Entwicklung aufgezeigt wurde.

5. Das eigene Leben führen:

   Diese Grundfähigkeit erstreckt sich zum einen auf die Selbstkompetenz und zum anderen auf die Sozialkompetenz. Es bezieht sich auf die Teilkompetenzen, die die individuelle Handlungsfähigkeit als autonom agierende Person im gesellschaftlichen Kontext betreffen. Dafür bedarf es Selbstvertrauen in die eigenen Fähigkeiten und eine kritische Urteils- und Handlungsfähigkeit, um eigene und fremde Bedürfnisse und Interessen voneinander zu unterscheiden und ihnen gegebenenfalls nachgehen zu können. Ist die Befriedigung eigener oder gemeinschaftlicher Bedürfnisse nicht aus eigener Kraft leistbar, braucht es zusätzlich soziale Kompetenzen. Ziel ist

es, eigene und fremde Ressourcen koordinieren zu können. Solche Teilkompetenzen sind auch für die kollektiven Aushandlungsprozesse in FLOSS-Gemeinschaften notwendig. Für den Bereich der Sozialkompetenz umfasst dies, dass der eigene Standpunkt gegenüber der Entwicklungsgemeinschaft vertreten werden kann. Im Fall von Konflikten mit anderen bedarf es zusätzlich kompetentes Handeln im Bereich der kommunikativen Lösung von Konflikten. In Kapitel 9.2.1 wurden die sozialen Teilkompetenzen beschrieben, die für die gemeinschaftlichen Entscheidungsfindungen in FLOSS-Gemeinschaften notwendig sind. Damit verbunden sind auch Fähigkeiten im kollektiven Umgang mit Konflikten.

Die Auflistung zeigt, dass es Überscheidungsbereiche zwischen der kompetenzorientierten und der auf Grundfähigkeiten beziehungsweise *Capabilities*-orientierten Evaluation gibt. Dennoch wurde von Nussbaum keine trennscharfe Heuristik vorgelegt, mit der sich einzelne Grundfähigkeiten klar auf die Erfahrungen einzelner Personen durch das Engagement in Praxisgemeinschaften übertragen lassen. Im Folgendem wird ein Versuch beschrieben, den Fähigkeitenansatz in Bezug auf die Ausgestaltung einer Heuristik weiter auszugestalten.

Lebensqualität als Zielbestimmung für ein gutes Leben

Im Theorieteil dieser Arbeit wurde bereits auf das Fehlen einer geeigneten Operationalisierung des Katalogs der Grundfähigkeiten für den bildungswissenschaftlichen Diskurs hingewiesen (vgl. Kapitel 3.1.5). Nussbaums Katalog der Grundfähigkeiten kann für sich keine Universalität beanspruchen, da er abendländisch geprägten Werte- und Normenvorstellungen entspringt. Menschen anderer Kulturräume würden diesen Katalog den eigenen kulturellen Ausprägungen anpassen (vgl. Nieke 2020b, S. 25). Deshalb schlägt Nieke die Einführung analytischer Kategorien unter dem Konzept der Lebensqualität vor, um die Bestimmung eines guten Lebens für einen wissenschaftlichen Diskurs zu öffnen. Lebensqualität setzt sich zusammen aus vier voneinander abhängigen Ordnungskategorien. Im Folgendem stelle ich die von Nieke vorgeschlagenen Kategorien samt semantisch benachbarten Termini in seinem Wortlaut dar (vgl. ebd., S. 27):

- Sicherheit (Lebensstandard, Lebensstil, Lebensbewältigung, Resonanz versus Zwang zur Singularität, digitale Welt)
- Schön (Hedonismus, Glück, Freude, Lust, Wohlbefinden, Zeitwohlstand und Ruhe)
- Sinn (Lebensform, Lebensgestaltung, das Ludische)
- Gewissheit (gutes Leben, postmaterielle Werte, Ataraxia, Serenitas)

Viele der von Nieke vorgeschlagenen Kategorien und den damit verbundenen Zielvorstellungen für Lebensqualität spiegeln sich in den in Kapitel 7 beschriebenen Interessen und Sinnstrukturen wieder. Diese führen entweder zu einem Engagement in FLOSS-Gemeinschaften oder begründen die Fortsetzung eines solchen. Daran ist zu erkennen,

dass ein Engagement in Praxisgemeinschaften der FLOSS-Bewegung sich in unterschiedlichen Aspekten auf die Lebensqualität auswirken kann.

11.6 Forschungsbedarf

In dieser Forschungsarbeit wurde wiederholt die Bedeutung von synchronen Austauschprozessen zwischen einzelnen Mitgliedern der Praxisgemeinschaften hervorgehoben. Es besteht die begründete Vermutung, dass ein großer Teil der lernwirksamen Kommunikation zwischen Mitgliedern eines informellen Lernnetzwerkes über synchrone, nichtöffentliche oder teilweise öffentliche Kommunikationskanäle erfolgt.[178] Ein lerntheoretisches Verständnis dieser Austauschprozesse zwischen Mitgliedern in Form der *peer education* würde die Theorie des informellen Lernens in Lernnetzwerken bereichern.

Ein mögliches Forschungsdesign könnte sich beispielsweise an der Methodik von Bolstad (2006) orientieren. Er fertigte Mitschnitte öffentlicher Chat-Kommunikation an, um daraus Rückschlüsse auf seinen Lernprozess zu ziehen. Ebenso bieten die öffentlich einsehbaren Austauschprozesse auf den FLOSS-Portalen zu Programmfehlern eine möglicherweise reichhaltige Datenbasis für die wissenschaftlichen Theoriebildung.

Neben dieser forschungsmethodischen Überlegung ist auch die Einordnung von konkreten Lernarrangements von informellem Lernen in Lernnetzwerken mit Blick auf Kompetenzaufbau weiterhin ein ergiebiges Forschungsfeld. Dafür bedarf es der Fortsetzung des Diskurses über eine Bildungstheorie, die Kompetenzmodelle mit dem Ziel der Befähigung zur Realisierung von Lebensqualität und der konstruktiven Auseinandersetzung mit gesellschaftlichen Problemen im Blick hat. Die Diskussion um Anerkennung informell erworbener Kompetenzen sollte sich nicht auf eine Employability im Sinnes des Humankapitals reduzieren, sondern wieder breiter aufgestellt und zusammen mit gemeinwohlorientiertem Engagement gedacht werden.

[178] Mit teilweise öffentlich werden hier öffentliche aber nicht persistente Kommunikationskanäle bezeichnet.

Literatur

Anderson, Elizabeth S. (1999): What is the Point of Equality? In: Ethics, 109, S. 287-337.

