Technische Hochschule **Rosenheim**

Fakultät für Informatik

Studiengang Studiengang-einsetzen

Was macht Open Source Projekte erfolgreich?

Bachelor Thesis

von

Paul-Gerhard Barbu

Datum der Abgabe: 07.06.2022
Erstprüfer: Prof. Dr. Gerd Beneken
Zweitprüfer: Prof. Dr. Wolfgang Mühlbauer

Eigenständigkeitserklärung / Declaration of Originality

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

München, den 07.06.2022

Paul-Gerhard Barbu

Abstract

Ziel dieser Arbeit war es mittels einer Datenerhebung von ausgewählten Open Source Projekten und einer Umfrage herauszufinden, welche Faktoren zum Erfolg der Projekte beitragen. Untersucht wurden der Einfluss von Lizenzen, Dokumentation, Beliebtheit, Vorhandensein vom Code of Conduct und Contributing Guide sowie von Sponsoren. Die Umfrage hatte 308 Teilnehmer und es wurden 108 Projekte analysiert.

Die Umfrage hat gezeigt, dass Dokumentation das wichtigste Element eines erfolgreichen Projektes darstellt 81% der Befragten gaben an Aufgrund schlechter Dokumentation ein Projekt nicht genutzt zu haben. Beliebtheit gilt hierbei als zweit wichtigstes Kriterium, 46% der Befragten gaben an Aufgrund geringer Beliebtheit eines Projektes es nicht genutzt zu haben.

Die Datenerfassung hat gezeigt, dass permissive Lizenzen die am häufigsten verwendeten Lizenzen sind. Außerdem sind permissiv lizenzierte Projekte erfolgreicher als restriktiv lizenzierte.

Schlagworte: Open Source, GitHub, Lizenzen, Dokumentation

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

Merge	Ein Merge beschreibt den Vorgang, bei dem zwei Dateien miteinander verglichen und zusammengefügt werden.
Branch	Ein Branch ist eine Abzweigung des original Verzeichnisses
Pull Request	Ein Pull Request auf GitHub ist eine Methode, um Team Mitglieder über ein Update auf einem Branch zu informieren. Akzeptierte Pull Requests führen zu einem Merge in ein anderen Branch.
(GitHub) Issues	GitHub Issues sind eine Methode, um Feedback, Aufgaben und Bugs zu verwalten.
Repository	Das GitHub Repository enthält alle Dateien des Projektes.
Commit	Vereinfacht dargestellt sind Commits Speicherpunkte in einem Repository

1 Einleitung

Open Source ist heutzutage ein fester Bestandteil der Softwareindustrie. Von Frontend-Entwicklung über Datenbanken bis hin zu Machine Learning, überall kommen Open Source Bibliotheken, Frameworks und Programme zum Einsatz.

Im Frontend werden verschiedene Frameworks bzw. Bibliotheken wie Angular oder React verwendet, im Fall von Datenbankmanagementsysteme gibt es ebenfalls eine Vielzahl an Optionen wie PostgreSQL, MySQL oder MongoDB. Im Bereich Machine Learning werde Frameworks wie TensorFlow, Keras oder SciKit-Learn genutzt.

Doch was macht diese Projekte erfolgreich? Mit dieser Bachelorarbeit soll die Frage beantwortet werden, *welche Faktoren haben Einfluss auf den Erfolg von Open Source Projekten*, insbesondere in der JavaScript / TypeScript Umgebung haben. In dieser Arbeit soll der Einfluss von **Lizenzen, Dokumentation, Code of Conduct, Contributing Guide, Beliebtheit und Vorhandensein von Sponsoren**, auf den Erfolg von Open Source Projekten erforscht werden.

Mittels einer Datenerhebung von ausgewählten Open Source Projekten sowie einer Umfrage soll dies herausgefunden werde. Ein zentraler Punkt dieser Ausarbeitung sind die extrinsischen sowie intrinsischen Anreize, die Nutzer zur Auswahl eines Produktes motivieren [Mid12]. Warum React und nicht Angular? Warum Debian und nicht Ubuntu? Aspekte wie die interne Führung und Organisation der Projekte wird hierbei nicht thematisiert.

1.1 Definition von Open Source

Diese Arbeit folgt der Definition für Open Source wie sie von der *Open Source Initiative* (OSI) vorgegeben wird [Ope07]. Entsprechend werden nur Projekte betrachtet die von der OSI genehmigten sind. Des Weiteren wird im Laufe dieser Arbeit zwischen restriktiven und permissiven Lizenzen unterschieden.

Eine permissive Lizenz ist eine sehr freizügige Lizenz, diese erlaubt die Nutzung, Modifikation und Weiterverbreitung ohne weitere Einschränkungen. Permissive Lizenzen können in proprietärer Software verwendet werden. Beispiele für permissive Lizenzen sind MIT, BSD und Apache [Ope07].

Restriktive Lizenzen erlauben ebenfalls die Nutzung, Modifikation und Weiterverbreitung der Software. Allerdings mit der Einschränkungen, dass Modifikationen und Weiterverbreitungen ebenfalls restriktiv lizenziert werden müssen. Wird beispielsweise eine GPL lizenzierte Bibliothek in einem Software Projekt verwendet, muss die gesamte Software ebenfalls GPL lizenziert werden. Dies gilt allerdings nur dann wenn die Software auch verbreitet wird, die Regel greift also nicht für private und interne Zwecke. Restriktive Bibliotheken und Frameworks können somit nicht in proprietärer Software eingesetzt werden. Beispiele für restriktive Lizenzen sind GPL, AGPL und LGPL [Ope07]

1.2 Erfolg definieren

Der Erfolg wird in der Literatur häufig in verschiedene Bereiche unterteilt. Im Artikel von Midha und Palvia wird zwischen *Markterfolg* und *technischem Erfolg* unterschieden. Midha et al. definieren Markterfolg als Grad des Nutzerinteresses an einem Projekt, welches sich in der Beliebtheit des Projektes widerspiegelt. Den technischen Erfolg definieren Midha et al. durch die Entwickleraktivität, d.h. durch den Aufwand, den die Entwickler für das Projekt betreiben beispielsweise die Häufigkeit und Frequenz von Updates und neuen Versionen [Mid12].

Im Artikel von Steward et al. wird zwischen *Nutzerinteresse* und *Entwickleraktivität* unterschieden [Ste06]. Subramaniam et al. gehen sogar weiter und unterteilen den Erfolg in *Nutzerinteresse, Entwicklerinteresse und Projektaktivität* [Sub09].

Diese Bereiche werden getrennt betrachtet, da verschiedene Faktoren unterschiedlichen Einfluss auf den Erfolg eines Projektes haben. Laut Midha et al. und Steward et al. wirkt sich wachsendes Interesse der Nutzern positiv auf den Markterfolg aus, während sich die Entwickleraktivität positiv den technischen Erfolg auswirken [Mid12, Ste06].

