Freie Universität Berlin

Bachelorarbeit am Institut für Informatik der Freien Universität Berlin

Arbeitsgruppe Software Engineering

Zugänglichkeit von Open-Source-Softwareprojekten - Herausforderungen beim Onboarding und Pull Request

Berthold Bußkamp

Matrikelnummer: 5236079

berthob98@inf.fu-berlin.de

Betreuer/in:	Prof. Dr. Lutz Prechelt
Eingereicht bei:	Prof. Dr. Lutz Prechelt
Zweitgutachter/in:	Prof. Dr. Claudia Müller-Birn

Berlin, 31. März 2023

Zusammenfassung

Hintergrund - Elementarer Bestandteil von Open-Source-Softwareprojekten ist die gemeinsame Entwicklungsarbeit an einem geteilten Quellcode. Die Integration neuer Teilnehmender in die Entwicklungsarbeit, das sogenannte Onboarding, ist ein komplexer Prozess und kann von diversen Faktoren erschwert werden. Einen entscheidenden Kipppunkt im Onboarding stellt der Begutachtungsprozess im Rahmen eines sogenannten Pull Requests dar.

Ziel - Ziel dieser Arbeit ist es, Herausforderungen in den Pull Requests von Neueinsteigenden herauszuarbeiten und eine Theorie der zugrundeliegenden Ursachen zu erstellen.

Methoden - In der vorliegenden Arbeit wird ein zweiteiliger Forschungsansatz auf Grundlage qualitativer Daten verfolgt. Den ersten Teil stellt eine eigene Beteiligung des Autors an einem Open-Source-Softwareprojekt im Rahmen einer Fallstudie dar. Den zweiten Teil bildet die Analyse einer breiteren empirischen Datenbasis, die mit den Methoden der Grounded Theory analysiert wird. Im Rahmen der Grounded Theory Methodology wurden Pull Requests von Einsteigenden in vier verschiedenen Open-Source-Softwareprojekten untersucht.

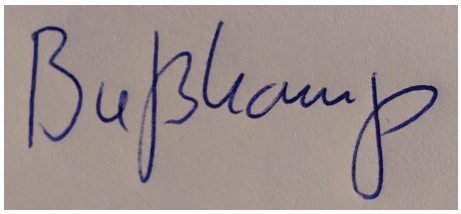
Ergebnisse - Das Ergebnis dieser Untersuchungen ist eine Reihe von Konzepten, die Herausforderungen in den Pull Requests von Neueinsteigenden darstellen. Diese Konzepte sind Teil eines theoretischen Modells, das ihre Zusammenhänge anhand von drei abstrakten Kernkonzepten erklärt.

Schlussfolgerungen - Der Vorgang des Onboardings wird durch Phänomene beeinträchtigt, die von beiden beteiligten Seiten – den Neueinsteigenden und den bereits Teilnehmenden - beeinflusst wird. Auf der Seite der Neueinsteigenden sind es oftmals unzureichende projektspezifische Kenntnisse, die eine Hürde beim Onboarding darstellen. Auf Seiten der bereits integrierten Projekteilnehmenden wird das projektspezifische Zeitmanagement immer wieder zu einer Herausforderung für das Onboarding der Neueinsteigenden.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

31. März 2023



Berthold Bußkamp

Inhaltsverzeichnis

1 Einleitung

Elementarer Bestandteil von Open-Source-Softwareprojekten (OSS-Projekten) ist die gemeinsame Entwicklungsarbeit an einem geteilten Quellcode. Eine aktive Community hat sich als einer der wesentlichen Vorteile erfolgreicher Open-Source-Softwareprojekte gegenüber anderen Vorgehensmodellen erwiesen [1]. Fehler in der Software und neue Funktionalitäten können schneller erkannt und umgesetzt werden [54].

Die Integration neuer Teilnehmender in die Entwicklungsarbeit, das sogenannte Onboarding, ist ein komplexer Prozess und kann von diversen Faktoren erschwert werden [62] [61]. Einsteigende müssen sich mit dem System der Codeverwaltung vertraut machen, die Projektspezifika verstehen und in eine neue Codebasis einarbeiten. Probleme können u. a. auftreten, wenn Informationen fehlen oder kommunikative Prozesse nicht funktionieren [62] [61]. Während vertraute Entwickelnde des Projektes, teilweise ohne auf andere Teilnehmende angewiesen zu sein, den gemeinsamen Quellcode verändern dürfen, sind Einsteigende meist auf die Begutachtung und Genehmigung durch etablierte Teilnehmende angewiesen. Dieser Begutachtungsprozess im Rahmen eines sogenannten Pull Requests (PR) stellt einen der entscheidenden Kipppunkte im Onboarding von potentiellen neuen Entwickelnden dar. Zum Zeitpunkt eines Pull Requests haben die Einsteigenden meist schon einen gewissen Entwicklungsaufwand erbracht, ein Scheitern an diesem Punkt kann besonders frustrierend sein:

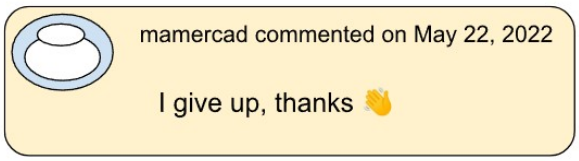


Abbildung 1: Kommentar unter einem gescheiterten Pull Request [44]

Auch für laufende OSS-Projekte sind gescheiterte PRs nicht gern gesehen, da der Begutachtungsprozess Projektressourcen konsumiert, ohne einen Mehrwert für das Projekt zu liefern. Die vorliegende Arbeit untersucht PR-Prozesse in OSS-Projekten als bedeutenden Aspekt des Onboardings von Einsteigenden mit dem Ziel, typische Probleme zu identifizieren und darüber zur Optimierung dieser Prozesse beizutragen.

2 Begriffsklärungen

Im Folgenden werden einige zentraler Begriffe definiert, um eine verständliche und kongruente Terminologie im Rahmen dieser Arbeit zu etablieren. Die Definitionen orientieren sich an dem Wortgebrauch anderer wissenschaftlicher Arbeiten.

- **Onboarding:** Einstiegsprozess einer zuvor unbeteiligten Person in die Entwicklungsarbeit eines Projekts
- **Teilnehmende:** Personen, die in Kontakt mit einem Projekt getreten sind
- **Einsteigende:** Personen, die in die Entwicklungsarbeit eines Projekts einsteigen (engl. Newcomers)

- **Entwickelnde:** Personen, die Quellcode zu einem Projekt beigetragen haben (engl. Developers)
- **Reviewer:** Personen, die einen Pull Request begutachten (engl. Reviewers)
- **Maintainer:** Reviewer mit Schreibrechten für den Quellcode
- **Patch:** Eine Veränderung des Quellcodes innerhalb einer Versionsverwaltung
- **Pull Request:** Anfrage zur Annahme eines Patches in den Quellcode eines Projekts. Teil eines Pull Requests können Kommentare und ein Begutachtungsprozess sein (synonym Merge Request, Change Request)

3 Stand der Forschung

Die Darstellung zum Stand der Forschung basiert auf ausgewählten Forschungsarbeiten und gliedert sich in zwei Teile.

- Im ersten Teil (Kap. 3.1, Kap 3.2) werden allgemeine Arbeiten zur Definition des Begriffs Open Source und den Teilnehmenden an OSS-Projekten vorgestellt. Anhand dieser Arbeiten wird der Rahmen der zu betrachtenden Projekte präzisiert.
- Im zweiten Teil (Kap. 3.3) werden Arbeiten mit direktem Bezug zum Kernthema der Bachelorarbeit vorgestellt, die zwei thematische Aspekte berücksichtigen. Zum einen Arbeiten, die das Onboarding in OSS-Projekten untersuchen und zum anderen Arbeiten, die den Verlauf von PRs untersuchen. Der Untersuchungsgegenstand der Bachelorarbeit befindet sich in der Schnittmenge beider Aspekte.

3.1 Open-Source-Projekte

Nach den Definitionen der Open Source Initiative (OSI) zeichnet sich Open-Source-Software hauptsächlich über die freie Verfügbarkeit, Modifizierbarkeit, Nutzbarkeit und die freie Weitergabe des Quellcodes aus [38]. Die Bezeichnung Open Source beschreibt also vornehmlich die Lizenzierung von Software. Die Art und Weise der Entwicklung ist dadurch nicht weitergehend bestimmt und erlaubt somit die verschiedensten Projektformen unter dem Sammelbegriff. Aus der Offenheit dieses Begriffs resultiert der Bedarf nach weiteren Differenzierungen. So haben einige Arbeiten ein weitergehendes Verständnis des Begriffs „Open Source“ geprägt.

“Basar” Stil

Einen viel diskutierten Grundstein der Idee, eines über die Lizenzierung hinaus reichenden Verständnis des Begriffs Open Source, lieferte Eric S. Raymonds erstmals 1997 veröffentlichter Essay, “The Cathedral and the Bazaar” [55]. In diesem Werk stellt er zwei fundamental unterschiedliche Entwicklungsstile gegenüber. Zum einen

die "Kathedrale" und zum anderen den "Basar". Ein Unterschied, den Raymond zwischen den beiden Stilen ausmacht, ist die Einbeziehung der Nutzerschaft in den Entwicklungsprozess. Während Projekte im "Kathedralen"-Stil die Softwareentwicklung einer kleinen Gruppe an Beteiligten vorbehalten und ihren Quellcode nur mit der Fertigstellung expliziter Veröffentlichungen bereitstellen, ist im "Basar"-Stil der Quellcode jeder Person, jederzeit zugänglich und ermöglicht somit jedem die Beteiligung am Entwicklungsprozess. Einen Vorteil des "Basar"-Stils sieht Raymond in der größeren Anzahl der Projektbeteiligen, die u. a. eine bessere Fehlersuche ermöglichen. Neben der Lizenzierung wird also der Entwicklungsstil zu einem Merkmal verschiedener Projektarten gemacht, die sich innerhalb des Begriffs Open Source unterscheiden.

Open Source Entwicklungsstil

Brown und Booch beschreiben einen Entwicklungsstil, welcher sich in der Open-Source-Software-Bewegung entwickelt hat, in welchem eine große Gemeinschaft in regem Austausch eine Software entwickelt[5]. Die Kommunikation findet dabei für jeden offen einsehbar statt und das Projekt ist grundsätzlich für die Beteiligung aller Personen offen. Die Beschreibung dieses Entwicklungsstil grenzt sich von Raymonds "Basar"-Stil dadurch ab, dass sie die Autorität und Entscheidungsmacht in allen interessanten OSS-Projekten bei einer Art gütigem Diktator sieht. Dieser übt die Autorität über das Design der Software aus und bietet den Projekten damit eine Koordination.

„Community managed" Führungsstil

O'Mahony untersucht explizit die Frage des Führungsstils von Open-Source-Projekten und definiert einen Stil, der als "community managed" bezeichnet wird [48]. In "community managed" Projekten gelten demnach die folgenden 5 Grundsätze:

- Unabhängigkeit - Die Führung des Projekts ist unabhängig von Sponsoren
- Pluralismus - Die Führung bewahrt sich unterschiedliche Ansätze, Methoden und Sichtweisen
- Repräsentation - Sich beteiligende Mitglieder werden in gemeinschaftlichen Entscheidungen repräsentiert
- Dezentrale Entscheidungsfindung - Entscheidungen werden zu einem gewissen Teil dezentral getroffen
- Autonome Teilnahme - Die Führung des Projekts heißt neue Mitglieder willkommen und ermöglicht es ihnen, sich nach ihren eigenen Vorstellungen zu beteiligen

Neben einem für alle Mitglieder offenem Entwicklungsstil wird also ein Führungsstil definiert, der sich durch einen offenen Entscheidungsfindungsprozess auszeichnet.

Software Project Governance Framework

Capra & Wasserman entwickelten das „Software Project Governance Framework“ (SPGF) [11]. Das SPGF ermöglicht eine Einordnung von Softwareprojekten auf einem Spektrum zwischen Closed- und Open-Source anhand des Entwicklungsstils und des Führungsstils der Projekte. Aus einer Umfrage unter 70 leitenden Personen von Closed- und Open-Source-Softwareprojekten haben Capra & Wasserman vier Charakteristika herausgearbeitet, die das Management von Softwareprojekten in Hinblick auf den Open Source Begriff charakterisieren. Zu diesen vier Charakteristika wurde ferner ein Evaluationsmodell entwickelt, das eine schnell erfassbare Einordnung von Softwareprojekten ermöglicht.

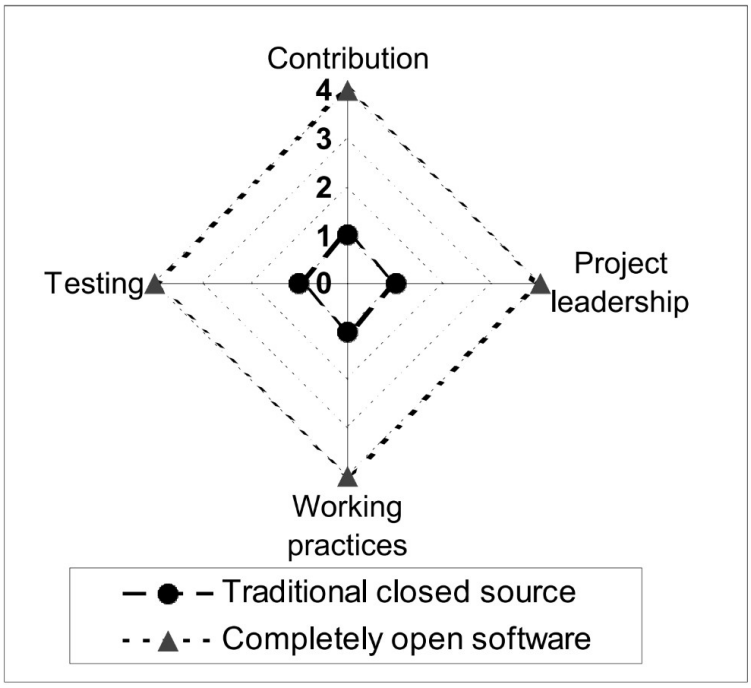


Abbildung 2: Eine grafische Darstellung der Projektmanagement Dimensionen [11]

Quelle: Capra & Wasserman, 2008 [11]

Das aufgestellte Modell (Abb. 2) verortet OSS-Projekte in einem Spektrum zwischen Closed- und Open-Source-Ansätzen. Die Projekte werden in vier verschiedenen Kategorien auf einer Skala von 1 bis 4 bewertet. Je niedriger der Wert ist, desto näher befindet sich das Projekt an einem Closed-Source-Ansatz.