Arbeitskreis Deutscher Qualifikationsrahmen (2011): Deutscher Qualifikationsrahmen für lebenslanges Lernen. URL: https://www.dqr.de/media/content/Der_Deutsche_Qualifikationsrah-men_fue_lebenslanges_Lernen.pdf, Zugriff am: 22.11.2020.

Aristoteles (2007): Rhetorik. Reclam, Stuttgart.

Balzert, Helmut (2009): Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. 3. Auflage, Spektrum, Heidelberg.

Bandura, Albert (1977): Social Learning Theory. Prentice-Hall, Englewood-Cliffs.

Barcomb, Ann; Grottke, Michael; Stauffert, Jan-Philipp; Riehle, Dirk; Jahn, Sabrina (2015): How Developers Acquire FLOSS Skills. In: Open Source Systems: Adoption and Impact. 11th IFIP WG 2.13 International Conference 2015, S. 23-32.

Berger, Peter L.; Luckmann, Thomas (1991): The Social Construction of Reality. A Treatise in the Sociology of Knowledge. Penguin Books, London.

Bergquist, Magnus (2020): Open-Source Software Development as Gift Culture: Work and Identity Formation in an Internet Community. In: Garsten, Christina; Wulff, Helena: New Technologies at Work: People, Screens and Social Virtuality, Routledge, New York, S. 223-242.

Bloh, Egon; Lehmann, Burkhard (2002): Online-Pädagogik - der Dritte Weg? Präliminarien zur neuen Domäne der Online-(Lehr-)Lernnetzwerke (OLN). In: Bloh, Egon; Lehmann, Burkhard (Hrsg.): Online Pädagogik. Schneider, Hohengehren, S. 11-128.

Bolstad, Sverre Helge (2006): Learning and knowledge in FLOSS. Situated learning and organizational knowledge-conversion in community-based free/libre open source software development. Master Thesis at University of Bergen, URL: https://flosshub.org/-sites/flosshub.org/files/Learning-and-knowledge-in-FLOSS.pdf, Zugriff am: 11.04.2020.

Bourdieu, Pierre (1983): Ökonomisches Kapital, kulturelles Kapital, soziales Kapital. In: Kreckel, Reinhard (Hrsg.): Soziale Ungleichheiten. Otto Schwartz & Co, Göttingen, S. 183-198.

Brand, Andreas (2009): Softwareentwicklung im Netzwerk. Kooperation, Hierarchie und Wettbewerb in einem Open Source-Projekt. Rainer Hamp, München.

Brand, Klaus Werner; Büsser, Detlef; Rucht, Dieter (1986): Aufbruch in eine andere Gesellschaft: Neue soziale Bewegungen in der Bundesrepublik. Campus, Frankfurt/Main.

Bruner, Jerome (1983): Child's Talk. Learning to Use Language. Oxford University Press, Oxford.

Capiluppi, Andrea; Barahona, Jesus M. Gonzalez; Herraiz; Israel; Robles, Gregorio (2007): Adapting the "Staged Model for Software Evolution" to FLOSS. URL: https://www.researchgate.net/publication/220875490_Adapting_the_staged_model_for_software_evolution_to_freelibreopen_source_software, Zugriff am: 30.1.2020.

Coelen, Thomas; Gusinde, Frank; Lieske, Nina; Trautmann, Matthias (2016): Informelles Lernen in der Schule. In: Rohs, Matthias (Hrsg.): Handbuch informelles Lernen.

Springer VS, Wiesbaden, S. 325-342.

Debian (o.D.): About History; URL: https://www.debian.org/intro/about#history, Zugriff am 05.05.2021.

Dehnbostel, Peter (2018): Beruf und informelles Lernen. In: Harring, Marius; Witte, Matthias D.; Burger, Timo (Hrsg.): Handbuch informelles Lernen. Interdisziplinäre und internationale Perspektiven. 2. Auflage, Beltz Juventa; Weinheim, S. 426-439.

della Porta, Donatella (2007): The Global Justice Movement: An Introduction. In: della Porta, Donatella (Eds.): The Global Justice Movement. Cross-national and Transnational Perspectives. Paradigm Publishers, Boulder, S. 1-28.

Dewe, Bernd; Weber, Peter J. (2007): Einführung in moderne Lernformen. Beltz, Weinheim.

Dewey, John (2001): Democracy and Education. The Pennsylvania State University, URL: https://nsee.memberclicks.net/assets/docs/KnowledgeCenter/BuildingExpEduc/Books-Reports/10.%20democracy%20and%20education%20by%20dewey.pdf, Zugriff am 05.05.2021 [E-Book].

Dillenbourg, Pierre (1999): What do you mean by collaborative learning? In: Dillenbourg, Piere (Eds.): Collaborative-learning: Cognitive and Computational Approaches. Elsevier, Oxford, S. 1-19.

Dohmen, Günther (2001): Das informelle Lernen. Die internationale Erschließung einer bisher vernachlässigten Grundform menschlichen Lernens für das lebenslange Lernen aller. Bundesministerium für Bildung und Forschung, Bonn.

Dohmen, Günther (2018): Das informelle Lernen. In: Harring, Marius; Witte, Matthias D.; Burger, Timo (Hrsg.): Handbuch informelles Lernen. Interdisziplinäre und internationale Perspektiven. 2. Auflage, Beltz Juventa, Weinheim, S. 53-60.

Döring, Nicola (2001): Virtuelle Gemeinschaften als Lerngemeinschaften!? Zwischen Utopie und Dystopie. URL:https://www.die-bonn.de/zeitschrift/32001/positionen4.htm, Zugriff am 19.02.2020.

Downes, Stephen (2010): Learning Networks and Connective Knowledge. URL: https://-www.downes.ca/files/Learning%20Networks%20and%20Connective%20Knowledge%20-Yuen.pdf, Zugriff am 18.01.2020.

Downes, Stephen (2013): MOOC - The Resurgence of Community in Online Learning. URL: https://halfanhour.blogspot.com/2013/05/mooc-resurgence-of-community-in-online.html, Zugriff am 24.03.2021.

Düx, Wiebken; Prein, Gerald; Sass, Erich; Tully, Claus (2008): Kompetenzerwerb im freiwilligen Engagement. Eine empirische Studie zum informellen Lernen im Jugendalter. 2. Auflage, VS, Wiesbaden.

Ebner, Martin; Lorenz, Anja (2012): Web 2.0 als Basistechnologien für CSCL-Umgebungen. In: Haake, Jörg M.; Schwabe, Gerhard; Wessner , Martin: CSCL-Kompendium 2.0 Lehr- und Handbuch zum computerunterstützten, kooperativen Lernen. 2. Ausgabe, Oldenbourg Wissenschaftsverlag, München, S. 97-110.