In dieser Arbeit werden die Erfolgsfaktoren betrachtet, die zum Markt- bzw. technischen Erfolg eines Projektes beitragen. In Midha et al. wurde der Markterfolg mittels der Downloadzahlen gemessen [Mid12]. Diese Arbeit erweitert die Erfolgsmetrik zusätzlich, um GitHub Sterne, da sich diese Metrik ebenfalls gut eignet, um die Beliebtheit eines Projektes zu messen. Die Downloadzahlen werden hierbei von NPM bezogen. Für den technischen Erfolg verwenden Midha et al. die Anzahl der Commits [Mid12]. Zusätzlich werden in dieser Arbeit noch die Anzahl der Mitwirkenden als technischen Erfolg gewertet.

2 Erfolgsfaktoren

Im folgenden Kapitel werden verschiedene mögliche Erfolgsfaktoren mittels Literatur analysiert und basierend darauf Hypothesen abgeleitet. In späteren Kapiteln werden diese Hypothesen anschließend mittels einer Datenerhebung und Umfrage geprüft.

2.1 Lizenzen

Laut Subramaniam et al. spielen Lizenzen eine signifikante Rolle für den Erfolg von Open Source Software. Freizügige Lizenzen wie MIT oder BSD haben einen positiven Einfluss vor allem auf Softwareentwickler. Denn Entwickler nutzen OSS, um es in eigene Projekte einzubauen, gegebenenfalls zu modifizieren und ein Endprodukt mit der OSS Komponente weiterzuverbreiten. Das ist mit restriktiven Lizenzen wie GPL meist nicht bedingungslos umsetzbar. Restriktive Lizenzen wie GPL wirken sich daher negativ bis neutral auf den Erfolg von OSS aus. Wenn die Software allerdings an Endnutzer gerichtet ist, wie zum Beispiel der Instant-Messaging-Dienst Telegram[1], spielt die Lizenz eine weniger wichtige Rolle, da Weiterverbreitung und Modifizierung für diese Nutzergruppe keine Rolle spielen [Sub09].

Stewart et al. widerspricht der zweiten Aussage von Subramaniam et al. laut ihnen haben permissive Lizenzen nicht nur auf das Entwicklerinteresse, sondern auch auf das Nutzerinteresse einen positiven Einfluss. Während restriktive Lizenzen einen nicht signifikanten Einfluss auf Entwickleraktivität hätten. [Ste06]

Laut Midha und Palvia wirken sich permissive Lizenzen positiv auf den Markterfolg aus, allerdings nur zu Beginn eines Projektes. Restriktive Lizenzen wiederum wirken sich negativ auf den technischen Erfolg aus [Mid12].

Meine Hypothese ist, dass sich offene Lizenzen durchgängig positiv auf ein Projekt auswirken. Offene Lizenzen werden tendenziell eher von Unternehmen verwendet als Projekte mit restriktiven Lizenzen, das führt zum einen dazu, dass die Beliebtheit und Bekanntheit des Projektes steigt, als auch die Wahrscheinlichkeit das die Unternehmen zum Open Source Projekt etwas beitragen oder Sponsoren werden.

H 1. *Offene Lizenzen haben positiven Einfluss auf den Markterfolg.*

Steigt die Beliebtheit eines Projekts, so steigt auch das Interesse von Open Source Entwickler an einem renommierten Projekt mitzuwirken.

H 2. *Offene Lizenzen haben positiven Einfluss auf den technischen Erfolg.*

1 https://telegram.org/

2.2 Dokumentation

Dokumentationen spielen eine entscheidende Rolle für den Erfolg eines Projekts. Ohne eine gute Dokumentation ist die Software für die Benutzer als auch Mitwirkenden schwer zugänglich. Mailing Listen und StackOverflow können als eine Ergänzung zur Dokumentation dienen, allerdings kann diese dadurch nicht ersetzt werden [Ban13].

Laut Adam Scott muss eine gute Dokumentation vollständig und einfach zu lesen sein, damit Entwickler jeglichen Erfahrungshintergrunds in der Lage sind die Software zu nutzten. Außerdem sollte die Dokumentation, wenn möglich Beispiele beinhalten, um Nutzern zu ermöglichen Frameworks und Bibliotheken schneller zu verstehen und einzubinden. Des Weiteren sollte die Dokumentation immer auf dem aktuellen Stand der Software sein, da diese sonst für Verwirrung sorgen könnte. [Sco18]

Eine gute Dokumentation ist auch ein Mittel, um neue Nutzer zu gewinnen. Im Artikel von Dagenais et al. heißt es, dass schon das Vorhandensein eines *Getting Started* Tutorials, den Nutzer beim Entscheidungsprozess positiv beeinflussen kann [Dag10]. Sind Nutzer die ersten Schritte mit einer neuen Programmiersprache, Framework oder Bibliothek gegangen, steigt die Wahrscheinlichkeit, dieses Produkt auch zu nutzen. Beispielsweise hat die Web-Bibliothek ReactJS[2] auf der Homepage simple Beispiele, die live editiert werden können, ohne sich vorher etwas herunterladen zu müssen, um einen ersten Eindruck von React zu gewinnen. Ferner gibt es ein sehr ausführliches Tutorial[3] welches über 5 Kapitel mit 21 Unterkapitel alle Grundbausteine von React abdeckt, um das gesamte Feature-Set in Kürze zu präsentieren und neue Entwickler von React zu überzeugen [Rea].

Laut einer GitHub Umfrage im Jahr 2017 sind unvollständige oder verwirrende Dokumentationen das größte Problem für Open Source Nutzer. Eine gute Dokumentation hingegen lädt nicht nur neue Nutzer ein, sondern kann auch Nutzer zu Mitwirkenden machen. Sei es durch das Erstellen von Issues oder eines ersten Pull Requests. Hierfür spielen vor allem Contributing Guides und ein Code of Conduct eine wichtige Rolle, hierzu mehr im nächsten Kapitel [Git17].

Dokumentationen spielen eine wichtige Rolle, um neue Nutzer als auch neue Mitwirkende für ein Projekt zu gewinnen. Daher wird die Hypothese aufgestellt, dass Projekte mit guter Dokumentation, einen höheren Markterfolg haben.

H 3. *Gute Dokumentationen ziehen mehr Nutzer an und führen so zu einem höheren Markterfolg.*

2 https://reactjs.org/
3 https://reactjs.org/docs/getting-started.html

2.3 Die Community und die Projektentwicklung

Ein weiterer Erfolgsfaktor der OSS ist die Community, die sowohl aus Nutzern als auch aus OSS-Entwickler besteht. Es liegt in der Verantwortung der Projektleiter bzw. Projekteigentümer die Community aufzubauen und zu pflegen [Ban13, Gita].