Die vorgestellten Interpretationen des Open Source Begriffs charakterisieren ein zweipoliges Spektrum, in dem auf der einen Seite völlig offene, dezentrale Projekte stehen und auf der anderen Seite komplett geschlossene Projekte. In der Praxis finden sich entlang dieses Spektrums viele verschiedene Modelle, die die verschiedenen Aspekte unterschiedlich offen oder geschlossen umsetzen.

Aus dem dargestellten Spektrum an Vorgehensmodellen sind für die weiteren Betrachtungen in dieser Bachelorarbeit folgende Aspekte bedeutsam: Eine Open-Source-Lizenzierung und ein Entwicklungsstil, der für jeden offen und einsehbar ist, werden

als obligatorische Randbedingung vorausgesetzt. Der Führungsstil wird als ein variables Element betrachtet, in dem sich die untersuchten OSS-Projekte durchaus unterscheiden können.

3.2 Teilnehmende

Untersuchungen über die Teilnehmenden von OSS-Projekten beschäftigen sich mit einer Vielzahl von Eigenschaften der Beteiligten [14]. Um den Umfang der Bachelorarbeit in einem angemessenen Rahmen zu halten, wird die Vorstellung des Stands der Forschung zu den individuellen Eigenschaften von Entwickelnden auf 4 Merkmale fokussiert. Das Thema Motivation bedarf einer gesonderten Betrachtung. Doch zunächst werden die drei Eigenschaften dargestellt, die von Schweik & English als essenzielle Eigenschaften herausgearbeitet werden [12, S. 46]:

- Projekterfahrung
- Technische Fähigkeiten
- Bezahlung

Zu allen drei Eigenschaften zeigen verschiedene Arbeiten Bezüge zum Onboarding in OSS-Projekten auf.

Projekterfahrung

Aufgrund der kurzen Projektzugehörigkeit ist die Projekterfahrung von Einsteigenden oft gering und beschränkt sich auf Nutzererfahrungen mit der Software. Schweik & English beschreiben diese Eigenschaft als ein Spektrum von einsteigenden bis sehr erfahrenen Entwickelnden [12, S. 46]. Igor Steinmacher et al. beobachten fehlende Expertise in der Domäne und fehlende Expertise über Projektpraktiken als zwei Barrieren im Onboarding von Neueinsteigenden [61].

Technischen Fähigkeiten

Während die kurze Projektzugehörigkeit bei Einsteigenden eine geringe Projekterfahrung impliziert, können sich ihre technischen Fähigkeiten stark unterscheiden [12, S. 46]. Eine Umfrage von Ghosh et al. aus dem Jahr 2002 ergab eine folgende Verteilung der Bildungsabschlüsse: 9% mit einem Doktortitel, 28% mit einem Masterabschluss, 33% mit einem Bachelorabschluss und 30% ohne Hochschulabschluss [14]. Auf die Frage nach ihrem professionellen Hintergrund gaben 63% an, sich beruflich mit IT zu beschäftigen, 11,1% gaben an außerhalb der IT beschäftigt zu sein und 20,9% waren Studierende (15,8% Studierende im Bereich IT, 5,1% Studierende anderer Fächer) [14]. Igor Steinmacher et al. beschreiben eine Vielzahl von technischen Hürden im Onboarding Prozess [61] [62]. Diese betreffen unter anderem Technologien, Programmiersprachen, Versionskontrollsysteme oder das Code-Verständnis. Darüber hinaus haben Bird et al. gezeigt, dass die Darstellung eigener technischer Fähigkeiten die Wahrscheinlichkeit für einen erfolgreichen Einstieg in OSS-Projekte erhöht [3].

Bezahlung

Verschiedene Studien kamen auf unterschiedliche Verteilungen zwischen bezahlten und unbezahlten Entwickelnden im Bereich der OSS-Projekte. In einer Studie von Gerosa at al. aus dem Jahr 2021 zur Motivation von Entwickelnden gaben 26% der Befragten an Geld für ihre Beteiligung an OSS-Projekten erhalten zu haben [13]. Vorhergehende Umfragen kamen auf andere Zahlen. So berichten Lakhani & Wolf von ca. 40% bezahlter Entwickelnder [41]. Dabei spielt allerdings die Fragestellung eine entscheidende Rolle. So gaben nur 13% an, direkt für ihre Open Source Arbeit bezahlt worden zu sein, aber fast 38% berichteten, dass ihre Vorgesetzten die Nutzung der Arbeitszeit für Open-Source-Softwareprojekte erlaubten. Weitere Studien zu der Frage mit unterschiedlichen Ansätzen stammen von Riehle et al. [56] und Zhang et al. [67]. Die beiden Studien verweisen darauf, dass der Anteil der kommerziellen Beteiligung je nach OSS-Projekt variiert [56] [67]. Igor Steinmacher et al. geben zu bedenken, dass es möglich ist, dass die veränderte Motivation von bezahlten Einsteigenden die Barrieren im Onboarding für sie abschwächen [61].

Motivation

Die Bereitschaft zur unbezahlten Arbeit wirft die Frage nach der Motivation der Entwickelnden auf [12, S. 46]. So beschäftigen sich viele Arbeiten mit der Frage der Motivation von OSS-Entwickelnden und welche Rolle eine Bezahlung dabei spielt [57] [63] [14] [13] [41]. Typische in der Literatur diskutierte Motivationen sind eine Bezahlung der Tätigkeiten, die Möglichkeit etwas zu lernen, die Freude an der Kreativität der Arbeit oder die Verbesserung der Karrierechancen. Die Arbeiten unterscheiden in der Regel intrinsische und extrinsische Motivationen, wenngleich diese Kategorien in den verschiedenen Arbeiten leicht unterschiedlich behandelt werden. Drei der Arbeiten berichten von einer Internalisierung einzelner extrinsischer Motivationen [57] [13] [41].

Zusammenfassend lässt sich sagen, dass keine einzelne Motivation in der allgemeinen OSS-Landschaft vorherrschend ist. Stattdessen werden die Entwickelnden jeweils unterschiedlich, teilweise von mehreren Motivationen gleichzeitig animiert [41]. Dabei gibt eine Mehrheit an, nicht von Geld motiviert zu werden [41] [14] [13]. Teilnehmende die von Geld motiviert werden, investieren jeweils mehr Zeit in die Projekte [41] und liefern mehr Beiträge [57].

3.3 Arbeiten zum Untersuchungsgegenstand

Nach der Vorstellung des allgemeinen Forschungsstands zu OSS-Projekten folgt im Weiteren die Betrachtung von Forschungsarbeiten, die einen direkten Bezug zu den in dieser Bachelorarbeit untersuchten Sachverhalten haben. Da diese Arbeit explizit den sich in Pull Requests abspielenden Teil des Onboardings untersucht, befindet sie sich in der Schnittmenge zwei verschiedener Felder der Literatur. Zum einen Arbeiten, die das Onboarding in OSS-Projekten untersuchen, zum anderen Arbeiten, die den Ablauf von Pull Requests betrachten. Entlang dieser beiden Forschungsfelder orientiert sich folgende Zusammenfassung der Literatur.

3.3.1 Onboarding

Ein weitverbreitetes Verständnis des Onboarding Prozesses repräsentiert das **Onion Model** (Abb. 3), das die verschiedenen Stufen eines Einstiegs in OSS-Projekte von außen nach innen darstellt [60]. Durch Beiträge zu den Projekten rücken die Einsteigenden dabei immer weiter in den Kern des Projekts vor [64] [60]. Einige Arbeiten haben Limitierungen dieses Modells herausgearbeitet [49] [36]. Laut Herraiz et al. folgen nicht alle Einsteigenden der Reihenfolge der Schritte des Modells. So haben beispielsweise die meisten (52,5%) der untersuchten Einsteigenden eine Änderung zu einem Projekt beigetragen, bevor sie dem Projekt einen Fehler gemeldet haben (s. g. Bug Report) [36]. Außerdem folgten in einer Gegenüberstellung von bezahlten und nicht bezahlten Einsteigenden nur unbezahlte Einsteigende dem Onion Modell, während bezahlte Einsteigende dem Modell nicht folgten und schneller den Kern des Projekts erreichten. Steinmacher et al. merken an, dass nicht alle Einsteigenden überhaupt beabsichtigen, Kernentwickelnde zu werden [60]. Daten dazu, wie viele Beteiligte an OSS-Projekten einen tiefen Einstieg in die Projekte beabsichtigen, fehlen.

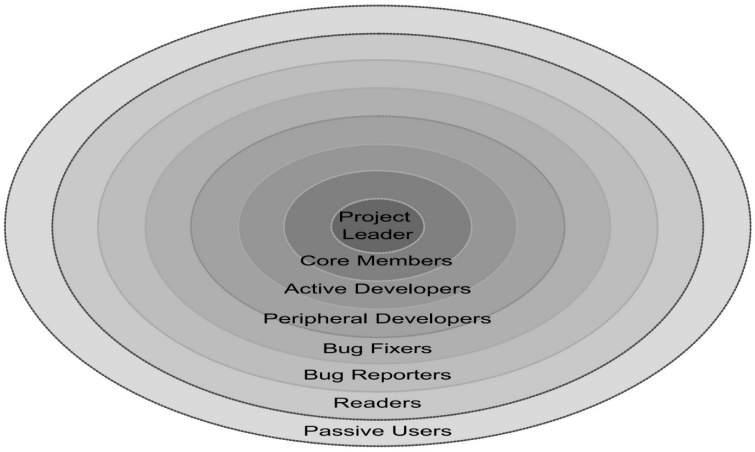


Abbildung 3: Die Struktur einer OSS-Community [47]

Quelle: Nakakoji et al. 2003 [47]

Im Folgenden werden zwei Arbeiten von Steinmacher et al. zum Onboarding in OSS-Projekten vorgestellt, da sie die ausführlichsten Arbeiten auf diesem Gebiet sind und auch frühere Arbeiten anderer Autor*innen mit einbeziehen. Das **Developer Joining Modell** beschreibt das Onboarding von Einsteigenden als ein Wirken verschiedener Kräfte, das **Barrieren Modell** beschreibt die Barrieren, welche Einsteigende teilweise während des Onboardings überwinden müssen [60]. Das Developer Joining Modell besteht aus zwei Elementen, zum einen die Phasen des Onboardings und zum anderen die vier Kräfte, die einen Einstieg positiv bzw. negativ beeinflussen (Abbildung 4).

Die Motivation (vgl. Kap 3.2) wirkt in dem Modell während des gesamten Onboarding Prozesses [60]. Nimmt die Motivation ab, kann dies zum Ausstieg aus einem Projekt führen. Die Attraktivität ergibt sich aus dem Handeln und den Eigenschaften eines Projekts. Zusammen mit der Motivation ist sie die Kraft, welche Entwickelnde

zu einem konkreten Projekt treibt. Die erhaltenden Kräfte sorgen dafür, dass neue und etablierte Entwickelnde vermehrt Beiträge zu den Projekten liefern. Für neue Entwickelnde können dies bspw. Hilfestellungen sein. Die hinderlichen Faktoren sind Barrieren, die das Onboarding negativ beeinflussen. Diese Hindernisse stören sowohl Entwickelnde, die Mitglied eines Projekts werden wollen, als auch Personen, welche nur einen einmaligen Beitrag zu einem Projekt liefern möchten. Sie können die Beiträge verzögern oder Entwickelnde zum Aufgeben bringen.

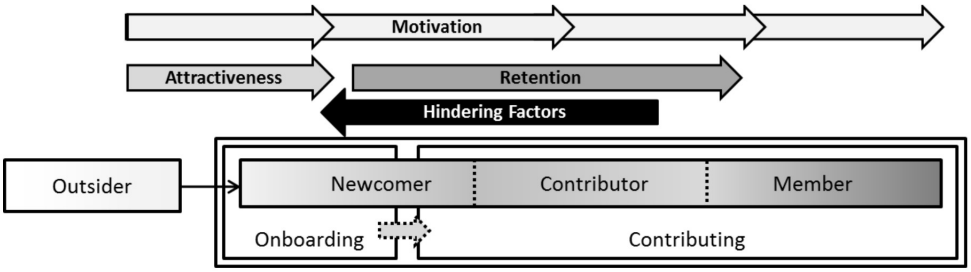


Abbildung 4: Die Stufen und Kräfte des Einstiegsprozesses [60]

Quelle: Steinmacher et al. 2014 [60]

Steinmacher et al. identifizieren mittels Systematic Literature Review, Umfragen, offen gestellten Fragen und Interviews insgesamt 58 verschiedene Hindernisse aus sechs verschiedenen Kategorien, die das Onboarding von Einsteigenden in OSS-Projekten stören können [62]. Die folgende Grafik der Arbeit (Abb. 5) stellt das resultierende Barrieren Modell dar:

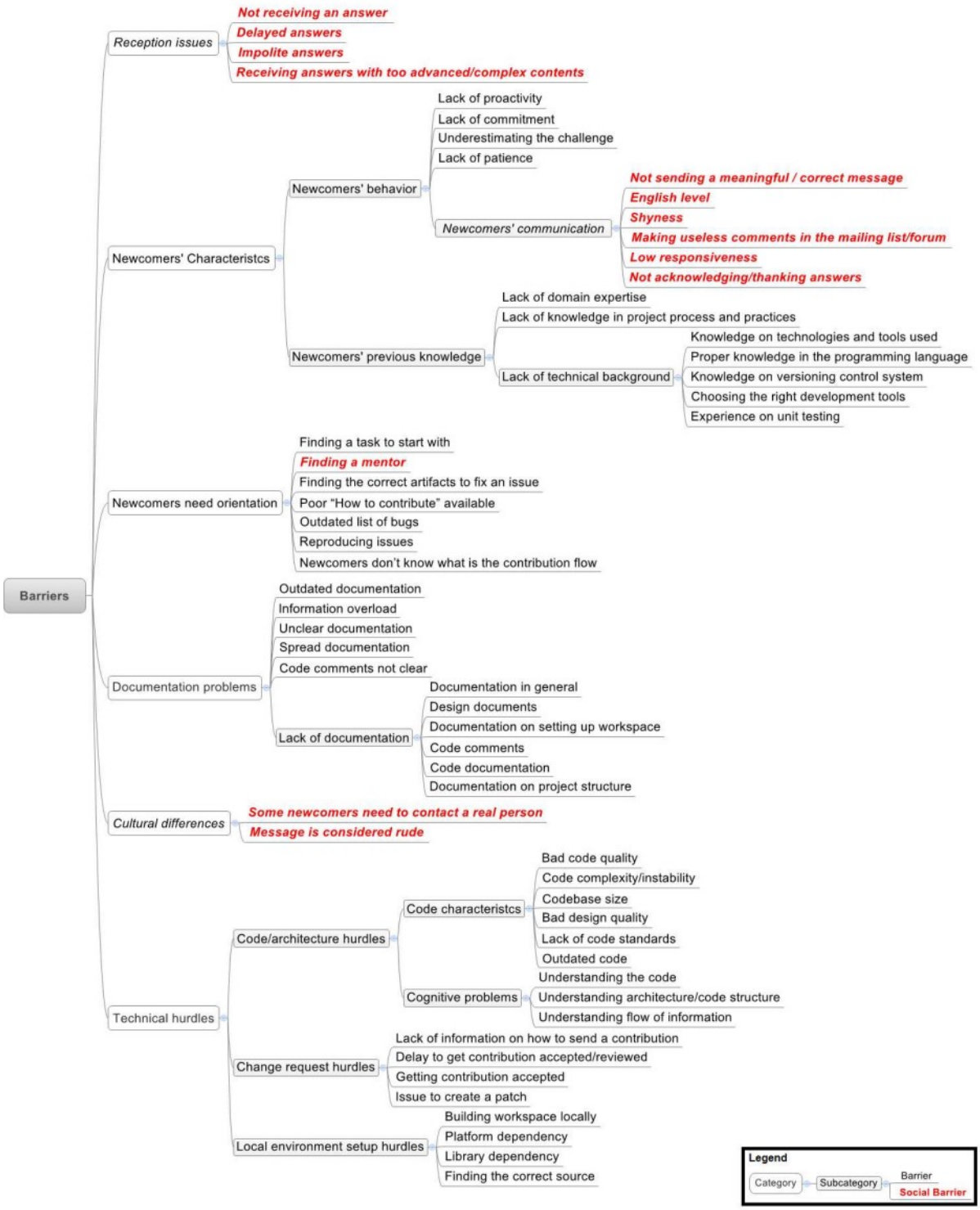


Abbildung 5: Modell der Barrieren für Neueinsteigende [62]

Quelle: Steinmacher et al. 2015 [62]

Im Folgenden werden die für diese Bachelorarbeit besonders bedeutenden Barrieren aus dem Modell hervorgehoben. Steinmacher et al. beschreiben als technische

Hürden u. a. kognitive Probleme der Einsteigenden im Verständnis des Codes, der Architektur und der Code Struktur [62] [61]. Andere beschriebene technische Hürden betreffen den Ablauf der Pull Requests (Change Requests). Dabei werden u. a. Probleme, einen Patch zu erstellen, Verzögerung der Begutachtung und ein Mangel an Informationen, wie beigetragen werden kann, genannt. Neben den technischen Hürden werden auch einige Charakteristika der Einsteigenden als Barrieren beschrieben. Besonders relevant für diese Bachelorarbeit sind dabei zum einen ein Mangel an Wissen über die Domäne und über Prozesse der Projekte und zum anderen ein Mangel an technischem Wissen, wie z.B. über das Versionskontrollsystem oder die Programmiersprache. Weitere relevante Hürden aus dem Modell betreffen das Antwortverhalten, dabei werden u. a. ausbleibende Antworten, verspätete Antworten und zu komplizierte Antworten beschrieben.

3.3.2 Pull Requests

Li et al. präsentieren ein simples Modell (Abb. 6) zum Verständnis des möglichen Verlaufs von Pull Requests [43]. Die handelnden Teilnehmenden des Begutachtungsprozesses sind dabei die **Autor*innen** (engl. oftmals Authors oder Contributors) und die **Reviewer** der Pull Requests. Die möglichen Ergebnisse des Prozesses sind eine Annahme oder Ablehnung der Pull Requests durch die Reviewer, sowie das Verweisen der Pull Requests seitens der Autor*innen [43].

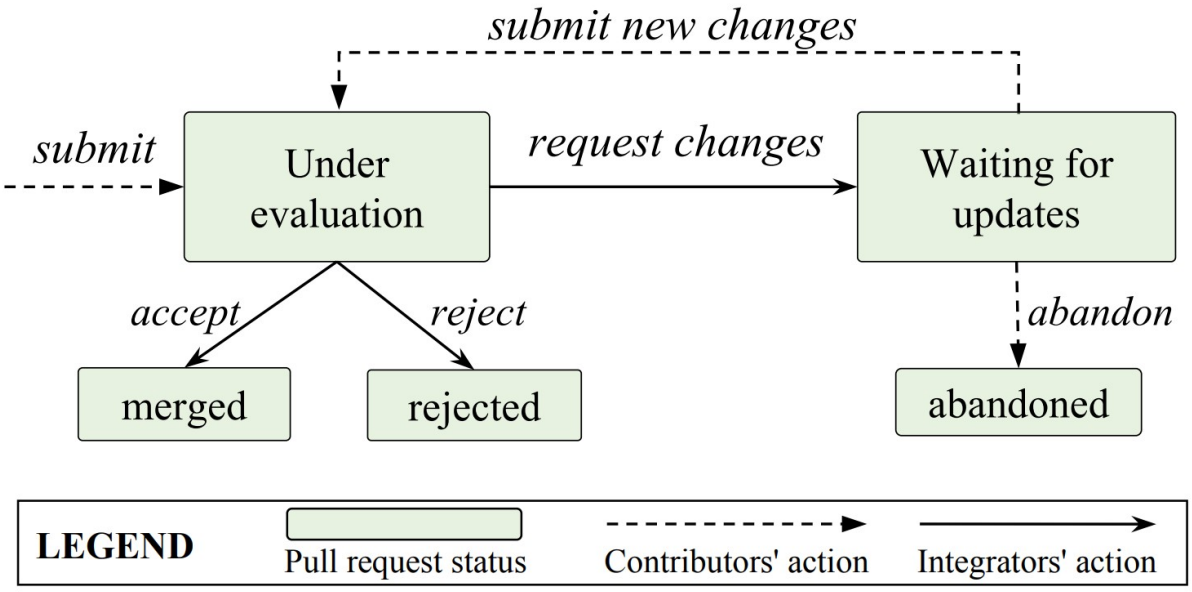


Abbildung 6: Begutachtungsprozess eines Pull Request [43]

Quelle: Li et al. 2021 [43]

Um einen Patch in den Quellcode zu integrieren, muss die Rolle des Reviewers in letzter Instanz von einer Person mit Schreibrechten auf den Quellcode ausgeübt werden (einem Maintainer). Die Rolle der Autor*innen hingegen kann sowohl von Personen mit Schreibrechten als auch von Personen ohne Schreibrechte ausgeführt werden.

Bei Autor*innen mit Schreibrechten dienen die Pull Requests dazu, anderen Personen die Möglichkeit der Diskussion des Codes zu bieten, während die Pull Requests für Autor*innen ohne Schreibrechte ein notwendiger Prozess sind. Eine Vielzahl von Arbeiten beschäftigt sich mit der Frage, welche Faktoren einen Einfluss auf die Annahmeentscheidung eines Pull Requests haben. Zhang et al. sammelten im Rahmen einer Systematic Literature Review diese Faktoren und evaluierten ihre Bedeutsamkeit anschließend in einem quantitativen Ansatz anhand von 3.347.937 Pull Requests aus 11.230 Projekten [66]. Ein bedeutender Faktor für die Annahmeentscheidung eines Pull Requests ist unter anderem die bisherige Erfahrung der Autor*innen. Wenn die/der Autor*in eines Pull Requests Core-Mitglied des Projekts ist oder wenn er/sie selbst vorher schon viele Pull Requests begutachtet (reviewed) hat, wurden seine/ihre Pull Requests statistisch häufiger angenommen. Außerdem wurden Pull Requests mit einer höheren Lebensdauer statistisch seltener angenommen[66]. Diese Korrelation zwischen der Lebenszeit und dem Ausgang eines Pull Requests sollte nicht mit einer Kausalität verwechselt werden [46]. Nur, weil Pull Requests, die länger dauern, statistisch eher scheitern, ist die Dauer nicht automatisch der Grund für das Scheitern. Es könnte beispielsweise auch einfach länger dauern, zu ermitteln, dass ein Pull Request ungeeignet ist. Soares et al. liefern initiale Hinweise für eine mögliche Richtung dieser Korrelation, welche darauf hindeuten, dass eine längere Dauer von Pull Requests zu Ablehnungen führt [46]. Darüber hinaus untersucht die Arbeit, welche Faktoren einen Einfluss auf die Lebensdauer von Pull Requests haben. Soares et al. zeigen, dass die Pull Requests von unerfahrenen Autor*innen länger dauern und dass eine fehlende Erfahrung mit dem Konzept Pull Requests einen Einfluss auf die Lebensdauer von Pull Requests hat [46]. Außerdem ist bei kommentierten Pull Requests die Geschwindigkeit der ersten Antwort des Reviewers ein bedeutender Faktor für die Dauer eines Pull Requests [65].

Das Antwortverhalten der Beteiligten eines Pull Requests ist ein wiederkehrendes Thema in vielen Arbeiten, die sich mit dem Verlauf von Pull Requests beschäftigen. In einer Umfrage von Gousios et al. unter Entwickelnden gaben mehr als 15% an, dass es schwer ist, eine zeitnahe Antwort auf einen Pull Request zu erhalten [33]. Einige gaben an, dass es bei ihnen in der Folge zu Frustration oder Stress kommt. Außerdem gaben Befragte an, dass schnellere Antworten die Barrieren für Einsteigende in das Projekt senken würden. Neben dem Antwortverhalten der Reviewer können auch verspätete oder ausbleibende Antworten der Autor*innen den Begutachtungsprozess negativ beeinflussen [34]. Ein besonderes Phänomen dabei sind hit-and-run Pull Requests, bei denen die Autor*innen nach dem initialen Erstellen eines Pull Request nicht mehr zu erreichen sind [34]. Besonders Pull Requests von Neueinsteigenden werden häufig abgebrochen [39]. Li et al. haben 12 Gründe für das Verlassen (unmittelbar oder im späteren Verlauf) von Pull Requests durch die Autor*innen in einer Umfrage ermittelt (vgl. Tab. 1) [43]. Die Gründe für das Verlassen der Pull Requests können inhaltlich, prozessbezogen oder persönlicher Art sein.

Tabelle 1: Gründe, aus denen Autor*innen Pull Requests abgebrochen haben [43]

Reason	Aspect	Votes(%)
$\mathcal{R}1$ Lack of answer from integrators	Process	42.4
$\mathcal{R}2$ Lack of time	Personal	36.6
$\mathcal{R}3$ Pull request is obsolete	Pull request	32.8
$\mathcal{R}4$ Not treated seriously by integrators	Process	28.6
$\mathcal{R}5$ Inadequate community demand	Pull request	22.8
$\mathcal{R}6$ Lack of interest	Personal	22.3
$\mathcal{R}7$ Lack of knowledge about the requested changes	Personal	20.0
$\mathcal{R}8$ Integrators do not reach a consensus	Process	17.7
$\mathcal{R}9$ Tasks of higher priority	Personal	15.8
$\mathcal{R}10$ It requires more effort than anticipated	Pull request	15.5
$\mathcal{R}11$ Disagree with integrators' opinions	Process	13.8
$\mathcal{R}12$ Tedious process of updating the pull request	Process	13.4
Other	- - -	1.2

Quelle: Li et al., 2021, S. 7 [43]

Teilnehmende der Umfrage machten auf den Einstieg bezogene Angaben [43]. Zum einen, dass Schwierigkeiten mit Git (Versionsverwaltung) besonders für Einsteigende überfordernd sein können, zum anderen, dass der Verlust des Interesses für Einsteigende aufgrund ihrer losen Bindung an das Thema eine besondere Relevanz habe.

Zusammenfassend lässt sich sagen, dass Einsteigenden in ein Projekt einer Vielzahl von potentiellen Barrieren gegenüberstehen. Diese ergeben sich u. a. aus ihren eigenen Charakteristika aber auch denen des Projekts. Ein wichtiger Teil des Einstiegs in OSS-Projekte ist das Beisteuern von Quellcode [36] [64] [60], ein elementarer Prozess dabei sind Pull Requests. Pull Requests von Einsteigenden in OSS-Projekte werden häufiger selbst aufgegeben als die Pull Requests von erfahrenen Entwickelnden [39]. Außerdem wirkt sich die Ablehnung von Pull Requests negativ auf die Bereitschaft von potentiellen Einsteigenden zu zukünftige Beteiligungen aus [42]. Die Einflussfaktoren auf die Annahmewahrscheinlichkeit wie auch die Gründe für das Aufgeben

von Pull Requests sind vielschichtig und betreffen u. a. sowohl Charakteristika der Autor*innen, als auch das Verhalten von Reviewern und Maintainern während des Ablaufs eines Pull Requests.

4 Methodische Vorgehensweise

Während eine Vielzahl von quantitativen Arbeiten zum Verlauf von Pull Requests existiert und einige von ihnen Aspekte des Onboardings berücksichtigen, existieren wenige qualitative Arbeiten zum Verlauf von Pull Requests. Vorliegende qualitative Arbeiten nutzen in der Regel Interviews als Datengrundlage und sind deshalb auf die Selbstwahrnehmung der Teilnehmenden angewiesen. Qualitative Arbeiten zum Verlauf von Pull Requests mit explizitem Fokus auf Einsteigende sind dem Autor nicht bekannt. In der vorliegenden Arbeit wird ein zweiteiliger Forschungsansatz auf Grundlage qualitativer Daten verfolgt. Den ersten Teil der Arbeit stellt eine eigene Beteiligung des Autors an einem OSS-Projekt im Rahmen einer **Fallstudie** dar. Den zweiten Teil der Arbeit bildet die Analyse einer breiteren empirischen Datenbasis, die dem Ansatz der **Grounded Theory Methodology** (GTM) folgt. Ziel der Herangehensweise ist die explorative Entdeckung von Hürden im Open-Source-Softwareprojekt-Onboarding-Prozess und eine Theoriebildung der zugrundeliegenden Ursachen.