Eseryel, U. Yeliz; Wei, Kangning; Crowston, Kevin (2020): Decision-making Processes in Community-based Free/Libre Open Source Software-development Teams with Internal Governance: An Extension to Decision-making Theory. In: Communications of the Association for Information Systems, 46, S. 484-510.

Ettrich, Matthias (2004): Koordination und Kommunikation in Open-Source-Projekten. In: Gehring, Robert A.; Lutterbeck; Bernd (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Lehmanns Media, Berlin, S. 179-192.

Europäische Kommission (2001): Mitteilung der Kommission. Einen europäischen Raum des lebenslangen Lernens schaffen. URL: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=COM:2001:0678:FIN:DE:PDF, Zugriff am 03.06.2019.

Faulstich, Peter (2007): Lebenserfahrung Älterer als Lernvoraussetzung für gute Arbeit. In: Faulstich, Peter; Bayer, Mechthild (Hrsg.): Lernalter. Weiterbildung statt Altersarmut. VSA, Hamburg, S. 8-28.

Faulstich, Peter (2013): Menschliches Lernen. Eine kritisch-pragmatische Lerntheorie. transcript, Bielefeld.

Flick, Uwe (2016): Qualitative Sozialforschung. Eine Einführung. 7. Auflage, Rowohlt, Reinbek.

Free Software Foundation (o.D.): About FSF. URL: https://www.fsf.org/about, Zugriff am 05.05.2021.

Frost, Ingo; Helmeth, Wolfgang; Rohs, Matthias (2007): Informelles Lernen in der beruflichen Bildung. Die Diskussion in Europa und die Realität in Afrika. In: ZEP: Zeitschrift für internationale Bildungsforschung und Entwicklungspädagogik, 30 (4), S. 2-8.

Gamalielsson, Jonas; Lundell, Björn (2014): Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? erschienen in: The Journal of Systems and Software, Elsevier, 89, S. 128-145.

Gardner, Howard (1985): The Mind's New Science. A History of the Cognitive Revolution. Basic Books, New York.

Gergen, Kenneth J.; Wortham, Stanton (2001): Social Construction and Pedagogical Practice. In: Gergen; Kenneth J.: Social Construction in Context. Sage Publications, London, S. 115-136.

Gerstenmaier, Jochen; Mandl, Heinz (1995): Wissenserwerb unter konstruktivistischer Perspektive. In: Zeitschrift für Pädagogik, Beltz, 41 (6), S. 867-888.

Gerstenmaier, Jochen; Mandl, Heinz (2018): Konstruktivistische Ansätze in der Erwachsenenbildung und Weiterbildung. In: Tippelt, Rudolf; Hippel, Aiga von: Handbuch Erwachsenenbildung/Weiterbildung. 6. Auflage, Springer VS, Wiesbaden, S. 221-233.

Ghosh, Rishab A. (2007): Understanding Free Software Developers: Findings from the FLOSS Study. In: Feller, Joseph; Fitzgerald, Brian; Hissam, Scott A.; Lakhani, Karim R.: Perspectives on Free and Open Source Software. MIT Press, Cambridge, S. 23-45.

Ghosh, Rishab A.; Glott, Rüdiger; Krieger, Bernhard; Robles, Gregorio (2002): Free/Libre and Open Source Software: Survey and Study. Part 4: Survey of Developers. URL: https://www.math.unipd.it/~bellio/FLOSS%20Final%20Report%20-%20Part%204%20--%20Survey%20of%20Developers.pdf, Zugriff am: 28.11.2020.

Gläser, Jochen; Laudel, Grit (2009): Experteninterviews und qualitative Inhaltsanalyse. 3. Auflage, VS, Wiesbaden.

Glasersfeld, Ernst von (1996): Radikaler Konstruktivismus. Ideen, Ergebnisse, Probleme. Suhrkamp, Frankfurt/Main.

Glasersfeld, Ernst von; Fischer, Rudi (Hrsg.) (2013): Wege des Wissens. Konstruktivistische Erkundungen durch unser Denken. 2. korrigierte Auflage, Carl-Auer, Heidelberg.

Glott, Rüdiger; Meiszner, Andreas; Sowe, Sulayman K. (2007): FLOSSCom - Using the Principles of Informal Learning Environments of FLOSS Communities to Improve ICT Supported Formal Education. URL: https://flosshub.org/sites/flosshub.org/files/-FLOSSCom_WP2_Phase_1_Report_v070709_1.pdf, Zugriff am: 28.11.2020.

Göhlich, Michael; Zirfas, Jörg (2007): Lernen. Ein pädagogischer Grundbegriff. W. Kohlhammer, Stuttgart.

Gräsel, Cornelia (2011): Was ist Empirische Bildungsforschung? In: Reinders, Heinz; Ditton, Hartmut; Gräsel, Cornelia; Gniewosz, Burkhard (Hrsg.): Empirische Bildungsforschung. Gegenstandsbereiche. VS, S. 13-28.

Grassmuck, Volker (2004): Freie Software. Zwischen Privat- und Gemeineigentum. 2. Auflage, Bundeszentrale für politische Bildung, Bonn.

Haake, Jörg M.; Schwabe, Gerhard; Wessner, Martin (2012): Grundlagen. In: Haake, Jörg M.; Schwabe, Gerhard; Wessner , Martin: CSCL-Kompendium 2.0 Lehr- und Handbuch zum computerunterstützten, kooperativen Lernen. 2. Ausgabe, Oldenbourg Wissenschaftsverlag, München, S. 1-5.

Hambridge, Sally (1995): Netiquette Guidelines. Request For Comments Nr.: 1855 ,URL: https://tools.ietf.org/html/rfc1855, Zugriff am 25.02.2019.

Hannemann, Anna; Klamma, Ralf; Jarke, Matthias (2012): Soziale Interaktion in Open-Source-Community. In: Strahringer, Susanne (Hrsg.): HMD – Praxis der Wirtschaftsinformatik: Open Source – Konzepte, Risiken, Trends. dpunkt, 283, S. 26-37.

Hars, Alexander; Ou, Shaosong (2001): Working for Free? – Motivations of Participating in Open Source Projects. Proceedings of the 34th Hawaii International Conference on System Sciences – 2001. URL: https://www.cin.ufpe.br/~if722/2006.2/readings/-02/hars%20&%20ou%202001.pdf, Zugriff am: 02.05.2021.