In einem Leitfaden von GitHub wird die Bedeutung der Community und wie diese aufgebaut wird näher erklärt. Hierbei ist vor allem die Rede von einem Code of Conduct und Contributing Guide [Gita]. Das Ziel ist es eine offene und wachsende Community aufzubauen, die dem technischen Erfolg dienen soll.

Das *Code of Conduct* ist ein Dokument, welches die Erwartungen an das Verhalten der Projektteilnehmer festlegt. Das Übernehmen und Durchsetzten des Code of Conducts kann dazu beitragen, eine positive und soziale Atmosphäre für alle zu schaffen [Gitd]. Häufig findet sich hier im Hauptverzeichnis des Projektes die Datei `CODE_OF_CONDUCT.md`. Als Vorlagen dient in der Regel das *Contributor Covenant*[4].

Der *Contributing Guide* ist eine Einführung, für neue Mitwirkende, die sich an dem jeweiligen Projekt beteiligen wollen. Hier finden sich die Anleitungen und Vorlagen für Bug Reports, Feature Requests, vorgehen bei Pull Requests, sowie Richtlinien bezüglich Coding Styles und Testabdeckung [Gitc]. Einige Projekte beginnen ihre `CONTRIBUTING.md` mit einem Dank an den Leser und künftigen Mitwirkenden, wie beispielsweise Chakra-UI[5], mit den Worten *"Thanks for showing interest to contribute to Chakra UI, you rock!"*. Somit wird die Hypothese aufgestellt, dass Projekte mit einem Code of Conduct bzw. Contributing Guide eine höheren technischen Erfolg haben.

H 4. *Das Vorhandensein eines Code of Conduct führt zu einem höheren technischen Erfolg.*

H 5. *Das Vorhandensein eines Contributing Guides führt zu einem höherem technischen Erfolg.*

4 https://www.contributor-covenant.org/
5 https://github.com/chakra-ui/chakra-ui

2.4 Finanzierung und Sponsoren von Open Source

Ein weiterer Faktor, der Nutzer davon überzeugen kann ein gewisses Framework oder eine Bibliothek gegenüber einer anderen zu nutzen, ist die finanzielle Sicherheit eines Projekts. Es wird vermutet, dass Projekte mit Sponsoren beliebter bei Nutzern sind als Projekte ohne. Die Information, ob ein Projekt Sponsoren hat, ist für einen neuen potenziellen Nutzer hierbei in der Regel sehr leicht ersichtlich. VueJS beispielsweise macht sowohl auf der Website als auch in der README.md auf die Sponsoren aufmerksam und bedankt sich für ihre Unterstützung, wie in Abbildung 2.1 zu erkennen ist.

Daraus folgt die Hypothese:

H 6. *Sponsoren haben einen positiven Einfluss auf die OSS-Auswahl, demnach haben gesponserte Projekte einen höheren Markterfolg.*

Gleichzeitig kann die finanzielle Unterstützung der Sponsoren die Produktivität fördern. Die Projekte werden schneller weiterentwickelt und sorgfältiger gewartet. Daraus folgt die Hypothese:

H 7. *Gesponserte Projekte haben einen höheren technischen Erfolg.*

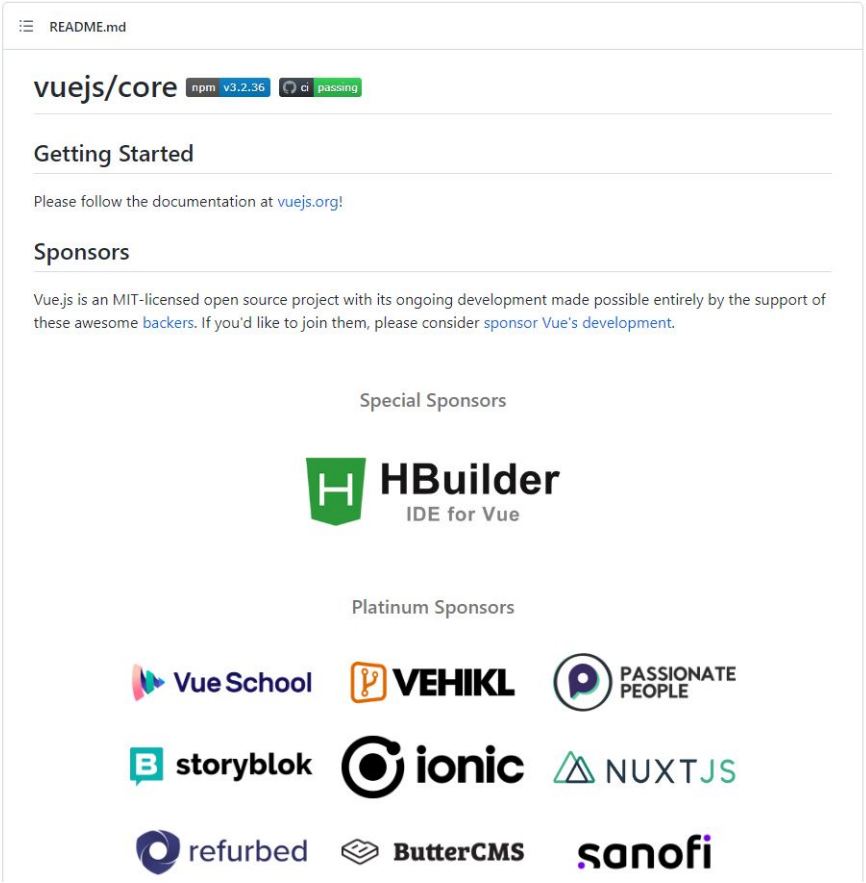


Abbildung 2.1 VueJS Top Sponsoren

2.5 Beliebtheit

Des Weiteren kann die Beliebtheit eines Projekts für die Nutzer eine wichtige Rolle spielen. Midha et al. fanden eine starke Korrelation zwischen der vergangenen Beliebtheit eines Projekts, und der aktuellen Beliebtheit. Der Grund hierfür sei, dass die Beliebtheit als Entscheidungskriterium bei der Auswahl eines Projektes verwendet wird [Mid12].