4.1 Fallstudie Opencast

Durch die Integration einer Fallstudie soll die Bachelorarbeit nicht allein auf theoretischen Annahmen basieren, sondern es sollen auch konkrete, eigene Erfahrungen in einem Open-Source-Softwareprojekt in die Analysen und Auswertungen einfließen. Als Fallstudie wurde eine **teilnehmende Beobachtung** am Softwareprojekt Opencast gewählt. Bis zum Zeitpunkt der Fallstudie war dem Autor die Software Opencast nur aus der Anwendung bekannt. Mit Aufnahme der Fallstudie wurde der Prozess des Einstiegs in ein Open-Source-Projekt persönlich durchlaufen. Teil des Einstiegs war die Kontaktaufnahme mit dem Entwicklungsumfeld, die Einarbeitung in den Quellcode, der Entwurf und die Umsetzung eigener Patches und die anschließende Integration der Patches mittels Pull Requests. Opencast ist eine Open-Source-Software zur Videoaufnahme, -verwaltung und -verarbeitung, die für die Verwendung an Hochschulen entwickelt wurde [51]. Dozenten können damit über eine Website Videos für Studierende zur Verfügung stellen. Die serviceorientierte Architektur der Software kann über mehrere Server verteilt arbeiten, dabei übernehmen die verschiedenen Server unterschiedliche Rollen. Zu Beginn der Fallstudie wurden für die Auswahl der möglichen Anpassungen zwei Kernentwickler des Projektes kontaktiert. Einer der Entwickler bot eine Liste mit potentiellen Themen für wissenschaftliche Arbeiten an [40]. Ein Thema auf der Liste war die "Job dispatching infrastructure". Um einen besseren Eindruck über die gewünschten Anforderungen an die Infrastruktur zu erhalten, wurden offene Issues im Bug Tracking des Projektes gesammelt. Darüber hinaus wurden die geplanten Anpassungen in einigen Videokonferenzen mit Kernentwicklern des Projektes besprochen. Im Rahmen der Fallstudie entstanden insgesamt 5 Pull Requests mit einem Gesamtumfang von 430 neuen und 193 entfernten Zeilen Code. Der Großteil der Änderungen betrifft die "Job dispatching infrastructure", welche die

Verteilung der sogenannten Jobs auf die Services der Software umsetzt. Ein Pull Request führt beispielsweise eine neue Funktionalität ein, die es ermöglicht, für Jobs, die von Hardwarebeschleunigung profitieren, präferierte Server anzugeben. Die Idee der Funktionalität ist es, effektiv einzelne Server mit besonderer Hardware nutzen zu können. Diese [6] und zwei weitere Pull Requests [7] [9] wurden bis zum aktuellen Zeitpunkt angenommen. Für die Annahme der anderen beiden Pull Requests [8] [10] sind noch vereinzelte Verbesserungen notwendig. Für eine der beiden Pull Requests existiert bereits positives Feedback, das auf eine potentielle Annahme hindeutet [10]. Beobachtungen von Schwierigkeiten während des gesamten Verlaufs des Einstiegs wurden in Feldnotizen mit Hinweisen auf entstandene Dokumente festgehalten. Die Daten der teilnehmenden Beobachtung sind sehr spezifisch, da sie einen einzigen Fall behandeln, bieten dafür aber wichtige Einsichten in Aspekte des Onboardings, welche in der Außenansicht nicht unmittelbar erkennbar sind. Die Daten teilnehmender Beobachtungen können subjektiv sein und bedürfen daher einer Reflexion der Perspektive des Forschenden [35, S. 225]. Da der Prozess bis zur Annahme eines Pull Requests mehr Zeit in Anspruch nimmt, als es der zeitliche Rahmen einer Bachelorarbeit ermöglicht, liefen Teile der Fallstudie am Opencast Projekt der Anmeldung der Bachelorarbeit voraus. Um den empfohlenen Umfang einer Bachelorarbeit einzuhalten, wird auf eine Vorstellung der Ergebnisse der teilnehmenden Beobachtung am Opencast Projekt verzichtet. Die gesammelten praktischen Erfahrungen mit dem Onboarding sind dennoch äußerst hilfreich für das Verständnis der Datengrundlage im Zusammenhang mit der Grounded Theory Methodology in dieser Bachelorarbeit.

4.2 Empirische Datengrundlage

Die Datenbasis für die Grounded Theory Methode bilden die Pull Requests von vier verschiedenen Open-Source-Softwareprojekten auf der Webseite github.com. Eine bedeutende Abwägung in der Auswahl der Projekte war, wie unterschiedlich die Projekte sein sollen. Wählt man sich zu stark unterscheidende Projekte, gibt es kaum gemeinsame Beobachtungen, die auf alle Projekte zutreffen. Wählt man den Rahmen zu klein, sind die Beobachtungen sehr spezifisch und es fehlt ihnen möglicherweise an Relevanz für andere Projekte. Für die Projektauswahl war unter anderem die Größe der Projekte von Bedeutung. Sie sollten groß genug sein, um jeweils eine hinreichende Menge an Pull Request Daten von verschiedenen Autor*innen zu liefern. Andererseits sollte die Menge überschaubar bleiben, um einen Gesamteindruck der Neueinstiege in den jeweiligen Projekten mit angemessenem Aufwand zu ermöglichen. Anschließend an die, in der Fallstudie gesammelten praktischen Erfahrungen mit dem **Opencast** Projekt, liefert die Analyse der Gesamtheit der Pull Requests des Opencast Projekts das erste von vier Projekten für die Grounded Theory Methode. Die verbleibenden drei Projekte wurden wie folgt ausgewählt. Die Projekte **ILIAS** und **Sponge** wurden durch die Analyse öffentlicher GitHub-Daten ermittelt. Dabei wurden automatisiert Projekte ermittelt, die dem Opencast Softwareprojekt in den Metriken "Stars" (290 - 400) und "Forks" (200 - 1000) ähneln. Für eine Liste von über 900 ermittelten Repositories wurden u. a. die Daten, Anzahl der Pull Requests, Anzahl der Beteiligten mit mindestens einem Pull Request und Anzahl offener Issues erhoben, um einen Eindruck über den Entwicklungsumfang zu erhalten. Außerdem wurde für jedes Projekt

der Gini-Koeffizient der Pull Request Verteilung auf die Gruppe der Autor*innen ermittelt, um eine Kennziffer für die Ungleichverteilungen des Entwicklungsaufwands zu erhalten. Anhand dieser Metriken wurden die beiden Projekte ILIAS und Sponge selektiert. Ziel war es, Projekte zu finden, die vielversprechend im Hinblick auf eine gemeinschaftliche Arbeitsweise sind, da diese dem Untersuchungsgegenstand dieser Arbeit entspricht. Das Projekt **OhMyZSH** (OMZ) wurde nicht über das zuvor beschriebene Verfahren ermittelt, sondern bei der Websuche entdeckt, fällt aber wegen der starken Gleichverteilung der gestellten Pull Requests auf verschiedene Autor*innen auf und liefert damit eine interessante Arbeitsweise, die sich von denen der anderen Projekte unterscheidet. Neben der Existenz einer gemeinschaftlichen Arbeitsweise wurde bei der Auswahl der Projekte auch der Führungsstil berücksichtigt. Ziel war es, mögliche Unterschiede im Onboarding, die aus dem Führungsstil resultieren, zu ermitteln. Während die Führungsstruktur von Opencast und ILIAS viele Zugehörige externer Institutionen wie Hochschulen und privater Dienstleister beinhaltet [52] [45, Community > Liste der Mitglieder], geben die Teilnehmenden von Sponge und OMZ keine Zugehörigkeit zu externen Institutionen an und werden nur von Spenden, die das Projekt erhält, finanziert oder engagieren sich ohne finanziellen Ausgleich [59][2].

Die Datengrundlage der vier ausgewählten OSS-Projekte zur Auswertung von Pull Requests ist erheblich vielfältiger als die Datengrundlage der teilnehmenden Beobachtung, da sie das Onboarding vieler verschiedener Einsteigender dokumentiert. Die Informationen sind allerdings auf die Außenansicht der Phänomene beschränkt. Durch diese Außenansicht fällt es schwer, Phänomene, die sich durch das Ausbleiben von Handlungen darstellen, vollends nachzuvollziehen. Verlassen Autor*innen beispielsweise einen Pull Request, hinterlassen sie selten eine klare Angabe von Gründen. Die möglichen Gründe müssen im vorhergehenden Verlauf der Pull Request gesucht werden und sind nicht immer eindeutig. Durch die im Literaturteil dieser Arbeit beschriebenen vielseitigen Einflussfaktoren ist auch die Vergleichbarkeit von Pull Requests eingeschränkt. Pull Requests unterscheiden sich teilweise in einer Vielzahl dieser Faktoren. Vermutet man einen Unterschied in einem dieser Einflussfaktoren als Ursache für ein bestimmtes Phänomen, fällt es aufgrund der Vielzahl an Faktoren schwer, alle anderen Faktoren als Ursache auszuschließen.

Zusammenfassend kann festgestellt werden, dass die Auswahl der untersuchten Projekte weder zufällig noch repräsentativ erfolgte. Die Gesamtdatengrundlage der erfolgten Analysen basiert auf den vier Open-Source-Softwareprojekten:

- **Opencast**: Eine Videoplattform für akademische Institutionen [53]
- **ILIAS**: Eine e-Learning Plattform [37]
- **Sponge**: Ein Plugin Framework für das Videospiel Minecraft [58]
- **OhMyZSH** (OMZ): Eine Modifikations-Framework für die “zsh” Unix-Shell [50]

4.3 Grounded Theory Methodology

Die Grounded Theory Methodology (GTM) ist ein qualitativ interpretativer Ansatz aus der Sozialforschung mit dem Ziel der Entwicklung gegenstandsbezogener Theo-

rien [35, S. 97]. Das Vorgehen ist dabei ein Wechselspiel, in dem Episoden der Theoriebildung immer wieder Ausgangspunkt der wiederkehrenden Materialgewinnung und Analyse sind [35, S. 98]. Diese Arbeitsweise, während der Untersuchungen immer wieder neue Daten zu gewinnen, wird theoretisches Sampling genannt [35, S. 227]. Die anfänglich offene Forschungshaltung wird dabei im Verlauf der Untersuchungen zu einer Forschungsfrage ausdifferenziert [35, S. 98].

Der Ablauf der Grounded Theory wird anhand einer Reihe von Praktiken organisiert. Der Prozess beginnt mit einer ersten Anwendung des **offenen Kodierens**, in welcher auftretende Phänomene in den Ausgangsdaten anhand von Codes (Schlüsselwörtern) kodiert werden [35, S. 99]. In der Analysemethode des konstanten Vergleichens werden die Phänomene der Datengrundlage untereinander verglichen und zu hierarchischen Konzepten zusammengefasst [35, S. 98]. Im Prozess des **axialen Kodierens** werden diese Konzepte durch die Identifizierung von Beziehungen kategorisiert [35, S. 99]. In der finalen Praktik des **selektiven Kodierens** werden die Konzepte anhand von Kernkonzepten verbunden und strukturiert [35, S. 99]. Alle diese Praktiken können in der Arbeitsweise des **theoretischen Samplings** ein Anlass für erneute Materialgewinnung sein. [35, S. 99]

Im Rahmen dieser Arbeit wurden die Pull Requests anfänglich anhand zufällig ausgewählter Autor*innen gesammelt. Dabei wurden ihre ersten PRs in den Projekten untersucht und nur bei besonderem Interesse später folgende PRs untersucht. Im Verlauf dieser Arbeit wechselte das Vorgehen bei der Auswahl der PRs dann dazu, gezielt längere PRs auszuwählen, da diese häufig die ereignisreicheren Verläufe bieten. Danach wurden zur Auswahl weiterer PRs automatisiert Listen generiert, die alle PRs beinhalten, die mindestens drei Kommentare hatten und von Autor*innen mit weniger als 16 PRs kamen, um erneut die Datengrundlage entsprechend der Forschungsinteressen zu schärfen. Insgesamt wurden 214 PRs aus 4 verschiedenen Projekten untersucht, dabei wurden die PRs teilweise mehrfach codiert, um neue Erkenntnisse anzuwenden. Zur Organisation und Analyse der Daten wurde die Software MAXQDA verwendet. Die Strukturierung der Codes erlebt über den Verlauf der Untersuchung insgesamt drei große Umstellungen. Dabei wandelte sich die Sichtweise von einer anfänglichen Fokussierung der einzelnen (kommunikativen) Handlungen zu einer Sichtweise, in der die auftretenden Probleme der Hauptgegenstand der Untersuchung sind. Die finale Codestruktur ist auf oberster Ebene danach aufgeteilt, ob die Codes **Ursache**, **Thematik** oder **Umgangsweise** eines Problems darstellen. Eine vierte Oberkategorie bilden kommunikative **Probleme** und kommunikative **Hilfeleistungen**. Innerhalb der vier Oberkategorien werden weitere Unterkategorien unterschieden, beispielsweise ob die Thematik eines Problems technisch, projektspezifisch oder formell ist.

5 Ergebnisse

Im Kapitel zum GTM-Ansatz wurden die methodischen Grundlagen und Begrifflichkeiten zu den Analysen der qualitativen Daten aus den Pull Requests erläutert. Mit Blick auf das Thema der Bachelorarbeit konzentrierten sich die Analysen auf Phänomene in Pull Requests, bei denen der eingebrachte Quellcode nicht in die OSS-

Projekte übernommen wurde, obwohl die Vorschläge von den Reviewern nicht explizit abgelehnt wurden.

Im Ergebnis der Untersuchungen konnten die aufgenommen und im weiteren Text beispielhaft dargestellten Phänomene auf 11 Konzepte verteilt werden, die sich aufgrund bestehender Gemeinsamkeiten drei Kategorien zuordnen lassen:

- Defizitäres Antwortverhalten
  - Zeitmangel
  - Maintainer Bottleneck
  - Unmaintainable Code
  - Vergessene PRs
- Veränderte Randbedingungen
  - Merge Konflikte
  - Versionssprünge
  - Obsoleszenz
- Projektspezifische Unwissenheit
  - Projektstruktur
  - Projektverlauf
  - Nutzungsverhalten
  - Codestruktur

Im Weiteren werden zunächst die Konzepte in den drei Kategorien vorgestellt. Abschließend wird ein theoretisches Beziehungsmodell entwickelt, das anhand von Plausibilitätsargumenten mögliche Verbindungen zwischen den gebildeten Konzepten graphisch wiedergibt. Die drei gefundenen Kategorien werden durch drei abstrakte Kernkonzepte miteinander verbunden:

- Die **Lebensdauer** (bzw. die auftretenden Verzögerungen) eines Pull Requests von der Eröffnung bis zur Übernahme
- Die **Korrekturen**, die nach der Eröffnung des Pull Requests notwendig sind
- Der **Aufwand** (bzw. die entstehende Mehrarbeit), der nach der Eröffnung des Pull Requests eintritt und für die Übernahme notwendig ist

5.1 Defizitäres Antwortverhalten

Die erste der drei gebildeten Kategorien bündelt Konzepte, aufgrund derer sich insbesondere das Antwortverhalten der Beteiligten verzögert und ausbleibt.