Haug, Simone; Wedekind, Joachim (2013): cMOOC - ein alternatives Lehr-/Lernszenarium? In: Schulmeister, Rolf (Hrsg.): MOOCs — Massive Open Online Courses. Offene Bildung oder Geschäftsmodell? Waxmann, Münster, S. 161-206.

Hertel, Guido; Niedner, Sven; Herrmann, Stefanie (2003): Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel. In: Research Policy. Elsevier, 32 (7), S. 1159-1177.

Hinze, Udo (2008): Computerbasiertes kooperatives Lernen (CSCL) als technische und pädagogische Herausforderung. In: Gross, Friederike von; Marotzki, Winfried; Sander, Uwe (Hrsg.): Internet – Bildung – Gemeinschaft. VS, Wiesbaden, S. 241-262.

Holtgrewe, Ursula (2004): Heterogene Ingenieure – Open Source und Freie Software zwischen technischer und sozialer Innovation. In: Gehring, Robert A.; Lutterbeck, Bernd (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Lehmanns Media, Berlin, S. 339-352.

Holzkamp, Klaus (1995): Lernen. Subjektwissenschaftliche Grundlegung. Campus, Frankfurt/Main.

Iske, Stefan (2018): Digitale Medien und informelles Lernen. In: Harring, Marius; Witte, Matthias D.; Burger, Matthias (Hrsg.): Handbuch informelles Lernen. Interdisziplinäre und internationale Perspektiven. 2. Auflage, Beltz Juventa, Weinheim, S. 516-538.

Johnson, David W.; Johnson, Roger T. (2008): Social Interdependence Theory and Cooperative Learning: The Teacher's Role. In: Robyn M. Gillies, Robyn M.; Ashman, Adrian F., Terwel, Jan: The Teacher's Role in Implementing Cooperative Learning in the Classroom. Springer Science + Business, New Yorck, S. 9-37.

Kade, Sylvia (1999): Qualitative Erwachsenenbildungsforschung. Methoden und Ergebnisse. erschienen in: Tippelt, Rudolf: Handbuch Erwachsenenbildung / Weiterbildung. 2. Auflage, Leske + Budrich, Opladen, S. 340-359.

Kahnwald, Nina (2013): Informelles Lernen in virtuellen Gemeinschaften. Nutzungspraktiken zwischen Information und Partizipation. Waxmann, Münster.

Kerres, Michael; Nattland, Axel (2012): Didaktische Konzeption von CSCL-Arrangements. In: Haake, Jörg M.; Schwabe, Gerhard; Wessner , Martin: CSCL-Kompendium 2.0 Lehr- und Handbuch zum computerunterstützten, kooperativen Lernen. 2. Ausgabe, Oldenbourg Wissenschaftsverlag, München, S. 254-260.

Khartitoniouk, Svetlana; Stewin, Patrick (2004): Grundlagen und Erfahrungen. Einleitung. In: Gehring, Robert A.; Lutterbeck, Bernd (Hrsg.): Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Lehmanns Media, Berlin, S. 1-16.

Kluge, Susann (1999): Empirisch begründete Typenbildung. Zur Konstruktion von Typen und Typologien in der qualitativen Sozialforschung. Springer Fachmedien, Wiesbaden.

Kolb, David A. (1984): Experiential Learning: Experience As The Source Of Learning And Development. Prentice Hall, Englewood Cliffs.

Konrad, Klaus (2014): Lernen lernen – allein und mit anderen. Konzepte, Lösungen, Beispiele. Springer VS, Wiesbaden.

Krapp, Andreas (1992a): Konzepte und Forschungsansätze zur Analyse des Zusammenhangs von Interesse, Lernen und Leistung. In: Krapp, Andreas; Prenzel, Manfred (Hrsg.): Interesse. Lernen, Leistung. Neuere Ansätze der pädagogisch-psychologischen Interessensforschung. Aschendorf, Münster, S. 9-52.

Krapp, Andreas (1992b): Das Interessenkonstrukt. Bestimmungsmerkmale der Interes-

senhandlung und des individuellen Interesses aus Sicht einer Person-Gegenstands-Konzeption. In: Krapp, Andreas; Prenzel, Manfred (Hrsg.): Interesse. Lernen, Leistung. Neuere Ansätze der pädagogisch-psychologischen Interessensforschung. Aschendorf, Münster, S. 297-329.

Kraus, Anja; Budde, Jürgen; Hietzge, Maud; Wulf, Christian (2017): ‚Schweigendes' Wissen in Lernen und Erziehung, Bildung und Sozialisation. Einführung. In: Kraus, Anja; Budde, Jürgen; Hietzge, Maud; Wulf, Christian (Hrsg.): ‚Schweigendes' Wissen in Lernen und Erziehung, Bildung und Sozialisation. Juventa/Beltz, Weinheim, S. 11-15.

Kreitz, Robert; Miethe, Ingrid; Tervooren, Anja (Hrsg.) (2016): Theorien in der qualitativen Bildungsforschung – Qualitative Bildungsforschung als Theoriegenerierung. Barbara Budrich, Opladen.

Krogh, Georg von; Haefliger, Stefan; Spaeth, Sebastian; Wallin, Martin W. (2012): Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. In: MIS Quarterly, 2012, 36 (2), S. 649-676.

Krogh, Georg von; Hippel, Eric von (2006): The Promise of Research on Open Source Software. In: Management Science, 52 (7), S. 975-983.

Kuckartz, Udo (2010): Einführung in die computergestützte Analyse qualitativer Daten. 3. Auflage, VS, Wiesbaden.

Kuckartz, Udo (2016): Typenbildung und typenbildende Inhaltsanalyse in der empirischen Sozialforschung. In: Schnell, Martin W.; Schulz, Christian; Kuckartz, Udo; Dunger, Christine: Junge Menschen sprechen mit sterbenden Menschen. Springer VS, Wiesbaden, S. 31-53.

Kuckartz, Udo (2018): Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstützung. 4. Auflage, Beltz Juventa, Weinheim.

Kuk, Georg (2006): Strategic Interaction and Knowledge Sharing in the KDE Developer Mailing List. In: Management Science, 52 (7), S. 1031-1042.

Lakhani, Karim R.; Wolf, Robert G. (2003): Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects. 2003, MIT Sloan School of Management, URL: http://ssrn.com/abstract=443040, Zugriff am: 04.03.2020.

Lamnek, Siegfried: Qualitative Sozialforschung. 2010, 5. Auflagen, Beltz, Weinheim.

Lave, Jean; Wenger, Etienne (1991): Situated learning. Legitimate peripheral participation. Cambridge University Press, Cambridge.