Subramaniam et al. spreche vom sogenannten *Netzwerkeffekt*, dieser wirkt sich laut [Sub09] positiv auf den Erfolg von OSS aus. Der Begriff Netzwerkeffekt kommt aus der Volkswirtschaftslehre und beschreibt ein Phänomen, bei dem ein Produkt oder eine Dienstleistung einen zusätzlichen Wert erhält, wenn mehr Menschen diesen nutzten. Im Softwareumfeld wird der Netzwerkeffekt in, z.B. Anzahl an Tutorials oder Beiträge auf StackOverflow zeigen. Das wiederum schafft einen stärkeren Anreiz das beliebtere Tool zu nutzten. Im Vergleich von ReactJS und Svelte wie in der Tabelle 2.1 dargestellt wird, zeigt sich, dass ReactJS etwas mehr als 3-mal so viele Sterne auf GitHub hat, aber über 100-mal so viele Beiträge auf StackOverflow. Daraus ergibt sich die Hypothese:

H 8. *Beliebte Projekte werden von Nutzern bei der Wahl von OSS bevorzugt.*

Tabelle 2.1 React vs Svelte

	GitHub Stars	**npm downloads**	**Tutorials**[a]	**StackOverflow Fragen**
React	186k	14.6 mio	438 mio	380k
Svelte	57k	278k	2.3 mio	3k

a Anzahl der Google Ergebnisse

3 Datenerhebung von GitHub Projekten

Das Ziel dieser Datenerhebung ist es die Hypothesen H1, H2, H4, H5, und H8 zu prüfen. Hier wurden 108 Open Source Projekte auf GitHub ausgewählt. Die Studie von Midha et al. hat sich auf die Programmiersprachen C++ beschränkt, ähnlich wird sich auch diese Arbeit auf JavaScript (JS) und TypeScript (TS) beschränken. Grund hierfür ist, dass sich verschiedene Programmiersprachen schlecht miteinander vergleichen lassen [Mid12]. JavaScript wurde für die Datenerfassung ausgewählt, da es zurzeit die Sprache mit den meisten Projekten auf GitHub ist. TS wird mit aufgenommen, da es ein Obermenge von JS ist. TypeScript kann in JS-Scripten verwendet werden, dies hat zur Folge, dass diese als JavaScript kennzeichnen werden aber eigentlich Typescript sind. Aus diesem Grund wird TS explizit mit in die Datenerhebung mit aufgenommen.

Erhoben wurden nur Software-Projekte. Repositories wie E-Books, Tutorials oder Lehrplattformen wurden ausgeschlossen. Projekt Archive wurden ebenfalls ausgeschlossen.

3.1 Manuelle Datenerfassung

Mithilfe der GitHub Suchfunktion[1] wurden JS und TS Projekte ausgewählt. Die Ausgabe wurde nach Anzahl Sternen sortiert wiedergegeben und in dieser Reihenfolge erfasst. Die Ausgabe der ersten Suche beinhaltete überwiegende Mehrzahl der Projekte mit MIT Lizenz. Um eine höhere Diversität des Lizenzen-Milieus zu gewährleisten, wurde im Laufe der Datenerfassung explizit, nach nicht MIT lizenzierten Projekten gefiltert. Zum Erfassen der Daten wurde ein selbst entwickeltes Web-Interface verwendet, um die Erhebung zu vereinfachen. Wie in Abbildung 3.1 zu sehen ist wurden zunächst Identifikationsdaten notiert. Für GitHub wurde *<owner>/<project>* verwendet, wie sie auch in der URL oder auf der Project Page zu finden sind. Für NPM wurde der Package Name verwendet, falls dieser vorhanden war. Applikationen beispielsweise haben in den meisten Fällen kein NPM Package. Des Weiteren wurden auch Niveau der Dokumentation, Vorhandensein von Sponsoren, wer hinter dem Projekt steht und um was für ein Typ Projekt es sich handelt gesammelt. Alle Einträge wurden an ein NodeJS Backend gesendet und als CSV abgespeichert. Im den nächsten Unterkapiteln wird näher erläutert, wie die Kriterien, die von Hand erfasst wurden, zustande gekommen sind und welche Bedeutung die Werte in der CSV-Datei haben.

1 `https://github.com/search/advanced`

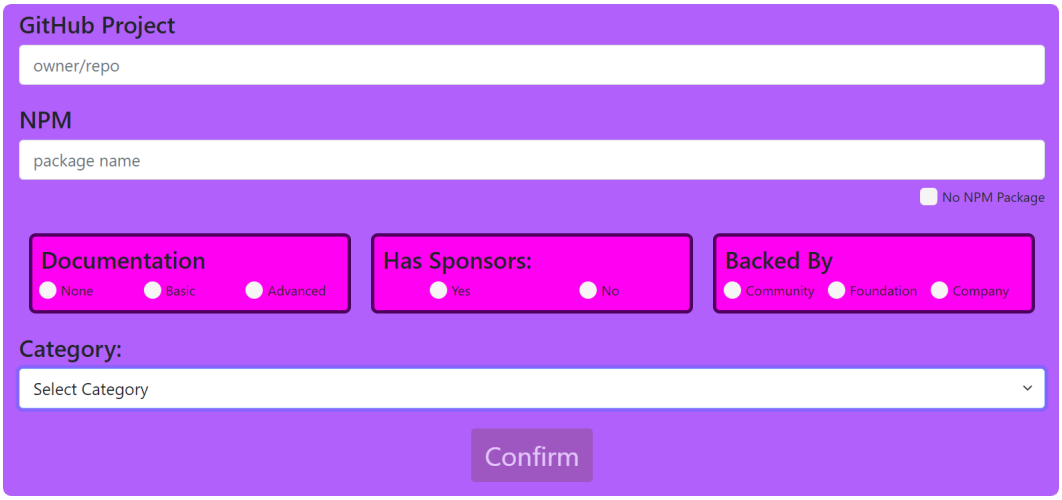


Abbildung 3.1 Web-Interface

3.1.1 Dokumentation

Die Dokumentation des Projektes wurde analysiert und nachfolgenden Kriterien kategorisiert:

0 = keine Dokumentation
1 = basis Dokumentation, rein textuell
2 = Dokumentationen mit Demos oder Live-Beispielen, Einführungsvideos oder ähnliches

3.1.2 Sponsoren

Für jedes Projekt wurde geprüft, ob es Sponsoren hat. Die Anzahl an Sponsoren bzw. die Einnahmen durch Sponsoren wurden nicht beachtet. Das Vorhandensein der Sponsoren wurde wie folgt codiert:

0 = hat keine Sponsoren
1 = hat Sponsoren

3.1.3 Kategorisierung der Mitwirkenden

Im nächsten Schritt wurde vermerkt wer hinter einem Projekt steht, hierbei wurde in drei Kategorien eingeteilt:

0 = Eines reines Community Projekte, welches unabhängig von Unternehmen ist. Beispiel: VueJS.
1 = Von einer Stiftung wie OpenJS[2] unterstütztes oder entwickeltes Projekt.
Beispiel: NodeJS.
2 = Projekte die von Unternehmen entwickelt werden, wie React von Facebook beispielsweise.

Diese Information fanden sich entweder auf der GitHub-Page, Homepage des Projektes oder das Unternehmen ist der Besitzer des Repositories.

2 `https://openjsf.org/`

3.1.4 Projektarten

Im letzten Schritt wurde das Projekt in eins der folgenden Kategorien zugeordnet:

- Utility
- UI
- Application
- Library
- Framework
- Test-Framework
- Open-Core
- API

Erläuterung der Kategorien

- **Utility**, wurden Projekte kategorisiert, die nicht direkt in Projekte eingebaut werden, sondern als Tool verwendet werden. Beispiel: `shelljs/shelljs`[3]
- **Application**, sind eigenständige Produkte wie draw.io[4]

3 `https://github.com/shelljs/shelljs`
4 `https://github.com/jgraph/drawio`

3.2 Automatisierte Datenerfassung

Im zweiten Teil der Erhebung wurden weitere Daten automatisiert gesammelt. Hierfür wurde die GitHub API[5], NPM API[6] und für einige Daten Web-Scraping verwendet.