Zeitmangel

Das Konzept Zeitmangel betrifft teilweise Reviewer und Autor*innen von Pull Requests. Trotz der vermeintlichen Trivialität des Konzeptes ist es bedeutend und dient als Grundlage für folgende Beobachtungen. Über dieses Konzept herrscht unter den Beteiligten ein ausgeprägtes Bewusstsein, so dass es häufig beobachtbar von den Teilnehmenden der Projekte angesprochen wird. Beobachtbar war das Konzept beispielsweise in einem Pull Request des OMZ Projektes, in dem der Autor von einem Maintainer auf die fehlende Zeit zur baldigen Begutachtung des PRs hingewiesen wurde. Der PR wurde vom Autor zuvor schon auf Wunsch des Maintainers korrigiert und von einer dritten Person positiv bewertet. Trotz Erinnerung durch den Autor blieb der Pull Request aufgrund von Zeitmangel aber seit über einem Jahr unbeantwortet (Abb. 7).

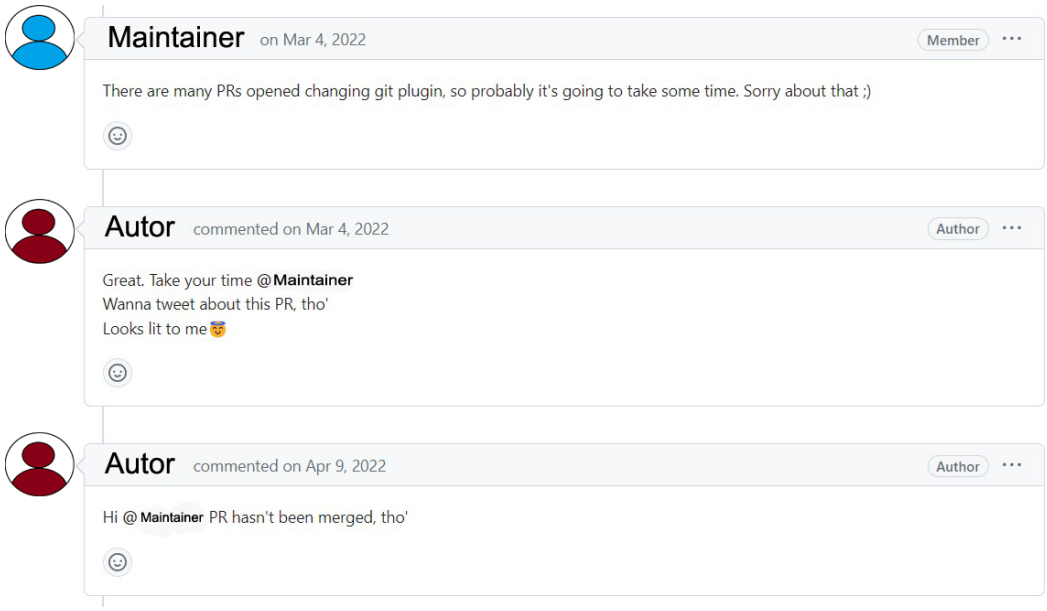


Abbildung 7: Auszug aus Pull Request OMZ #10748 [20]

In einem anderen PR des Projekts wird ebenfalls die Zeit als limitierender Faktor der Abarbeitung der offenen PRs benannt (Abb. 8).

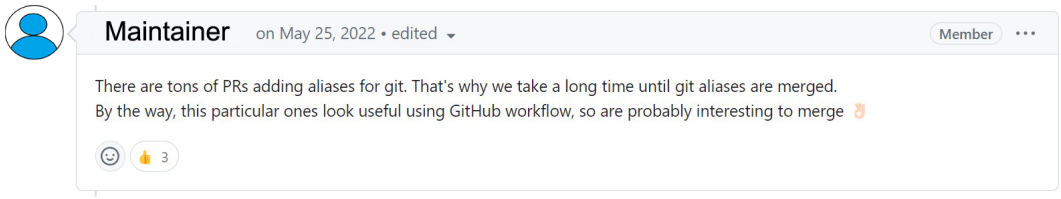


Abbildung 8: Auszug aus Pull Request OMZ #10261 [21]

Auch in anderen untersuchten Projekten wurde dieses Problem von den Maintainern benannt. Auf einer Konferenz des Opencast-Projekts 2021 beispielsweise benennt einer der Maintainer, dass zum Zeitpunkt des Vortrags ca. 90% der Begutachtungen

von nur zwei der Maintainern gemacht würden. Eine Folge dessen sei, dass Reviews teilweise ewig dauerten.

Neben den Reviewern weisen auch die Autor*innen wiederkehrend darauf hin, dass sie Pull Requests aufgrund von Zeitmangel nicht fortsetzen können. Ein Beispiel liefert ein Pull Request des OMZ Projekts. Obwohl ein Reviewer die PR als großartige und relevante Idee anerkennt, schließt der Autor die PR nach über einem Jahr, in dem er nicht die Zeit für die Vollendung des Pull Requests aufbringen konnte (Abbildung 9).

Abbildung 9: Auszug aus Pull Request OMZ #9795 [19]

Sowohl Autor*innen als auch Reviewer verweisen teilweise darauf, dass sie in ihrer Freizeit an den Projekten arbeiteten und daher nicht genügend Zeit aufwenden könnten. Darüber hinaus gibt es zahlreiche PRs, in denen nicht explizit Zeitmangel als Problem benannt wird, bei denen das langsame Antwortverhalten der Beteiligten jedoch auf ein solches Problem hindeutet. Das gilt besonders in Fällen, bei denen andere Gründe wie eine inhaltliche Überforderung nicht erkennbar und ein Vergessen der PR aufgrund von Erinnerungen unplausibel erscheinen. Die Beteiligten reagieren auf das verzögerte Antwortverhalten teilweise frustriert, teilweise verständnisvoll. Infolge der Frustration kommt es vereinzelt zu Abbrüchen der Pull Requests (Abb. 10).

Abbildung 10: Auszug aus Pull Request OMZ #9711 [18]

In Fällen, in denen nur einzelne Beteiligte unter Zeitmangel leiden, kann es vorkommen, dass andere Personen ihre Aufgaben übernehmen. (Abb. 11)

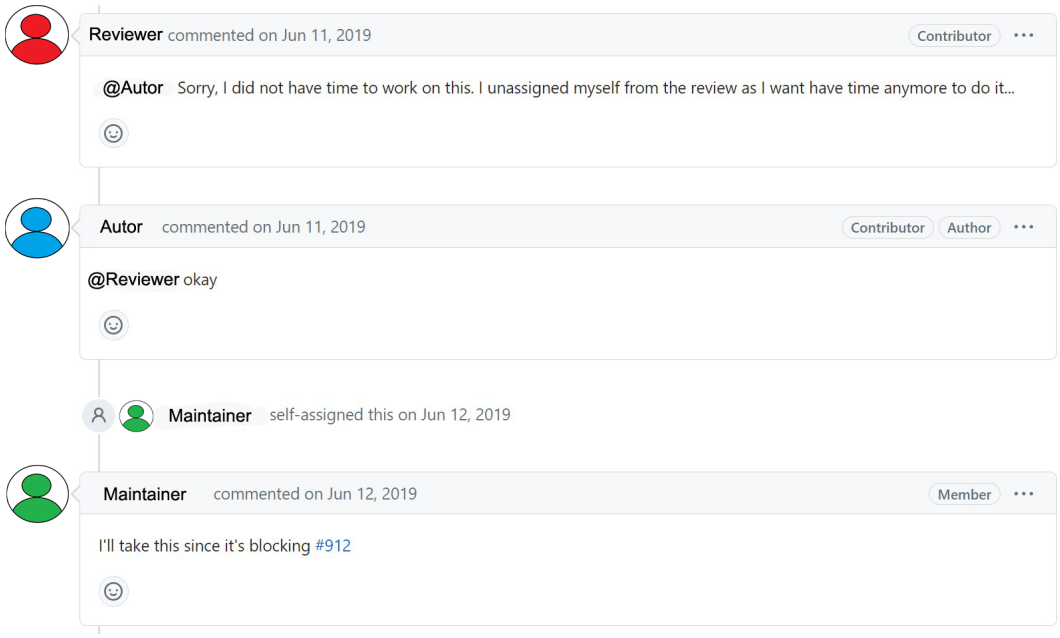


Abbildung 11: Auszug aus Pull Request Opencast #845 [26]

In zwei Sonderfällen des Zeitmangels ist es aufgrund unterschiedlicher Gründe nicht möglich, dass die Aufgaben von anderen Teilnehmenden übernommen werden. Einer dieser beiden Sonderfälle ist das Konzept des Maintainer Bottlenecks. Hierbei ist es aufgrund fehlender Befugnisse nicht möglich, dass die Beteiligten eines Pull Requests diesen abschließen.

Maintainer Bottleneck

Das Konzept Maintainer Bottleneck tritt auf, wenn ein Projekt über zu wenige Maintainer für die Bearbeitung der einkommenden Pull Requests verfügt. Dabei kann es sein, dass die Projekte zwar über genügend Reviewer verfügen, aber dennoch unter einem Rückstau bei den Pull Requests leiden, da die Reviewer nicht über die Befugnis verfügen, die PRs in letzter Instanz anzunehmen. Im Rahmen dieser Untersuchungen war dieses Konzept im OMZ-Projekt beobachtbar. Ein PR des Projekts hat beispielsweise nach der erfolgreichen Umsetzung von Korrekturen an dem PR und den positiven Bewertungen zwei verschiedener Reviewer seit über eineinhalb Jahren kein Urteil eines Maintainer erhalten (Abbildung 12). In dem Projekt ist es wiederkehrend zu beobachten, dass Reviewer die Autoren auf die Abhängigkeit von den Maintainern verweisen müssen.

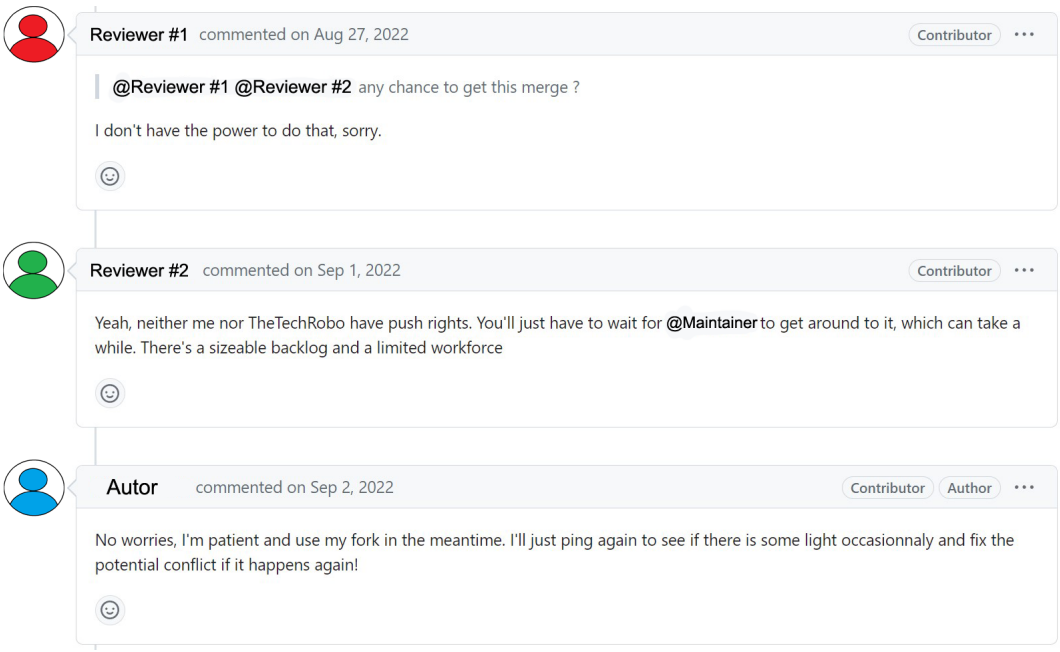


Abbildung 12: Auszug aus Pull Request OMZ #9676 [15]

Der zweite beobachtete Sonderfall des Zeitmangels betrifft PRs, welche unbeantwortet bleiben, weil ein bestimmter Reviewer nicht die notwendigen Kapazitäten aufbringen kann.

Unmaintainable Code

Das Konzept Unmaintainable Code betrifft PRs, bei denen Code-Teile des Projekts verändert werden, für die nur eine bestimmte Person die notwendigen Kenntnisse oder die alleinige Verantwortung hat. In allen vier untersuchten OSS-Projekten lassen sich implizite oder explizite Hinweise auf eine Unterteilung der Verantwortlichkeiten für verschiedene Teile des Quellcodes finden (Abbildung 13).

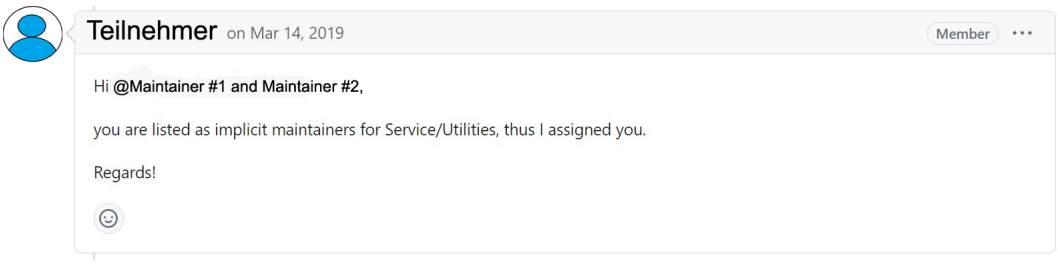


Abbildung 13: Auszug aus Pull Request ILIAS #1667 [28]

In einigen Fällen, kommt es dazu, dass Personen, welche die Verantwortlichkeit oder die notwendigen Kenntnisse für bestimmte Codestellen haben, nicht verfügbar sind, wodurch trotz Präsenz anderer Maintainer die PRs nicht fortgesetzt werden können. Das Technical Board des ILIAS Projekts hat in einem Strategiepapier aus dem Jahr 2017 selbst benannt, dass Teile ihres Codes keinen Maintainer haben. [4].