Levy, Steven (2010): Hackers. Heroes of the Computer Revolution. O'Reilly Media, Sebastopol.

Ley, Tobias; Seitlinger, Paul; Schöfegger, Karin; Lindstaedt, Stefanie N. (2012): Community-orientiertes Lernen . In: CSCL-Kompendium 2.0: Lehr- und Handbuch zum computerunterstützten, kooperativen Lernen. 2. Ausgabe, Oldenbourg Wissenschaftsverlag, München, S. 261-273.

Linux User (o.D.): Verzeichnis der Linux-Usergroups im deutschen Sprachraum. URL:

https://www.linux-user.de/LUG, Zugriff am 05.05.2021.

Marotzki, Winfried (2006): Forschungsmethoden und -methodologie der Erziehungswissenschaftlichen Biographieforschung. In: Krüger, Heinz-Hermann; Krüger; Marotzki, Winfried (Hrsg.): Handbuch erziehungswissenschaftliche Biographieforschung. 2. Auflage, VS, Wiesbaden, S. 111-136.

Marotzki, Winfried; Jörissen, Benjamin (2008): Wissen, Artikulation und Biographie: theoretische Aspekte einer Strukturalen Medienbildung. In: Fromme, Johannes; Sesink, Werner (Hrsg.): Pädagogische Medientheorie. VS, Wiesbaden, S. 51-70.

Marsick, Victoria J.; Watkins, Karen (2015): Informal and Incidental Learning in the Workplace. 2. Edition, Routledge, Abingdon.

Mayring, Philipp (2015): Qualitative Inhaltsanalyse. Grundlagen und Techniken. 12. Auflage, Beltz, Weinheim

McAdam, Doug; Snow, David A. (1997): Social Movements. Readings on Their Emergence, Mobilisation and Dynamics. Roxbury Publishing Company, Los Angeles.

McLuhan, Marshall (1962): The Gutenberg Galaxy: The Making of Typographic Man. University of Toronto Press, Toronto.

Meyer-Drawe, Käte (2001): Leiblichkeit und Sozialität. Phänomenologische Beiträge zu einer pädagogischen Theorie der Inter-Subjektivität. 3. Auflage, Wilhelm Fink, München.

Meyer-Drawe, Käte; Grabau, Christian (2018): Diskurse des informellen Lernens und deren Bedeutung im gesellschaftlichen Kontext. In: Harring, Marius; Witte, Matthias D.; Burger, Timo (Hrsg.): Handbuch informelles Lernen. Interdisziplinäre und internationale Perspektiven. 2. Auflage, Beltz Juventa, Weinheim, S. 62-72.

Mikkonen, Teemu; Vadén, Tere; Vainio, Niklas (2007): The Protestant ethic strikes back: Open source developers and the ethic of capitalism. URL: https://firstmonday.org/ojs/-index.php/fm/article/download/1623/1538, Zugriff am 21.03.2020.

Miller, George A. (2003): The cognitive revolution: a historical perspective. In: TRENDS in Cognitive Sciences, 7 (3), S. 141-144.

Mittelstraß, Jürgen (1989): Glanz und Elend der Geisteswissenschaften. Bibliotheks- und Informationssystem der Universität Oldenburg, Oldenburg.

Nakakoji, Kumiyo; Yamamoto, Yasuhiro; Nishinaka, Yoshiyuki; Kishida, Kouichi; Ye, Yunwen (2003): Evolution Patterns of Open-Source Software Systems and Communities. URL: https://www.researchgate.net/publication/2555433_Evolution_Patterns_of-_Open-Source_Software_Systems_and_Communities, Zugriff am: 20.03.2020.

Narayan, Deepa; Petesch, Patti. (2002): Voices of the Poor. From Many Lands. World Bank Group, Washington.

Nepstad, Sharon Erickson; Bob, Clifford (2006): When Do Leaders Matter? Hypotheses on Leadership Dynamics in Social Movements“. In: Mobilization: An International Quarterly. 11 (1), S. 1-22.

Nieke, Wolfgang (2012): Kompetenz und Kultur. Beiträge zur Orientierung in der Moder-

ne. Springer VS, Wiesbaden.

Nieke, Wolfgang (2020a): Kompetenzen. In: Bollweg, Petra; Buchna, Jennifer; Coelen, Thomas; Otto, Hans-Uwe (Hrsg.): Handbuch Ganztagsbildung. 2. Auflage, Springer VS, Wiesbaden, S. 355-366.

Nieke, Wolfgang (2020b): Lebensqualität als Orientierung für pädagogisches Denken. In: Hübner, Edwin; Weiss, Leonhard (Hrsg.): Resonanz und Lebensqualität. Weltbeziehung in Zeiten der Digitalisierung. Pädagogische Perspektiven, Barbara Budrich, Opladen, S. 23-58.

Nussbaum, Martha C. (2006): Education and Democratic Citizenship: Capabilities and Quality Education. In: Journal of Human Development, 7 (3), S. 385-395.

Nussbaum, Martha C. (2009): Tagore, Dewey, and the Imminent Demise of Liberal Education. In: Siegel, Harvey (Eds.): The Oxford Handbook of Philosophy of Education. [E-Book].

Nussbaum, Martha C. (2012): Gerechtigkeit oder das gute Leben. 7. Auflage, Suhrkamp, Frankfurt/Main.

Oelkers, Nina; Otto, Hans-Uwe; Ziegler, Holger (2010): Handlungsbefähigung und Wohlergehen: Der Capabilities-Ansatz als alternatives Fundament der Bildungs- und Wohlfahrtsforschung. In: Otto, Hans-Uwe; Ziegler, Holger (Hrsg.): Capabilities – Handlungsbefähigung und Verwirklichungschancen in der Erziehungswissenschaft. 2. Auflage, VS, Wiesbaden, S. 85-89.

Open Source Initiative (o.D.): About OSI. URL: https://opensource.org/about, Zugriff am 05.05.2021.

Open Source Survey (2017): The Open Source Survey is an open data project by GitHub and collaborators from academia, industry, and the broader open source community. URL: https://opensourcesurvey.org/2017/, Zugriff am 24.02.2021.

Otto, Hans-Uwe; Scherr, Albert; Ziegler, Holger (2010): Wieviel und welche Normativität benötigt die Soziale Arbeit? Befähigungsgerechtigkeit als Maßstab sozialarbeiterischer Kritik. In: neue praxis, 40, S. 137-163.

Overwien, Bernd (2016): Informelles Lernen und politische Bildung. In: Rohs, Matthias: Handbuch Informelles Lernen. Springer, Wiesbaden, S. 399-412.