Zu diesem Zweck wurde eine weitere NodeJS Applikation geschrieben, welche die CSV-Datei aus dem Kapitel 3.1 ausliest und mithilfe der APIs und Web-Scraping eine neue CSV-Datei generiert. In Abbildung 3.2 wird der Prozess der Datenerhebung grafisch dargestellt. In den folgenden Unterkapiteln wird aufgelistet welche Daten mittels welcher Methode gesammelt wurden.

3.2.1 Daten aus der GitHub API

Mittels der GitHub API wurden folgende Daten gesammelt:

- Anzahl der GitHub-Sternen
- Erstellungsdatum des Repositories
- Vorhandensein einer *CODE_OF_CONDUCT.md*
- Vorhandensein einer *CONTRIBUTING.md*
- Lizenz
- Anzahl der Commits in den letzten 12 Monaten
- Anzahl der Issues in den letzten 12 Monaten

3.2.2 Daten aus der NPM API

Mit der NPM API wurden die Downloads der letzten 7 Tage abgefragt.

3.2.3 Daten aus dem Web-Scraping

Mithilfe von Web-Scraping wurden folgende Daten erfasst:

- Die Anzahl von *UsedBy* auf der GitHub Page des Projektes.
- Anzahl der Gesamt-Commits auf dem default Branch
- Anzahl der Mitwirkenden

Anmerkung:

Ein Mitwirkender zählt nur dann als solcher, wenn dessen Commit entweder auf dem *default* oder *gh-pages* Branch liegt. Die Commits auf anderen Branches werden nur dann gezählt, wenn ein merge auf einen der vorherig erwähnten Branches stattfindet. Gleiches gilt für Commits [GHa].

3.2.4 Nachbearbeitung der Daten

Nachdem erheben der Daten war eine Nachbearbeitung notwendig. Einige Projekte hatten die Lizenz `other`. Der Grund hierfür ist, dass die `LICENSE.md` nicht dem Standardtext einer Lizenz entsprochen hat. Ein Beispiel wäre die Lizenz von meteor[7], welche nach dem offiziellen Lizenztext noch eine Anmerkung bezüglich benutzter Bibliotheken hat. Alle als `other` gekennzeichneten Projekte wurden manuell geprüft und korrigiert. Zudem wurden Projekte ohne Lizenzen oder Lizenzen die nicht von der *Open Source*

5 https://docs.github.com/en/rest
6 https://github.com/npm/registry
7 https://github.com/meteor/meteor/blob/devel/LICENSE

Initiative zugelassene sind komplett aussortiert. Nach dem gleichen Prinzip wurden Projekte kontrolliert, die laut API kein Contributing Guide bzw. kein Code of Conduct haben. Grund hierfür ist, dass GitHub das Vorhandensein dieser Dateien nur dann erkennt, wenn diese im Hauptverzeichnis liegen und exakt `CONTRIBUTING.md` bzw `CODE_OF_CONDUCT.md` heißen. Einige Projekte hatten Tippfehler in den Dateinamen oder den Contributing Guide bzw Code of Conduct war Teil der README.md statt eine eigene Datei bzw. auf der Website des Projektes.

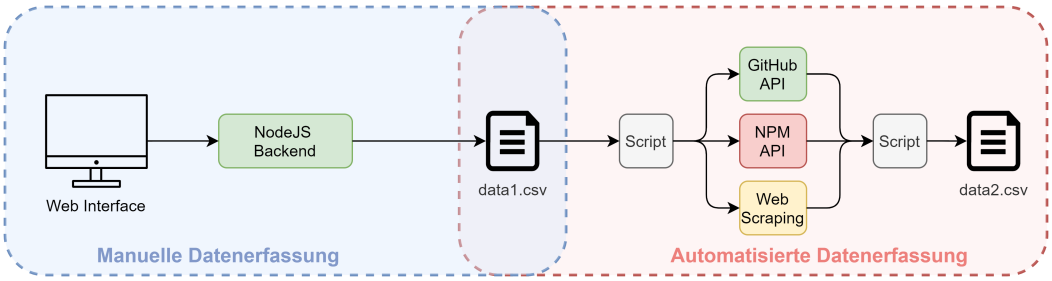


Abbildung 3.2 Prozess der Datenerhebung

4 Ergebnisse der Datenerhebung

Insgesamt wurden 108 Projekte erfasst und analysiert. In den folgenden Unterkapiteln werden die Ergebnisse der Datenerhebung dargelegt. Die Unterkapitel fassen jeweils die Hypothesen zusammen und betrachten die Daten aus dem entsprechenden Blickwinkel.

4.1 Lizenzen

Um die Hypothese H1 zu prüfen wurden der Datensatz in Projekte mit freizügigen und restriktiven Lizenzen. Die Aufteilung erfolgt hierbei, wie bereits im Kapitel 1.1 näher erklärt wurde. In der Tabelle 4.1 findet sich eine Zusammenfassung aller Lizenzen der Projekte, die erfasst wurden. 91% der Projekte haben eine freizügige Lizenz, wobei die MIT Lizenz mit 69% die beliebteste aller Lizenzen ist.

Um die Gruppen miteinander zu vergleichen wurde der Median der Anzahl der Sterne verwendet, da diese sehr stark verteilt ist. Der Median für permissive Lizenzen liegt bei 27.515 Sternen, für restriktive bei 21.595,5. Somit haben Projekte mit permissiven Lizenzen im Median 27,4% mehr Sterne als Projekte mit restriktiven Lizenzen.

Aufgrund der großen Mengenunterscheiden zwischen freizügigen und restriktiven Projekten wurde zusätzlich ein Ranking erstellt. Wie in Abbildung 4.1 zu sehen ist wurden hierfür die Projekte nach Sternen sortiert. Von 108 ausgewerteten Projekten liegt das erste mit restriktiver Lizenz auf Platz 35, das heißt die Top 31% der Projekte sind alle permissiv lizenziert. Projekte mit freizügigen Lizenzen übertreffen die restriktiven sowohl in der Menge als auch in der Beliebtheit.

Um die zweite Hypothese zu prüfen, wurden ebenfalls die Mediane verglichen, allerdings wurden hier Anzahl der Mitwirkenden sowie Commits der letzten 12 Monate betrachtet. Der Median wurde in diesem Fall gewählt, da sowohl die Anzahl Commits als die der Mitwirkenden starke Ausreißer haben.