Das Problem lässt sich in vivo in einer PR des Opencast Projekts beobachten. Einer der beiden Hauptmaintainer des Projekts ist auf die Vertrautheit eines Reviewers mit einer Technologie angewiesen (Abbildung 14). In diesem Fall wird das Problem umgangen, indem auf die Einschätzung eines Reviewers verzichtet wird und auf die Selbsteinschätzung der Autoren vertraut wird.

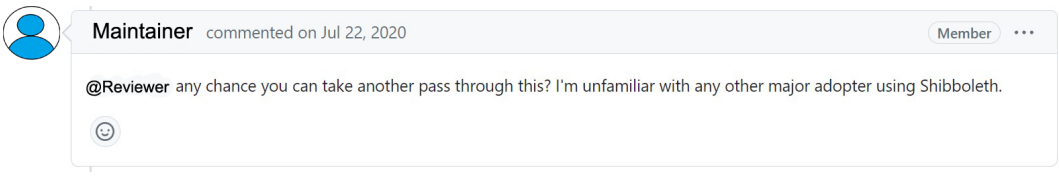


Abbildung 14: Auszug aus Pull Request Opencast #1607 [29]

In einem anderen Fall im ILIAS Projekt, ist eine PR seit über eineinhalb Jahren unbeantwortet trotz der wiederholten Adressierung des verantwortlichen Maintainers durch andere Maintainer (Abb. 15). Das Problem ist hier also kein Mangel an Maintainern. Die PR wird nicht weiter bearbeitet, da ein einzelner Verantwortlicher nicht verfügbar ist.

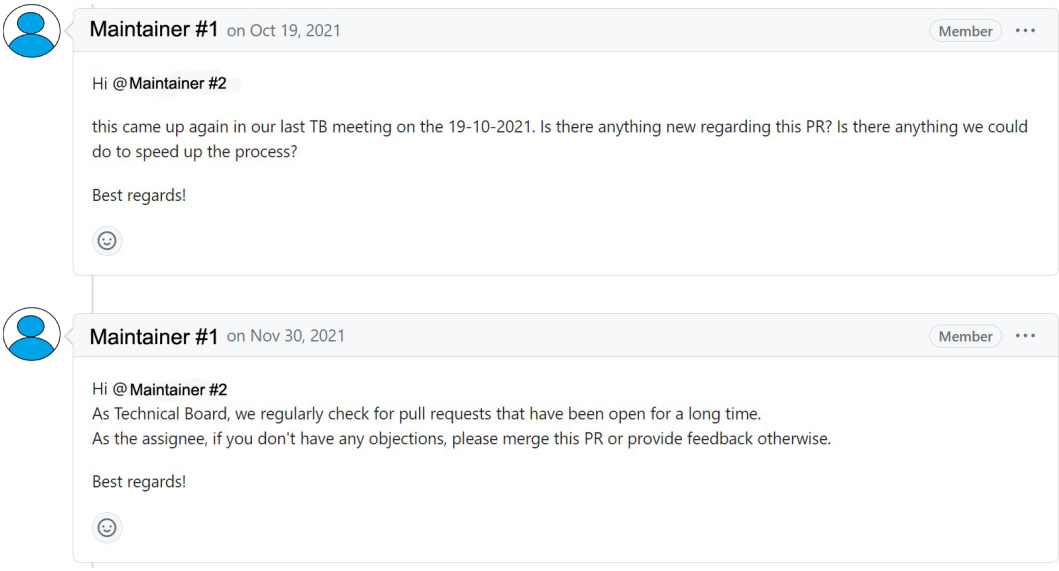


Abbildung 15: Auszug aus Pull Request ILIAS #3394 [23]

Vergessene Pull Requests

Ein weiteres Konzept, das zur verzögerten Fortsetzung von PRs führt, ist das simple Vergessen von PRs. Eine Ursache dafür ist die fehlende Information über den veränderten Status eines PRs. Ein Maintainer des Opencast Projekts verweist mehrmals explizit darauf, dass er einen Pull Request aufgrund fehlender Inkenntnissetzung über den veränderten Status nicht weitergehend bearbeitet hat (Abb. 16).

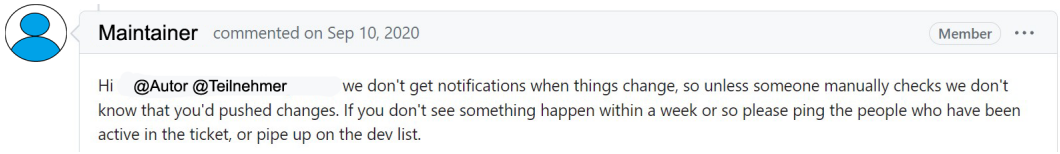


Abbildung 16: Auszug aus Pull Request Opencast #1607 [29]

Die Autoren von PRs gehen teilweise davon aus, dass das Hinzufügen neuer Commits zu einer bestehenden PR hinreichend signalisiert, dass die PR bereit für eine neue Begutachtung der Reviewer ist. Es kommt allerdings auch vor, dass einzelne Commits nur einen Teil der Änderungen umsetzen. So ist für die Reviewer ohne weitere Nachricht nicht erkennbar, dass sie einen PR erneut begutachten sollen (Abb. 17).

Abbildung 17: Auszug aus Pull Request Opencast #845 [26]

5.2 Veränderte Randbedingungen

Die zweite beobachtete Kategorie bündelt Ereignisse, die im Umfeld eines PRs auftreten und seine Fertigstellung stören. Infolge der Ereignisse muss es zu Korrekturen des PRs kommen, um diesen fertigstellen zu können. Diese Korrekturen bedeuten Mehrarbeit für die Beteiligten. Sie hätte verhindert werden können, wenn man die PRs vor Eintreten der Ereignisse fertiggestellt hätte.

Merge-Konflikte

Durch Veränderungen des Quellcodes eines Projekts können Pull Requests veralten, infolgedessen müssen die Pull Requests zuerst angepasst werden, bevor sie angenommen werden können. Für die Autor*innen der PR kann das teilweise mehrfache

Updaten der PRs deutliche Mehrarbeit bedeuten. In einzelnen Fällen ist beobachtbar, dass PRs aufgrund von Merge-Konflikten nicht fertiggestellt werden. Beispielhaft dafür ist ein PR des Sponge Projekts, der zwar bereits von einem Maintainer akzeptiert, aber noch nicht final gemerged wurde. Im Anschluss kam es zu Merge-Konflikten, die schlussendlich nicht mehr behoben wurden. Das Vorhaben, die PR neu zu erstellen, wurde ebenfalls nicht umgesetzt (Abb. 18).

Abbildung 18: Auszug aus Pull Request Sponge #1023 [24]

Maintainer der OSS-Projekte bemühen sich, aufkommende Merge-Konflikte zu erkennen und durch koordiniertes Handeln zu verhindern. Dabei werden ausgewählte PRs zurückgehalten, um in anderen PRs vorerst keine Konflikte hervorzurufen (Abb. 19).

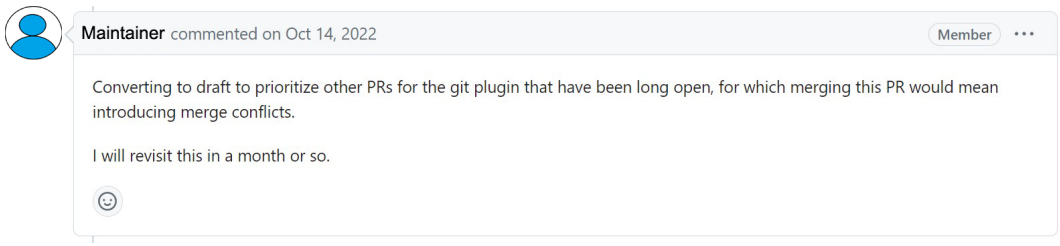


Abbildung 19: Auszug aus Pull Request OMZ #11246 [31]

Besonders relevant dabei sind PRs, die eine Vielzahl von Dateien mit subtilen Änderungen betreffen. Im Sponge Projekt beispielsweise werden solche PRs gesammelt und zu ausgewählten Zeitpunkten gemeinsam gemerged.

Versionssprünge

Drei der vier untersuchten Projekte betreiben für die unterschiedlichen Versionen der Software eigene Release-Branches. Vor der Veröffentlichung eines Major Updates erfolgt teilweise ein sogenannter Feature Freeze. Pull Requests, die neue Funktionalitäten einführen, sollen bis zum Zeitpunkt des Feature Freeze fertiggestellt werden.

Werden die PRs nicht rechtzeitig fertiggestellt, müssen sie an einen neuen Branch gerichtet werden (Abbildung 20).

Abbildung 20: Auszug aus Pull Request Sponge #1255 [22]

In diesen Fällen kann es notwendig sein, die PRs zu "rebasen", also den Ursprungs-Branch zu ändern. Häufig ist zu beobachten, dass neue Entwickelnde Probleme mit dem Rebasen ihrer PRs haben. In einigen Fällen bitten die Autor*innen um Hilfe für das Rebasen ihrer PRs oder lassen es von erfahrenen Entwickelnden umsetzen (Abb. 21). In anderen Fällen machen die Autor*innen beim Rebasen ihrer PRs Fehler und weichen im Anschluss auf neu eröffnete Kopien ihrer PRs aus.

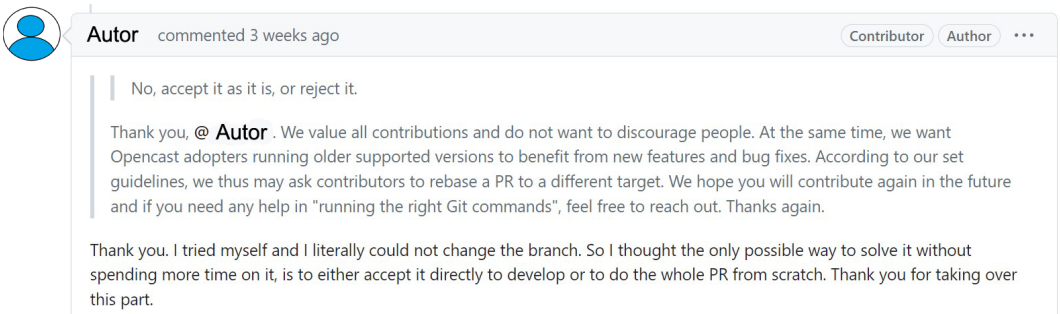


Abbildung 21: Auszug aus Pull Request Opencast #4716 [17]

Obsoleszenz

Obsoleszenz von Pull Requests kann eintreten, wenn Veränderungen im Umfeld der Pull Requests diesen ihre bisherige Bedeutung entziehen. Beobachtete Gründe für die Obsoleszenz waren ein verändertes Nutzungsverhalten der Autor*innen (Abb. 22) und die Umsetzung der Funktionalität durch neue Pull Requests. In den beobachteten Fällen folgte auf die Obsoleszenz eines Pull Requests meist die Schließung.

Abbildung 22: Auszug aus Pull Request OMZ #439 [16]

5.3 Projektspezifische Unwissenheit

Die letzte beobachtete Kategorie bündelt Konzepte von Unwissenheit, die im Verlauf von Pull Requests entweder zu Fragen oder zu Fehlern führen. Sowohl die Klärung der Fragen als auch das Beheben der Fehler sorgen im Verlauf der Pull Requests für Mehrarbeit für die Beteiligten. Alle folgenden Fälle betreffen dabei projektspezifische Inhalte, weshalb sie für in das Projekt einsteigende Personen besondere Relevanz haben.

Projektstruktur

Es ist beobachtbar, dass Einsteigende sich nicht über die verschiedenen, zur Versionierung der Software genutzten, Branches im Klaren sind. Ähnlich wie beim Konzept des Versionssprungs beschrieben, kann es auch beim Anlegen von PRs bereits dazu kommen, dass ein falscher Ziel-Branch ausgewählt wird. Infolgedessen kommt es zu vergleichbaren Problemen beim Rebasing wie im Unterkapitel Versionssprünge beschrieben.

Darüber hinaus nutzen einige Softwareprojekte mehrere Repositories zur Arbeit an verschiedenen Versionen oder Modulen einer Software. So ist es im Sponge-Projekt wiederkehrend beobachtbar, dass Einsteigende PRs in falschen Repositories eröffnen oder die PRs für fehlende Repositories erweitern müssen. Das Sponge-Projekt entwickelt mehrere Versionen der Software für unterschiedliche Umgebungen. Darüber hinaus wird die API-Komponente der Software in einem von der Implementierung getrennten Repository entwickelt.

Projektverlauf

In einigen PRs kommt es vor, dass Einsteigende aufgrund fehlender Übersicht über den Projektverlauf Pull Requests entwickeln, welche aufgrund vorhersehbarer Ereignisse überarbeitet werden müssen. Beispielsweise kommt es vor, dass Einsteigende Pull Requests erstellen, die bereits von anderen Entwickelnden umgesetzt werden oder Softwarekomponenten nutzen, die in naher Zukunft verändert werden. Infolgedessen werden die PRs entweder überflüssig oder müssen aufwändig umgearbeitet werden. Ein PR aus dem OMZ-Projekt beispielsweise setzt eine Verbesserung um, die bereits von einem anderen Autor erstellt wurde und muss daraufhin verworfen werden (Abb. 23).

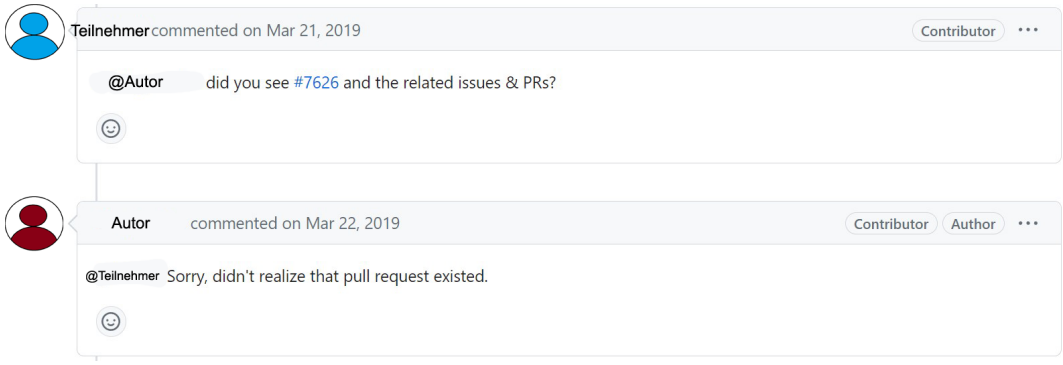


Abbildung 23: Auszug aus Pull Request OMZ #7671 [30]

Ein anderer PR aus dem Sponge-Projekt nutzt bestehende Teile der Software, die umstrukturiert werden und muss aufgrund dessen selbst aufwändig angepasst werden (Abb. 24).