Piaget, Jean; Inhelder, Bärbel (2000): The Psychology of the Child. Basic Books, New York.

Raschke, Joachim (1988): Soziale Bewegungen. Ein historisch-systematischer Grundriß. 2. Auflage, Campus, Frankfurt/Main

Rawls, John (1971): A Theory of Justice. Harvard University Press, Cambridge.

Raymond, Eric Steven (2000): Homesteading the Noosphere. URL: https://pdfs.semanticscholar.org/98a6/c566a7e28a48facb664c8689607a52c57a5d.pdf, Zugriff am 26.05.2020.

Raymond, Eric Steven; Moen, Rick (2014): How To Ask Questions The Smart Way. URL: http://www.catb.org/~esr/faqs/smart-questions.html#before, Zugriff am 27.10.2020.

Reinders, Heinz (2014): Jugend - Engagement - Politische Sozialisation. Gemeinnützige Tätigkeit und Entwicklung in der Adoleszenz. Springer VS, Wiesbaden.

Reinders, Heinz (2016): Service Learning – Theoretische Überlegungen und empirische Studien zu Lernen durch Engagement. Beltz Juventa, Weinheim.

Reinders, Heinz (2018): Gemeinnützige Tätigkeit und informelles Lernen. In: Harring, Marius; Witte, Matthias D.; Burger, Timo (Hrsg.): Handbuch informelles Lernen. Interdisziplinäre und internationale Perspektiven. 2018, 2. Auflage, Beltz Juventa; Weinheim, S. 440-454.

Reinders, Heinz; Youniss, James (2006): School-Based Required Community Service and Civic Development in Adolescents. In: Applied Developmental Science. 10 (1), S. 2-12.

Reinmann, Gabi (2007): Wissen – Lernen – Medien. E-Learning und Wissensmanagement als medienpädagogische Aufgaben. In: Sesink, Werner; Kerres, Michael; Moser, Heinz (Hrsg.): Jahrbuch Medienpädagogik 6. Medienpädagogik – Standortbestimmung einer erziehungswissenschaftlichen Disziplin. VS, Wiesbaden, S. 179-199.

Reinmann, Gabi (2016): Gestaltung akademischer Lehre: semantische Klärungen und theoretische Impulse zwischen Problem- und Forschungsorientierung. In: Zeitschrift für Hochschulentwicklung. 11 (5), S. 225-244.

Reinmann-Rothmeier, Gabi (2003): Didaktische Innovation durch Blended Learning. Leitlinien anhand eines Beispiels aus der Hochschule. Hans Huber, Bern.

Reinmann-Rothmeier, Gabi; Mandl, Heinz (1997): Lernumgebungen mit Neuen Medien gestalten. In: Günther, Wilfried; Mandl, Heinz (Hrsg.): Telelearning. Aufgaben und Chance für Bildung und Gesellschaft. Economica, Bonn, S. 105-113.

Rogers, Robert Johnson (2016): Developing Implicit and Explicit Knowledge of L2 Case Marking Under Incidental Learning Conditions. URL: https://discovery.ucl.ac.uk/id/eprint/1498849/3/Rogers%20Thesis%202016%20Final.pdf, Zugriff am 28.12.2020.

Rohs, Matthias (2013): Social Media und informelles Lernen. Potenziale von Bildungsprozessen im virtuellen Raum. In: DIE Zeitschrift für Erwachsenenbildung. 2 , S. 39-42.

Rosa, Hartmut (2016): Resonanz. Eine Soziologie der Weltbeziehung. Suhrkamp, Berlin.

Roth, Gerhard (2001): Fühlen, Denken, Handeln. Wie das Gehirn unser Verhalten steuert. Suhrkamp, Stuttgart.

Roth, Gerhard (2003): Aus Sicht des Gehirns. Suhrkamp, Frankfurt/Main.

Roth, Gerhard (2011): Bildung braucht Persönlichkeit. Wie Lernen gelingt. Klett-Cotta, Stuttgart, [E-Book].

Roth, Heinrich (1971): Pädagogische Anthropologie. Hermann Schroedel, Hannover.

Schön, Sandra; Ebner, Martin (2020): Ziele von Makerspaces. Didaktische Perspektiven. In: Heinzel,Viktoria; Seidl, Tobias; Stang, Richard (Hrsg.): Lernwelt Makerspace. Grundlagen, Konzepte und Perspektiven, DeGruyter, Berlin, S. 33-47.

Schuck, Hartwig (2012): Macht und Herrschaft. Eine realistische Analyse. In: Elbe, In-

go; Ellmers, Sven; Eufinger, Jan (Hrsg.): Anonyme Herrschaft. Zur Struktur moderner Machtverhältnisse. Westfälisches Dampfboot, Münster, S. 35-81.

Schümmer, Till; Haake, Jörg (2012): Kommunikation und Awareness. In: Haake, Jörg M.; Schwabe, Gerhard; Wessner, Martin: CSCL-Kompendium 2.0 Lehr- und Handbuch zum computerunterstützten, kooperativen Lernen. 2012, 2. Ausgabe, Oldenbourg Wissenschaftsverlag, München, S. 84-96.

Siemens, George (2004): Connectivism: A Learning Theory for the Digital Age. In: International Journal of Instructional Technology and Distance Learning; URL: http://-citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.87.3793, Zugriff am 16.01.2020.

Skinner, Burrhus F. (1971): Erziehung als Verhaltensformung. Grundlagen einer Technologie des Lehrens. E. Keimer, München.

SourceForge (o.D.), About SourceForge; URL: https://sourceforge.net/about, Zugriff am 05.05.2021.

Sowe, Sulayman K.; Stamelos, Ioannis; Angelis, Lefteris (2007): Understanding knowledge sharing activities in free/open source software projects: An empirical study. In: The Journal of Systems and Software, 81, S. 431-446.

Speck, Otto (2008): Hirnforschung und Erziehung. Eine pädagogische Auseinandersetzung mit neurobiologischen Erkenntnissen. Ernst Reinhardt, München.

Spitzer, Manfred (2002): Lernen. Gehirnforschung und die Schule des Lebens. Spektrum, Heidelberg.

Squire, Megan (2017): Considering the Use of Walled Gardens for FLOSS Project Communication. In: Balaguer, Federico; Cosmo, Roberto Di; Garrido, Alejandra; Kon, Fabio; Robles, Gregorio; Zacchiroli, Stefano: Open Source Systems: Towards Robust Practices. 13th IFIP WG 2.13 International Conference, OSS 2017 Buenos Aires, 2017 Proceedings, Springer, S. 3-13.