Im Median haben restriktive Projekte 78 Mitwirkende und 74,5 Commits, permissive Projekte haben 274,5 Mitwirkende und 182 Commits. In einen weiterem Ranking, sortiert nach Mitwirkenden befindet sich das erste Projekt mit restriktiver Lizenz auf Platz 51. Im Ranking nach Anzahl der Commits erreicht, wie in der Abbildung 4.2 zu erkennen ist das erste restriktive Projekt den 9. Platz.

Im Ranking der Anzahl der Commits befinden sich unter den ersten 20 Projekten zwei OSP mit restriktive Lizenzen.

Tabelle 4.1 Lizenzen der erfassten Projekte

Permissive Lizenz	Anz.
MIT	74
Apache 2.0	18
BSD 3-Clause	3
BSD 2-Clause	2
ISC	1
Gesamt	98

Restriktive Lizenz	Anz.
AGPLv3	4
GPLv3	3
LGPLv2.1	1
OSLv3.0	1
LGPLv3	1
Gesamt	10

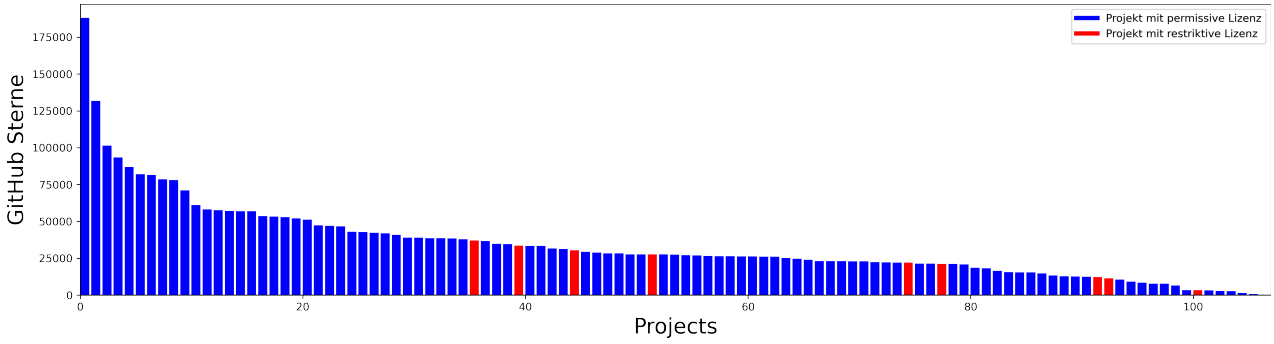


Abbildung 4.1 Effekt von Lizenzen auf GitHub Sterne

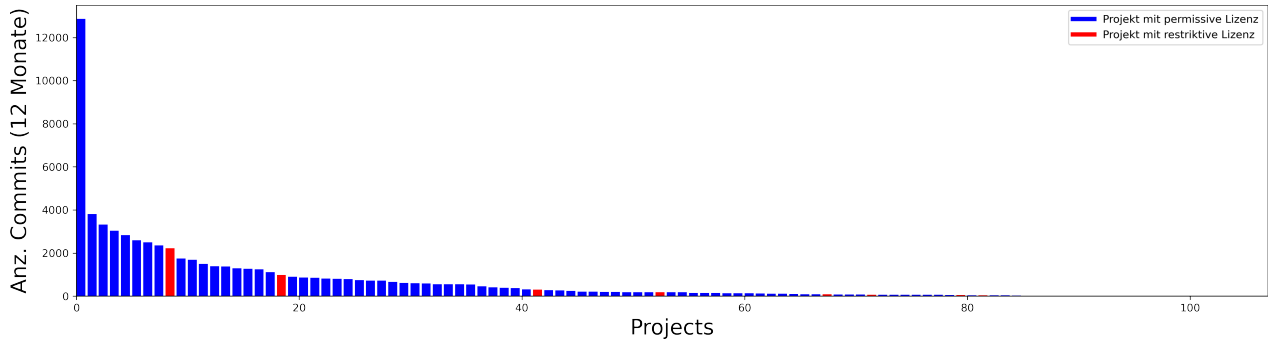


Abbildung 4.2 Effekt von Lizenzen auf Anzahl der Commits

4.2 Code of Conduct

Um Hypothese H4 zu testen wird der Datensatz erneut in zwei Gruppen geteilt, Projekte mit Code of Conduct und Projekte ohne. Verglichen wurden Anzahl der Commits und Mitwirkenden mittels Median.

Wie in Tabelle 4.2 zu erkennen ist, haben 53% der Projekte einen Code of Conduct. Im Vergleich beider Gruppen haben Projekte mit Code of Conduct 717% mehr Commits und 201% mehr Mitwirkender als Projekte ohne.

Tabelle 4.2 Relation von Code of Conduct und Beliebtheit

	Projekte mit Code of Conduct	**Projekte ohne Code of Conduct**
Anzahl	57	51
Median Commits	556	68
Median Mitwirkende	313	104

4.3 Contributing Guide

Wie im Kapitel 4.2 wurde auch hier der Datensatz in zwei Gruppen unterteilt. Projekte mit und ohne einen Contributing Guide (CG). Um Hypothese H5 zu testen soll der Zusammenhang zwischen dem Vorhandensein eines CG und Anzahl der Mitwirkenden bzw. Commits erforscht werden.

Anders als beim Code of Conduct, haben hier 86% aller Projekte einen CG. Verglichen wurden erneut der Median der Anzahl der Mitwirkenden und Anzahl der Commits, Projekte mit einem CG haben 463% mehr Commits und 412% mehr Mitwirkende.

Zusätzlich werden die Projekte nach Anzahl der Mitwirkenden (sieh Abbildung 4.3) sortiert, wie zu erkennen ist haben die Projekte mit den meisten Mitwirkenden alle einen CG. Das erste Projekt ohne belegt hier Platz 54. Nach dem gleichen Schema wurden auch nach Anzahl der Commits sortiert, das erste Projekt ohne CG belegt Platz 42. In beiden Vergleichen platzieren sich Projekte mit CG weit vor Projekten ohne.