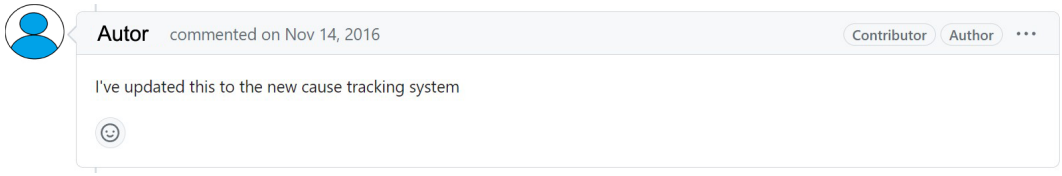


Abbildung 24: Auszug aus Pull Request Sponge #86 [25]

Nutzungsverhalten

Es ist wiederkehrend zu beobachten, dass Einsteigenden in Softwareprojekten das Nutzungsverhalten anderer Nutzenden unbewusst ist. Einige Autor*innen entwerfen PRs, die Konsequenzen für andere Nutzende übersehen. Erfahrene Teilnehmende haben teilweise ein Verständnis der verschiedenen Nutzungsverhalten, das es ihnen ermöglicht, derartige Pull Requests zu korrigieren. Eine solche Korrektur ist in einer PR des Opencast-Projekts zu finden. In dieser möchte die Autorin eine Funktionalität zugunsten einer anderen umbenennen. Einer der Maintainer erkennt, dass durch diese Umbenennung unverhältnismäßig viele Nutzende bestehende Routinen anpassen müssten (Abb. 25).

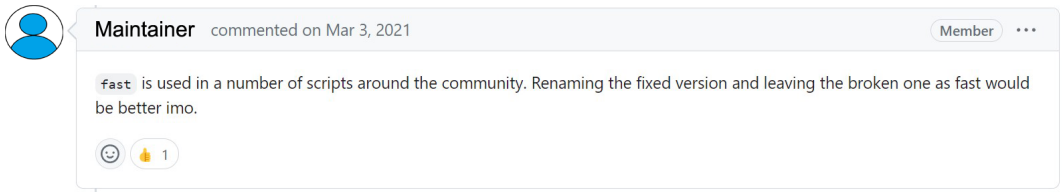


Abbildung 25: Auszug aus Pull Request Opencast #1861 [32]

Ein fast identischer Fall ist im OMZ-Projekt beobachtbar (Abb. 26).

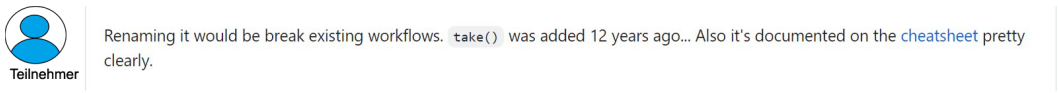


Abbildung 26: Auszug aus Pull Request OMZ #9749 [27]

Codestruktur

Neben dem Nutzungsverhalten ist es auch beobachtbar, dass Autor*innen unwissend über die Relevanz ihrer Änderungen für andere Codestellen sind. So kommt es dazu, dass Autor*innen Code entwickeln, der ihr spezifisch gewünschtes Verhalten erreicht, an anderer Stelle aber Konsequenzen hat, die ihnen nicht bewusst sind. In einem PR des ILIAS-Projektes beispielsweise überschreibt ein Autor den Rückgabewert einer Funktion, die an mehreren anderen Stellen benutzt wird. Laut der Einschätzung eines Maintainer des Projektes würde diese Änderung fundamentale Funktionen der Software behindern (Abb. 27).

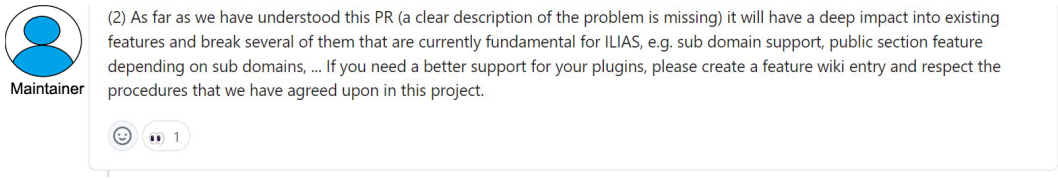


Abbildung 27: Auszug aus Pull Request ILIAS #1667 [28]

Alle diese Konzepte erfordern die Korrektur der gestellten PRs die sowohl Autor*innen als auch Reviewer mit Mehrarbeit belasten. In einigen wenigen Fällen geben die Autor*innen von PRs explizit an, aufgrund der Korrekturen einen PR nicht fertigzustellen. In anderen Fällen ist beobachtbar, dass PRs im Anschluss an notwendig gewordene Korrekturen verlassen werden. In diesen Fällen sind die Gründe aber nicht final feststellbar, da die Beteiligten nicht explizit benannt haben, warum sie die PRs nicht fortgesetzt haben. Ein Verlassen von Pull Requests aufgrund notwendig gewordener Korrekturen wird auch in der Literatur erwähnt. Li et al. haben als Gründe für das vorzeitige Verlassen von PRs u. a. die Überforderungen mit den Korrekturen und die unerwartete Mehrarbeit ermittelt [43].

5.4 Theoretisches Modell

Zwischen den vorgestellten Kategorien bestehen teilweise Zusammenhänge. Drei abstrakte Kernkonzepte stellen dabei die Zusammenhänge dar. In der Abbildung 30 sind diese Kernkonzepte orangefarben mit runden Ecken dargestellt. Kategorien aus der Grounded Theory Methodology sind in der Abbildung in beigefarbenen Rechtecken dargestellt, während einige der zu den Kategorien gehörenden Konzepte als blaue Rechtecke innerhalb der Kategorien abgebildet sind. Rote Parallelogramme stellen während der Anwendung der GTM beobachtete Konsequenzen dar.

Wie in 5.3 gezeigt, kann in verschiedenen Fällen projektspezifische Unwissenheit zu Fragen oder Fehlern in den PRs führen. Die notwendigen Korrekturen, um Fehler in den PRs zu korrigieren, erzeugen Mehrarbeit, die ohne diese Korrekturen nicht von

Nöten wäre. Dabei kommt es vor, dass Einsteigende von den notwendigen Anforderungen überfordert sind. Neben den Korrekturen führt auch die Beantwortung der aufkommenden Fragen zu Mehrarbeit für die Reviewer. Die entstandene Mehrarbeit erfordert zur Fertigstellung der PRs einen zeitlichen Aufwand, der zu Verzögerungen der PRs führt.

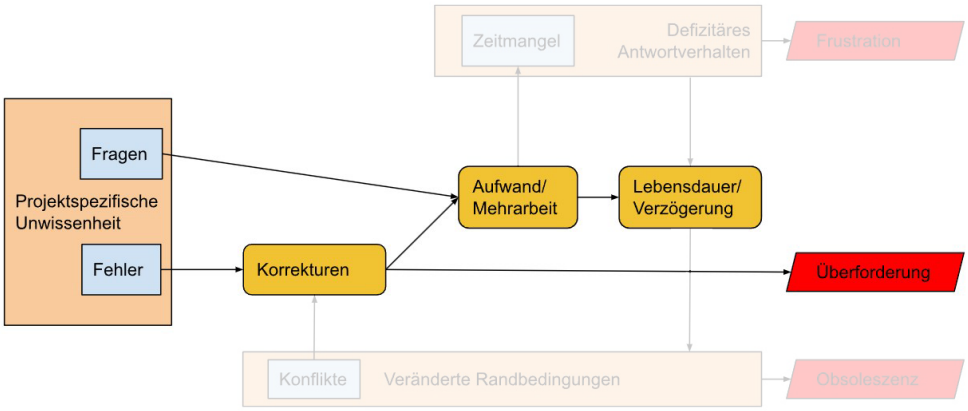


Abbildung 28: Theoretisches Modell „Herausforderungen beim Onboarding und Pull Request“ - Teil 1

Die entstehende Mehrarbeit in den PRs erhöht den Zeitaufwand, den die Reviewer aufbringen müssen, da sie beispielsweise Korrekturvorschläge entwerfen oder Fragen beantworten müssen. Außerdem müssen verbesserte PRs ein erneutes Mal begutachtet werden. In den Untersuchungen dieser Arbeit wurde beobachtet, dass die Reviewer in Open-Source-Projekten teilweise unter Zeitmangel leiden. Dieser erhöhte Zeitaufwand wirkt sich negativ auf den Gesamtarbeitsaufwand der OSS-Projekte aus und erhöht den Zeitmangel weiter. Eine Folge davon ist ein defizitäres Antwortverhalten (vgl. 5.1), das zur weiteren Verzögerung der PRs führt. Darüber hinaus war zu beobachten, dass einige Autor*innen frustriert auf das langsame Antwortverhalten der Reviewer reagierten und teilweise ihre PRs abbrachen.

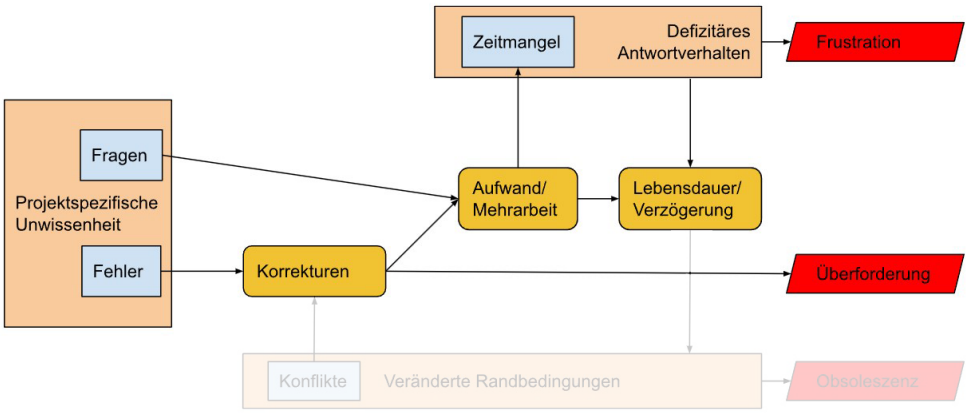


Abbildung 29: Theoretisches Modell „Herausforderungen beim Onboarding und Pull Request“ - Teil 2

Eine Folge der verzögerten PRs können die Konflikte sein, die in Kapitel 5.2 beschrieben wurden. Wären die PRs vor Auftreten der Veränderungen im Umfeld (Kategorie) der PRs fertiggestellt worden, hätten diese Konflikte vermieden werden können. Durch das Auftreten von Konflikten müssen an den PRs Korrekturen vorgenommen werden, bevor diese angenommen werden können. Diese Korrekturen bedeuten ein Auftreten von Mehrarbeit, die in vielen der beobachteten Fälle von den Autor*innen selbst übernommen wurden. Allerdings ist es auch beobachtbar, dass die Konflikte von Reviewern behoben wurden. Durch Veränderungen im Umfeld kann es außerdem zur Obsoleszenz und einer darauf folgende Schließung der Pull Requests kommen.

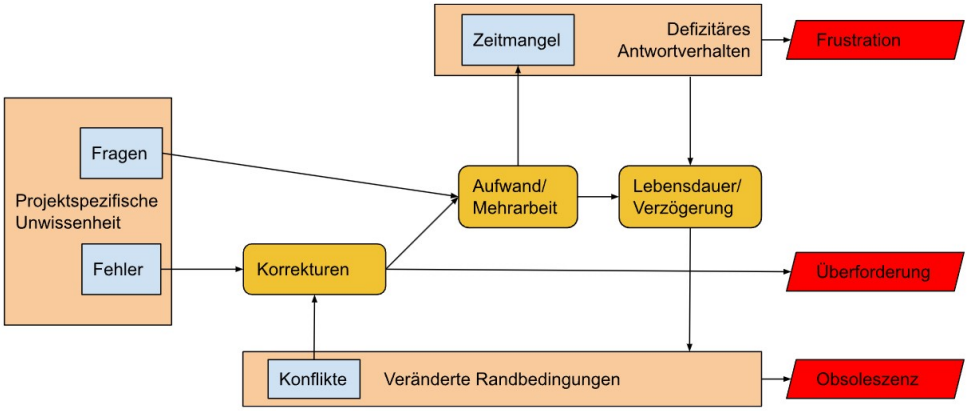


Abbildung 30: Theoretisches Modell „Herausforderungen beim Onboarding und Pull Request“ - Teil 3

Die Gültigkeit des entwickelten Modells wird sicher dadurch eingeschränkt, dass die aufgezeigten Beziehungen keine zwingenden Kausalitäten darstellen. Das Auftreten der Zusammenhänge ist von Bedingungen abhängig. So führt beispielsweise nicht jede Verzögerung einer PR zwangsweise zu Konflikten. Für das Eintreten des Phänomens ist es notwendig, dass Veränderungen im Umfeld der PR eintreten. Je länger ein PR jedoch offen ist, desto mehr Zeit ergibt sich für das potentielle Auftreten der Veränderungen im Umfeld. Außerdem ist bedeutend, dass sich der Großteil der dargestellten Zusammenhänge auf der Ebene eines einzelnen Pull Request abspielen kann. Der verstärkende Einfluss der Mehrarbeit auf den Zeitmangel spielt sich allerdings auf einer Projektebene ab. Die Arbeit an einem einzelnen PR wird selten auslastend für Teilnehmende sein, hinzukommende PRs können aber bereits zeitlich überlastete Teilnehmende noch weiter überlasten. Trotz der nur bedingten Gültigkeit des Modells, wird mit der grafischen Darstellung doch eine gewisse Abstraktion der Ergebnisse der vorliegenden Arbeit erreicht.

6 Fazit und Ausblick

Diese Arbeit untersucht die Pull Requests von Neueinsteigenden unter der Verwendung der Grounded Theory Methodology. Das Ergebnis dieser Untersuchungen ist eine Reihe von Konzepten, die Herausforderungen in den Pull Requests von Neuein-

steigenden darstellen. Die Konzepte sind Teil eines theoretischen Modells, das ihre Zusammenhänge anhand von drei abstrakten Kernkonzepten repräsentiert. Abschließend kann man feststellen, dass der Vorgang des Onboardings durch Phänomene beeinträchtigt wird, die von beiden beteiligten Seiten – den Neueinsteigenden und den bereits Teilnehmenden - beeinflusst wird. Auf der Seite der Neueinsteigenden sind es oftmals unzureichende projektspezifische Kenntnisse, die eine Hürde beim Onboarding darstellen. Auf Seiten der bereits integrierten Projekteilnehmenden wird das projektspezifische Zeitmanagement immer wieder zu einer Herausforderung für das Onboarding der Neueinsteigenden.