Stallmann, Richard M. (2015): Free Software, Free Society: Selected Essays of Richard M. Stallmann. 3. Auflage, Free Software Foundation, Boston.

Stegbauer, Christian; Rausch, Alexander (2001): Die schweigende Mehrheit – „Lurker“ in internetbasierten Diskussionsforen. In: Zeitschrift für Soziologie. 30 (1), S. 48-64.

Taubert, Niels C. (2006): Produktive Anarchie? Netzwerke freier Softwareentwicklung. transcript, Bielefeld.

Taubert, Niels C. (2008): Balancing Requirements of Decision and Action: Decision-Making and Implementation in Free/Open Source Software Projects. In: Science, Technology & Innovation Studies; 4 (1), S. 69-88.

Tepe, Daniel; Hepp, Andreas (2008a): Digitale Produktionsgemeinschaften: Open-Source-Bewegung als deterritoriale Vergemeinschaftung. In: Lutterbeck et al.: Open Source Jahrbuch 2008. Zwischen freier Software und Gesellschaftsmodell. Lehmanns Media, Berlin, S. 171-185.

Tepe, Daniel; Hepp, Andreas (2008b): Digitale Produktionsgemeinschaften. Die Open-Source-Bewegung zwischen kooperativer Softwareherstellung und deterritorialer poli-

tischer Vergemeinschaftung. In: Stegbauer, Christian; Jäckel, Michael (Hrsg.): Social Software. Formen der Kooperation in computerbasierten Netzwerken. VS Verlag für Sozialwisschenschaften, Wiesbaden, S. 27-48.

Uhl, Johanna (2019): Informelle Sprachlernbegegnungen mit dem Englischen von Kindern und Jugendlichen bei der Nutzung mobiler Technologien. URL: https://opus4-.kobv.de/opus4-ku-eichstaett/files/545/Dissertation_J_Uhl_final_PDFa.pdf, Zugriff am: 28.12.2020.

Vygotskij, Lev Semënovič (2002): Denken und Sprechen. Psychologische Untersuchungen. Herausgegeben und aus dem Russischen übersetzt von Joachim Lompscher und Georg Rückriem. Beltz, Weinheim.

Walker, Melanie (2006): Higher Education Pedagogies. A capabilities Approach. 2006, Open University Press, Berkshire.

Warner, Jason (2018): Thank you for 100 million repositories. URL: https://github.blog/-2018-11-08-100m-repos/, Zugriff am 25.03.2014.

Watson, John B. (1913): Psychology as the Behaviorist Views it. Psychological Review, 20, S. 158-177, URL: http://psychclassics.yorku.ca/Watson/views.htm, Zugriff am 19.06.2019.

Weber, Max; Winckelmann, Johannes (Hrsg.) (1980): Wirtschaft und Gesellschaft. Grundriß einer verstehenden Soziologie. 5. Auflage, Mohr Siebeck, Tübingen.

Wenger, Etienne (1998): Communities of Practice. Learning, Meaning, and Identity. Cambridge University Press, Cambridge.

Wenger, Etienne; McDermott, Richard; Snyder, William M. (2002): Cultivating Communities of Practice: A Guide to Managing Knowledge. Harvard Business School Press, Boston.

Wessner, Martin: Lerngruppen (2012): In: Haake, Jörg M.; Schwabe, Gerhard; Wessner , Martin: CSCL-Kompendium 2.0 Lehr- und Handbuch zum computerunterstützten, kooperativen Lernen. 2. Ausgabe, Oldenbourg Wissenschaftsverlag, München, S. 200-205.

Wolf, Karsten D.; Wudarski, Urszula (2018): Communicative Figurations of Expertization: DIY_MAKER and Multi-Player Online Gaming (MOG) as Cultures of Amateur Learning. In: Hepp, Andreas; Breiter, Andreas; Hasebrink, Uwe (Eds.): Communicative Figurations. Transforming Communications in Times of Deep Mediatization. Studies in Cross-Media Research, Palgrave Macmillan, Cham, S. 123-149.

Wood, David; Bruner, Jerome S.; Ross, Gail (1976): The Role of Tutoring in Problem Solving. In: Journal of Child Psychiatry and Psychology, 17, Pergamon Press, S. 89-100.

Ziegler, Holger (2018): Der Capabilities Ansatz und informelles Lernen. In: Harring, Marius; Witte, Matthias D.; Burger, Timo (Hrsg.): Handbuch informelles Lernen. Interdisziplinäre und internationale Perspektiven. 2. Auflage, Beltz Juventa, Weinheim, S. 73-85.

Zimmermann, Thomas (2004): Open Source und Freie Software – soziale Bewegung im virtuellen Raum? In: Gehring, Robert A.; Lutterbeck, Bernd: Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell. Lehmanns Media, Berlin, S. 353-368.

Anhang

Interviewleitfragen

Interesse

1. Warum hast du damals angefangen, dich an FLOSS-Projekten zu beteiligen?
2. Bist du heute noch aktiv? Hat sich dein Interesse gegenüber früher geändert?
3. Was erhoffst du durch dein Engagement zu erreichen?

Handlungsfelder

1. Was hast du im FLOSS-Projekt [Name] gemacht?
2. Warum hast du dir diesen Bereich ausgesucht?
3. Warst du früher in anderen Bereichen aktiv? Wenn ja: Warum hast du gewechselt?

Prozess

1. Was hat dir bei deinem Engagement geholfen?
2. Was hat dich dabei gehindert?
3. Welchen Bedeutung hat die Community vom Projekt [Name] für dich?
4. Welche Schlüsselereignisse waren für dich wichtig?
5. Wo findet die meiste Kommunikation zwischen Entwickler:innen statt?
6. Welchen Stellenwert hat die öffentliche Mailingliste?

Bildung

1. Wenn du die gesamte Zeit zurückblickst bis zu deinem ersten Beitrag zu einem gemeinschaftlichen FLOSS-Projekt: Was waren Prozesse oder auch Ereignisse, die deine Sicht auf dich selbst oder die Gesellschaft verändert haben?

Kompetenz

1. Was kannst du durch dein Engagement besser als früher?

Wissen

1. Welche Kenntnisse hast du vertieft bzw. dir neu angeeignet?

Fazit

1. Wie würde deiner Meinung nach ein ideales FLOSS-Projekt aussehen?
2. Welche Strukturen und Arbeitsprozesse sind notwendig, um eine aktive Community von Entwickler:innen aufzubauen und zu motivieren?