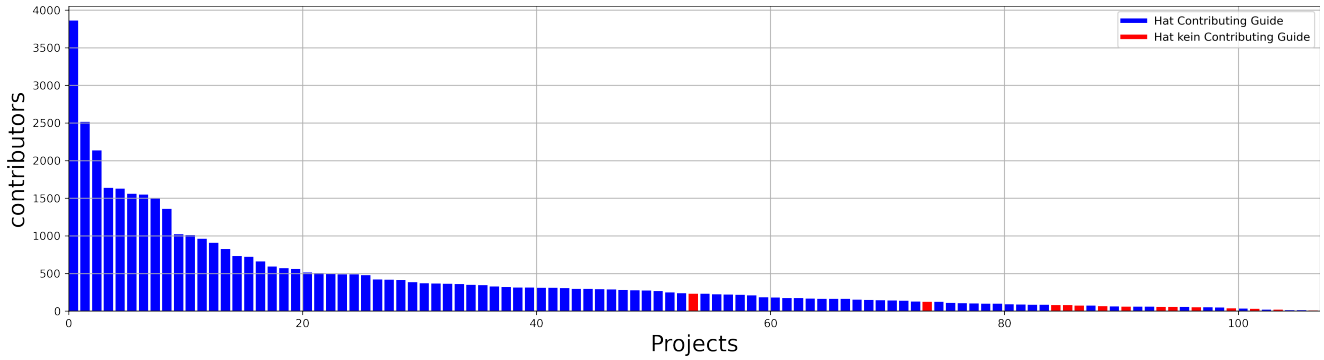


Abbildung 4.3 Einfluss von Contributing Guide auf Mitwirkende

4.4 Einfluss von Sponsoren auf den technischen Erfolg

Erneut wurde der Datensatz in zwei Gruppen geteilt, Projekte mit und ohne Sponsoren. Verglichen wurde der Median der Commits der letzten 12 Monate sowie die Anzahl an Mitwirkenden (sieh Tabelle 4.3). 52% der Projekte haben Sponsoren. Projekte mit Sponsoren haben 72% mehr Commits und 86% mehr Mitwirkende.

Tabelle 4.3 Relation von Sponsoren und Erfolg

	Projekte mit Sponsoren	Projekte ohne Sponsoren
Anzahl der Projekte	56	52
Median Commits der letzten 12 Monate	221	128
Median Anz. Mitwirkenden	289	155

5 Umfrage

Als Ergänzung zur Datenerfassung der GitHub Projekte wurde auch eine Umfrage durchgeführt. Ziel der Umfrage war es herauszufinden worauf die Nutzer von Open Source achten, wenn sie ein Projekt wählen. Das übergeordnete Ziel war es die Hypothesen H3, H6 und H8 zu prüfen. An der Online-Umfrage haben Softwareentwickler der adesso SE, sowie Studenten der TH Rosenheim teilgenommen. Insgesamt erhielt die Umfrage 308 Antworten. Zur Erstellung der Umfrage wurde *cryptpad.org* verwendet [Cry].

Hierbei sollten die Teilnehmer Aussagen zu den Themen Dokumentation, Beliebtheit, Sponsoren und Entwicklung bewerten. Als Bewertungsskala wurde eine 5-Punkte Likert-Skala verwendet [Lik32].

5.1 Dokumentation

Im ersten Teil der Umfrage mussten die Teilnehmenden zu zwei Aussagen verschiedene Kriterien bewerten. Diese Kriterien wurden gewählt basierend auf Erfahrung als Entwickler als auch Beobachtungen vieler Dokumentationen während der manuellen Datenerfassung zu aus Kapitel 3.1.1.

Die erste Aussage war *"Wie wichtig sind Ihnen folgende Aspekte der OSS-Dokumentation"*, mit den Punkten:

- Übersichtlichkeit
- Einfache Sprache
- Live-Demos
- Übersetzungen (z.B. Englisch -> Deutsch)
- Das Vorhandsein einer Getting Started Page
- Code Beispiele
- Strukturierung der Dokumentation

Die zweite Aussage war *"Die folgenden Punkte würden mich von der Nutzung eines OSS-Projekts abhalten oder mich möglicherweise dazu veranlassen, nach Alternativen zu suchen."*, mit den Punkten:

- Keine Code Beispiele
- Schlecht strukturierte Dokumentation
- Dokumentation zu komplex
- Fehlende Übersetzung (z.B. fehlende deutsche Übersetzung)
- Fehlende Getting Started Page

Die Aussagen wurden auf einer 5-Punkte Skala bewertet, hierbei entspricht 1 *Stimme gar nicht zu* und 5 *Stimme vollkommen zu*. Um die Aussagen miteinander zu vergleichen, wurde jeweils der Mittelwert berechnet.

Die zwei wichtigsten Aspekte einer Guten Dokumentation sind laut den Befragten, Code Beispiele und Übersichtlichkeit in beiden Fällen mit einem durchschnittlichen Wert von 4,36. Gefolgt von guter Struktur (4,09) und das Vorhandensein eines *Getting Started* (3,98). Einfache Sprache (3,08) und Live-Demos spielt für die Befragten eine weniger wichtige Rolle. Übersetzungen belegen mit 1,53 den letzten Platz und spielt somit die unwichtigste Rolle in einer guten Dokumentation.

Die zweite Aussage war *"Die folgenden Punkte würden mich von der Nutzung eines OSS-Projekts abhalten oder mich möglicherweise dazu veranlassen, nach Alternativen zu suchen."* Am höchsten bewertet wurde der

Punkt *Fehlende Code Beispiele* (4,07) und ist somit der wichtigste Teil einer guten Dokumentation, gefolgt von schlechter Struktur (3,91). Eine komplexe Dokumentation oder das Fehlen einer Getting Started Seite wurde hierbei als mittelmäßig kritisch eingestuft, mit einer durchschnittlichen Bewertung von 3,29 bzw. 3,10. Das Fehlen einer Übersetzung ist mit 1,37 das unwichtigste Kriterium. In den Abbildungen 5.1 und 5.2 wird das Meinungsbild der Teilnehmer grafisch dargestellt.

Zusammenfassend sind Code Beispiele eine Notwendigkeit für eine gute Dokumentation. Das Vorhandensein wird als wichtig eingestuft, während das Fehlen dazu führen könnte, dass Nutzer zu alternativen greifen würden. Übersetzungen hingegen werden als gleichgültig betrachtet, das Vorhandensein bietet wenig Mehrwert, die Abwesenheit wird nicht als kritisch betrachtet.

Beide Fragen hatten jeweils ein Freitextfeld, indem die Teilnehmenden die Möglichkeit hatten weitere Aspekte einer guten bzw. schlechten Dokumentation zu nennen. Diese Freitextfelder waren optional und wurden von 95 bzw. 71 der insgesamt 308 Teilnehmenden genutzt. Alle Einträge wurden kategorisiert und zusammengefasst.

In der Tabelle 5.1 finden sich die am häufigsten genannten Punkte. Aktualität, Vollständigkeit und UX waren die am meisten genannten Eigenschaften bezüglich guter Dokumentation, die jeweiligen Pendants, veraltete Dokumentation, unvollständige Dokumentation und schlechte UX waren die häufigsten genannten Gründe, um ein OSS Projekt nicht zu nutzen. Diese drei Merkmale vervollständigen somit die vorhin erwähnten *Must-Haves* einer guten Dokumentation.

Tabelle 5.1 Häufigste Erwähnungen der Freitexter

Gute Dokumentation	Erwähnungen
Aktualität	28
Vollständigkeit	21
Gute UX	14
Versionierung	10
Changelog vorhanden	6
Gute Code Beispiele	5

Schlechte Dokumentation	Erwähnungen
Veraltet	25
Unvollständig	15
Schlechte UX	11
Fehlerhaft	7
Keine Doku vorhanden	6
Code Beispiele funktionieren nicht	5

Anmerkung:

Im Fall von UX wurden folgende Punkte zusammengefasst: Suchfunktion/Verlinkung, Design und Übersichtlichkeit.