Die Validierung des theoretischen Modells dieser Arbeit könnte ein Untersuchungsgegenstand zukünftiger wissenschaftlicher Arbeiten sein. Ein Ziel könnte es sein, die statistische Signifikanz der gezeigten theoretischen Zusammenhänge quantitativ zu untersuchen. Darüber hinaus könnten zukünftige Arbeiten die Beziehung zwischen der Lebensdauer von Pull Requests und dem Erfahrungsstand der Autor*innen weitergehend untersuchen.

Literatur

[1] Alireza Amrollahi, Mohammad Khansari und Amir Manian. „How Open Source Software Succeeds? A Review of Research on Success of Open Source Software“. In: *International Journal of Information and Communication Technology Research* 6 (März 2014), S. 67–77. DOI: 10.13140/2.1.4835.1367.

[2] *Become a sponsor to Oh My Zsh*. https://github.com/sponsors/ohmyzsh. Dokumentation. [Online; Zugriffsdatum: 29.03.2023].

[3] Christian Bird u. a. „Quantitative Study of Open Source Immigration“. In: (2014).

[4] ILIAS Technical Board. *Strategy 2017 - Review*. https://docu.ilias.de/goto_docu_file_7342_download.html. Dokumentation. [Online; Zugriffsdatum: 29.03.2023]. 2017.

[5] Alan Brown und Grady Booch. „Reusing Open-Source Software and Practices: The Impact of Open-Source on Commercial Vendors“. In: *Software Reuse: Methods, Techniques, and Tools, 7th International Conference, ICSR-7, Lecture Notes in Computer Science*. Bd. 2319. Apr. 2002, S. 123–136. ISBN: 978-3-540-43483-2. DOI: 10.1007/3-540-46020-9_9.

[6] Berthold Bußkamp. *ADD specialist worker nodes #3741*. https://github.com/opencast/opencast/pull/3741. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[7] Berthold Bußkamp. *ADD support for f4v file type #4280*. https://github.com/opencast/opencast/pull/4280. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[8] Berthold Bußkamp. *Prioritization of jobs from older workflows #4393*. https://github.com/opencast/opencast/pull/4393. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[9] Berthold Bußkamp. *Remove unused class DispatchableComparator #4319*. https://github.com/opencast/opencast/pull/4319. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[10] Berthold Bußkamp. *Usage of CPU load for job dispatching #4212*. https://github.com/opencast/opencast/pull/4212. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[11] Eugenio Capra und Anthony Wasserman. „A Framework for Evaluating Managerial Styles in Open Source Projects“. In: Bd. 275. Juli 2008. ISBN: 978-0-387-09683-4. DOI: 10.1007/978-0-387-09684-1_1.

[12] Schweik Charles M. und English Robert C. *Internet Success : A Study of Open-Source Software Commons*. The MIT Press, 2012. ISBN: 9780262017251. URL: https://search.ebscohost.com/login.aspx?direct=true&db=e000xww&AN=462604&site=ehost-live.

[13] Marco Gerosa u. a. *The Shifting Sands of Motivation: Revisiting What Drives Contributors in Open Source*. 2021. arXiv: 2101.10291 [cs.SE].

[14] R Ghosh u. a. *The free/libre and open source software survey and study—FLOSS final report*. Jan. 2002.

[15] *GitHub.com - Add git aliases #9676*. https://github.com/ohmyzsh/ohmyzsh/pull/9676. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2021.

[16] *GitHub.com - Add git indication to show if a modified file is partially staged #439*. https://github.com/ohmyzsh/ohmyzsh/pull/439. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2011.

[17] *GitHub.com - added missing publish-uploaded-assets.xml workflow #4716*. https://github.com/opencast/opencast/pull/4716. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2023.

[18] *GitHub.com - Adding fly plugin (Concourse CI) #9711*. https://github.com/ohmyzsh/ohmyzsh/pull/9711. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2021.

[19] *GitHub.com - feat(aws): support sso profiles #9795*. https://github.com/ohmyzsh/ohmyzsh/pull/9795. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2021.

[20] *GitHub.com - feat(git): Add git branch rename alias #10748*. https://github.com/ohmyzsh/ohmyzsh/pull/10748. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[21] *GitHub.com - feat(git): add gupom and gupomi aliases #10261*. https://github.com/ohmyzsh/ohmyzsh/pull/10261. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[22] *GitHub.com - Feed animal event #1255*. https://github.com/SpongePowered/Sponge/pull/1255. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2017.

[23] *GitHub.com - Fix and Improve OpenID Connect integration #3394*. https://github.com/ILIAS-eLearning/ILIAS/pull/3394. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2021.

[24] *GitHub.com - Fix various (extremely) minor issues with Shadows #1023*. https://github.com/SpongePowered/Sponge/pull/1023. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2016.

[25] *GitHub.com - Implement simulated methods using a fake player, controlled by a PlayerSimulator #86*. https://github.com/SpongePowered/Sponge/pull/86. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2015.

[26] *GitHub.com - MH-13447, Admin Interface Feeds Endpoint #845*. https://github.com/opencast/opencast/pull/845. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2019.

[27] *GitHub.com - mkcd plugin - create directory and change into it #9749*. https://github.com/ohmyzsh/ohmyzsh/pull/9749. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2021.

[28] *GitHub.com - Return configured* $http_pathifitissetinilUtil ::_g etHttpPath$#1667. https://github.com/ILIAS-eLearning/ILIAS/pull/1667. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2019.

[29] *GitHub.com - Shibboleth dynamic login handler #1607*. https://github.com/opencast/opencast/pull/1607. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2020.

[30] *GitHub.com - Speed up rbenv plugin #7671*. https://github.com/ohmyzsh/ohmyzsh/pull/7671. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2019.

[31] *GitHub.com - style(git): use full-style arguments in aliases #11246*. https://github.com/ohmyzsh/ohmyzsh/pull/11246. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[32] *GitHub.com - Update fast.xml to work with republish workflows (update for MH-13617) #1861*. https://github.com/opencast/opencast/pull/1861. Pull Request. [Online; Zugriffsdatum: 29.03.2023]. 2020.

[33] Georgios Gousios, Margaret-Anne Storey und Alberto Bacchelli. „Work Practices and Challenges in Pull-Based Development: The Contributor's Perspective". In: *Proceedings of the 38th International Conference on Software Engineering*. ICSE '16. Austin, Texas: Association for Computing Machinery, 2016, S. 285–296. ISBN: 9781450339001. DOI: 10.1145/2884781.2884826. URL: https://doi.org/10.1145/2884781.2884826.

[34] Georgios Gousios u. a. „Work Practices and Challenges in Pull-Based Development: The Integrator's Perspective". In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Bd. 1. 2015, S. 358–368. DOI: 10.1109/ICSE.2015.55.

[35] *Hauptbegriffe qualitativer Sozialforschung / Ralf Bohnsack, Alexander Geimer, Michael Meuser (Hrsg.)*. ger. Opladen ; Toronto, 2018.

[36] Israel Herraiz u. a. „The Processes of Joining in Global Distributed Software Projects". In: *Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner*. GSD '06. Shanghai, China: Association for Computing Machinery, 2006, S. 27–33. ISBN: 1595934049. DOI: 10.1145/1138506.1138513. URL: https://doi.org/10.1145/1138506.1138513.

[37] *ILIAS-eLearning/ILIAS*. https://github.com/ILIAS-eLearning/ILIAS. Projektseite. [Online; Zugriffsdatum: 29.03.2023].

[38] OpenSource Initiative. *The Open Source Definition*. https://opensource.org/osd/. Website. [Online; Zugriffsdatum: 29.03.2023].

[39] Sayedhassan Khatoonabadi u. a. „On Wasted Contributions: Understanding the Dynamics of Contributor-Abandoned Pull Requests". In: *ACM Transactions on Software Engineering and Methodology* (Mai 2022). DOI: 10.1145/3530785.

[40] Lars Kiesow. *Thesis related to Opencast or something else*. https://hackmd.io/@lkiesow/opencast-thesis. Website. [Online; Zugriffsdatum: 29.03.2023]. 2023.

[41] Karim Lakhani und Robert Wolf. „Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects". In: *Perspectives on Free and Open Source Software* (Sep. 2003). DOI: 10.2139/ssrn.443040.

[42] Damien Legay, Alexandre Decan und Tom Mens. *On the impact of pull request decisions on future contributions*. 2018. arXiv: 1812.06269 [cs.SE].

[43] Zhixing Li u. a. „Are You Still Working on This An Empirical Study on Pull Request Abandonment". In: *IEEE Transactions on Software Engineering* PP (Jan. 2021), S. 1–1. DOI: 10.1109/TSE.2021.3053403.

[44] mamercad. *GitHub.com - Adding fly plugin (Concourse CI) #9711*. https://github.com/ohmyzsh/ohmyzsh/pull/9711. Kommentar. [Online; Zugriffsdatum: 29.03.2023]. 2022.

[45] *Mitgliederliste - Übersicht über institutionelle Mitglieder und Service-Provider*. https://docu.ilias.de/ilias.php?ref_id=3650&cmdClass=ilrepositorygui&cmdNode=8m&baseClass=ilRepositoryGUI. Dokumentation. [Online; Zugriffsdatum: 29.03.2023].

[46] Daricélio Moreira Soares u. a. „What factors influence the lifetime of pull requests?“ In: *Software: Practice and Experience* 51.6 (2021), S. 1173–1193. DOI: https://doi.org/10.1002/spe.2946. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2946. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2946.

[47] Kumiyo Nakakoji u. a. „Evolution Patterns of Open-Source Software Systems and Communities“. In: *International Workshop on Principles of Software Evolution (IWPSE)* (Jan. 2003). DOI: 10.1145/512035.512055.

[48] Siobhán O'Mahony. „The governance of open source initiatives: what does it mean to be community managed?“ eng. In: *Journal of management and governance* 11.2 (2007), S. 139–150. ISSN: 1385-3457.

[49] Christopher Oezbek, Lutz Prechelt und Florian Thiel. „The onion has cancer: Some social network analysis visualizations of Open Source project communication“. In: *Proceedings - International Conference on Software Engineering* (Apr. 2010). DOI: 10.1145/1833272.1833274.

[50] *ohmyzsh/ohmyzsh*. https://github.com/ohmyzsh/ohmyzsh. Projektseite. [Online; Zugriffsdatum: 29.03.2023].

[51] *Opencast - The free and open source solution for automated video capture and distribution at scale*. https://opencast.org/. Website. [Online; Zugriffsdatum: 29.03.2023].

[52] *Opencast Governance*. https://docs.opencast.org/develop/developer/#governance/. Dokumentation. [Online; Zugriffsdatum: 29.03.2023].

[53] *opencast/opencast*. https://github.com/opencast/opencast. Projektseite. [Online; Zugriffsdatum: 29.03.2023].

[54] J.W. Paulson, G. Succi und A. Eberlein. „An empirical study of open-source and closed-source software products“. In: *IEEE Transactions on Software Engineering* 30.4 (2004), S. 246–256. DOI: 10.1109/TSE.2004.1274044.

[55] Eric S. Raymond. *The cathedral and the bazaar : musings on Linux and open source by an accidental revolutionary / Eric S. Raymond*. eng. Rev. ed. [der Ausg.] 1999. Beijing ; Cambridge ; Farnham ; Köln ; Paris ; Sebastopol ; Taipei ; Tokyo: O'Reilly, 2001. ISBN: 0-596-00108-8.

[56] Dirk Riehle u. a. „Paid vs. Volunteer Work in Open Source“. In: *2014 47th Hawaii International Conference on System Sciences*. 2014, S. 3286–3295. DOI: 10.1109/HICSS.2014.407.

[57] Jeffrey Roberts, Il-Horn Hann und Sandra Slaughter. „Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects“. In: *Management Science* 52 (Juli 2006), S. 984–999. DOI: 10.1287/mnsc.1060.0554.

[58] *SpongePowered/Sponge*. https://github.com/SpongePowered/Sponge. Projektseite. [Online; Zugriffsdatum: 29.03.2023].

[59] *Staff*. https://docs.spongepowered.org/stable/en/about/staff.html. Dokumentation. [Online; Zugriffsdatum: 29.03.2023].

[60] Igor Steinmacher, Marco Aurelio Gerosa und David Redmiles. „Attracting, Onboarding, and Retaining Newcomer Developers in Open Source Software Projects“. In: *Workshop on Global Software Development in a CSCW Perspective held in conjunction with the17th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW'14)*. Feb. 2014.

[61] Igor Steinmacher u. a. „A systematic literature review on the barriers faced by newcomers to open source software projects“. In: *Information and Software Technology* 59 (Nov. 2014). DOI: 10.1016/j.infsof.2014.11.001.

[62] Igor Steinmacher u. a. „Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects“. In: *CSCW '15: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. März 2015. DOI: 10.1145/2675133.2675215.

[63] Joseph Taylor und Ramakrishna Dantu. „For Love or Money? Examining Reasons behind OSS Developers' Contributions“. In: *Information Systems Management* 39.2 (2022), S. 122–137. DOI: 10.1080/10580530.2021.1879323. eprint: https://doi.org/10.1080/10580530.2021.1879323. URL: https://doi.org/10.1080/10580530.2021.1879323.

[64] Yunwen Ye und Kouichi Kishida. „Toward an understanding of the motivation of open source software developers“. In: *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*. Juni 2003, S. 419–429. ISBN: 0-7695-1877-X. DOI: 10.1109/ICSE.2003.1201220.

[65] Xunhui Zhang u. a. „Pull Request Latency Explained: An Empirical Overview“. In: *Empirical Softw. Engg.* 27.6 (Nov. 2022). ISSN: 1382-3256. DOI: 10.1007/s10664-022-10143-4. URL: https://doi.org/10.1007/s10664-022-10143-4.

[66] Xunhui Zhang u. a. „Pull Request Decisions Explained: An Empirical Overview“. In: *IEEE Transactions on Software Engineering* 49.2 (2023), S. 849–871. DOI: 10.1109/TSE.2022.3165056.

[67] Yuxia Zhang u. a. „Companies' Participation in OSS Development–An Empirical Study of OpenStack“. In: *IEEE Transactions on Software Engineering* 47.10 (2021), S. 2242–2259. DOI: 10.1109/TSE.2019.2946156.

Abbildungsverzeichnis

Tabellenverzeichnis