Kategoriensystem der Interviews

Liste der Codes / Häufigkeit

Tätigkeiten / Rahmenbedingungen
- MISC_als_Spezialzitat 21
- Nutzer-Support 2
- Testen 3
- Upstream 7
- Nutzerhilfe 3
- Entwicklung 5
- Entwicklungsgemeinschaft 18
- Paket-Maintainer 9
- Bugfix 6
- Probleme v. FLOSS-Entw. 33
- Vorbedingungen 22
- Schlüsselereignis 28

Kategorienkomplex Interessen
- Austausch mit anderen 6
- Interesse Thema 4
- Erwartungen erfüllen 2
- Reputation 4
- Mitmachen_können 2
- Gleichgesinnte treffen 10
- FLOSS-Freundschaft 3
- Alternative_zu_propr. SW 5
- Lernen_durch_FLOSS 6
- FLOSS als Idee 15
- Paket in Distro zu bekommen 2
- Spaß 9
- Um Problem zu lösen 29
- Moderner als propr. SW 1
- Dinge verbessern 6
- Sicherheit 3
- FLOSS_als_Lohnarbeit 9
- Anerkennung 17
- Kritik_an_geschl_Lizenzen 3
- anderen_helfen 19
- Vorbildfunktion 3

Kategorienkomplex Lernprozess
- Trial-and-error 2
- Kommunikation 17
- F2F / Event 14
- Forum 7
- E-Mail 5
- Mailinglisten 18
- FLOSS-Portal 16
- Kommunikation (synchron) 24
- allgemein 3
- Idealprojekt 16
- Vorbilder 2
- Code lesen+verstehen 7
- Info-Quelle_pers._Umfeld 3
- Info-Quelle_Tutorial 2
- Info-Quelle_FLOSS community 5
- Info-Quelle_Foren 10
- Info-Quelle_Dokumentation 8
- Info-Quelle_Feedback 6

Kategorienkomplex Kompetenzaufbau
- Betriebssysteme 1
- Projektmanagement 2
- Stärke von Gemeinschaft 5
- Selbstvertrauen 10
- Entw.-Tools 5
- SW-Entwicklung 6
- Design 1
- Webseite_aufsetzen 1
- Umgang mit Menschen 12
- Paketierung 4
- In fremdes Projekt einarbeiten 2
- Programmier-/Dokustil 4
- Datenschutz 1
- allgemein 10
- Lizenzen 1
- Administration 8
- Software 16

Kategoriensystem der Mailinglisten

Liste der Codes / Häufigkeit

Metainformationen
- externe_Dev_Kommu 19
- Datum 445
- Zitat 296
- Logfile 8
- Changelog 98
- Code 6
- Inline-Quote 41
- Fullquote 143
- Anrede 238
- Abschied 322
- Autor_In 444
- Folgebeitrag 168
- Start 275

Privat 23
Entschuldigung 22
Anerkennung
- Anerk_speziell 47
- Anerk_alle 75

Fehler
- Fe_eigene 10
- Fe_speziell 6

Emotionen
- Em_Misc 5
- Em_negativ 2
- Em_positiv 60

Rolle 9
Name 53
Handlungs_Typ 264
- Ha_T_Translation 12
- Ha_T_Ext_SW 3
- Ha_T_Patch 10
- Ha_T_Mod_non_Code 17
- Ha_T_Kommu 8
- Ha_T_Testing 32
- Ha_T_Review 10

Kurzzusammenfassung

Fokus dieser Forschungsarbeit ist das kooperative und kollaborative Lernen bei der gemeinschaftlichen Softwareentwicklung. Dabei werden selbstorganisierte Lernnetzwerke betrachtet, die vorwiegend über das Internet und in ihrer Freizeit gemeinsam Freie/Libre beziehungsweise Open-Source Software (FLOSS) entwickeln. Die Akteur:innen bauen durch informelle Lernprozesse die dafür notwendigen Kompetenzen auf und unterstützen sich dabei gegenseitig. Dies passiert ohne Steuerung durch professionelle Lehrende und ohne vorbereitete Lehrpläne. Die Lernenden organisieren dabei im gegenseitigen Austausch ihre eigenen raum- und zeitdistanten E-Learning-Arrangements.

Anhand qualitativer und quantitativer Daten werden der Kompetenzaufbau und dessen Rahmenbedingungen von Softwareentwickler:innen durch ein gemeinnütziges Engagement in FLOSS-Projekten beschrieben. Diese Form des informellen Lernens in internetbasierten Lernnetzwerken war bislang kaum erforscht, obwohl informelles Lernen in allen Situationen, in denen Gruppen von Personen im gegenseitigen Austausch miteinander ein gemeinsames Ziel verfolgen, eine wichtige Rolle spielt. Zu diesem Zweck wurde die gemeinschaftliche Kommunikation über zwei öffentliche Mailingslisten eines FLOSS-Projektes über mehrere Jahre hinweg ausgewertet und mit den Beiträgen der Entwickler:innen auf der Versionsverwaltung in Beziehung gesetzt. Weiterhin erfolgten 14 Expert:inneninterviews mit FLOSS-Entwickler:innen aus unterschiedlichen Projekten. Dadurch lässt sich das informelle Lernen in internetgestützten Lernnetzwerken mit allen relevanten Einflussgrößen darstellen und in ihrer Bedeutung für die befragten Entwickler:innen einschätzen.

Ein besonderer Forschungsschwerpunkt ist das Lernen in einem sozialen Kontext. Ziel ist die Darstellung von Einflussfaktoren anderer Lernender auf das individuelle Lernen einzelner Entwickler:innen. Als Analyseraster fungiert hierbei die Theorie der Communities of Practice. Mittels dieser Theorie wurden die Austauschprozesse innerhalb einer Entwicklungsgemeinschaft betrachtet. Ergänzt wird dieser Ansatz durch wichtige Merkmale des individuellen Lernens wie Situiertheit und Interessen. Dadurch lässt sich eine umfassende lerntheoretische Betrachtung des individuellen Kompetenzaufbaus durch eine Beteiligung an solchen Lernnetzwerken durchführen. Es erfolgt die Beschreibung von Unterschieden im Aufbau von Teilkompetenzen in unterschiedlichen Kompetenzbereichen. Weiterhin wird eine Typologisierung von FLOSS-Entwickler:innen vorgestellt. Eine Typologie aus vier Typen, bestehend aus professionellen Programmierer:innen, Maintainer:innen, Gemeinschaftsorientierten und den sozialen Unterstützer:innen zeigt die charakteristischen Merkmalskombinationen, an denen die Rahmenbedingungen des informellen Lernens demonstriert werden.

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Dissertation selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Wustermark, den 23.05.2021

Klaus Muttray