Die abschließende Frage zum Thema Dokumentation war: *"Hat eine schlechte Dokumentation Sie jemals dazu gebracht, ein alternatives Projekt zu wählen?"*. 81% der Teilnehmen haben diese Frage mit *"Ja"* beantwortet.

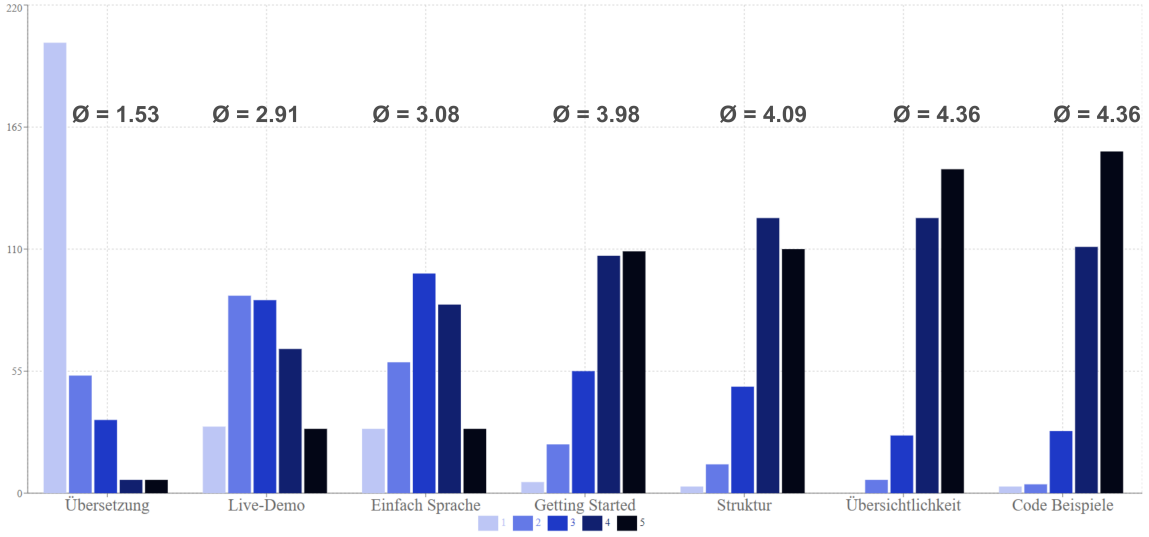


Abbildung 5.1 Antworten: Was zeichnet gute Dokumentation aus?

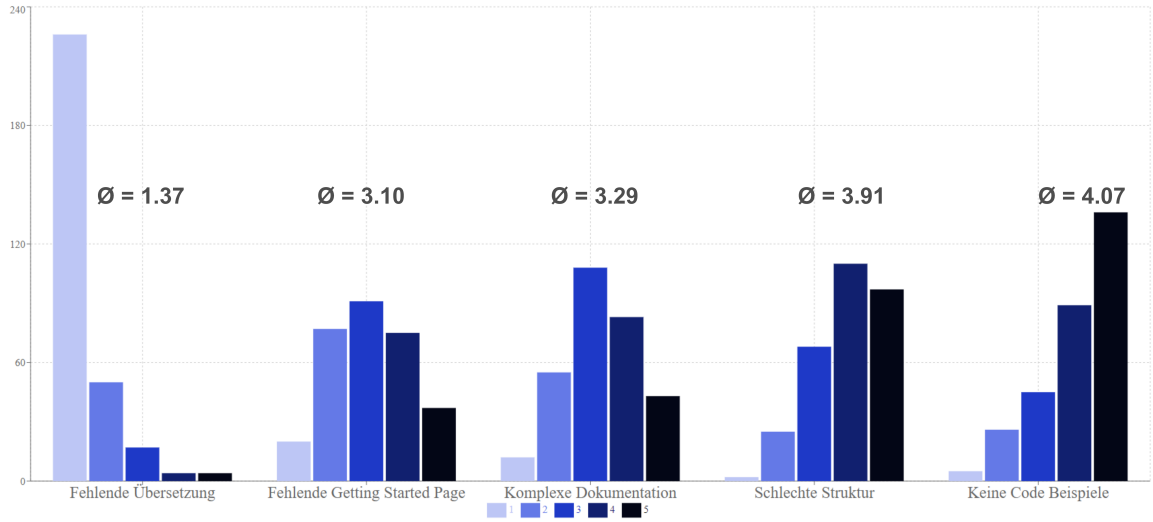


Abbildung 5.2 Antworten: Was zeichnet schlechte Dokumentation aus?

5.2 Beliebtheit

Im zweiten Teil der Umfrage, soll geklärt werden, welchen Einfluss die Beliebtheit, bei der Wahl eines Projektes hat. Wie zuvor auch sollten die Teilnehmer Kriterien auf einer 5-Punkte Skala bewerten. Die Mittelwerte der Antworten finden sich in Tabelle 5.2 wieder.

Die Aussage war *"Wie sehr achten Sie bei der Auswahl von OSS auf..."*, mit den Punkten:

- Anzahl der GitHub Sterne
- Anzahl der Mitwirkenden eines Projektes
- Anzahl von Sponsoren
- Trends
- Anzahl an Fragen und Antworten auf StackOverflow

Diese Punkte wurden gewählt, da es klassischen Vergleichsmerkmalen sind, um die Beliebtheit von OSS zu vergleichen. GitHub Sterne und Downloads sind eine schnelle und einfache Methode, um die aktuelle Beliebtheit eines Projektes zu vergleichen. Trends hingegen stellen GitHub Sterne oder Downloads über Zeit dar, häufig genutzte Tools für die Analyse von Trends sind *StackOverflow Trends*[1], *npm trends*[2] oder *Google Trends*[3].

Anders als bei der Dokumentation gibt es hier kein Kriterium mit starker Zustimmung. Die Downloads sind hierbei mit einer durchschnittlich Zustimmung von 3,27 die beliebtes Vergleichsmetrik für Beliebtheit gefolgt von Sternen (2,92). Anzahl der Mitwirkenden (2,68), StackOverflow Fragen und Antworten (2,60) sowie Trends (2,23) werden tendenziell weniger beachtet. Die Anzahl der Sponsoren wird mit einer durchschnittlichen Bewertung (1,7) fast gar nicht beachtet.

Des Weiteren gaben 46% der Teilnehmer an, dass die geringer Popularität eines Projekts sie schon mal davon abgehalten hat dieses zu nutzten.

Tabelle 5.2 Einfluss der Beliebtheitsmerkmalen bei der Wahl von OSS

Downloads	Sterne	Mitwirkende	StackOverflow Fragen/Antworten	Trends	Sponsoren
3.27	2.92	2.68	2.60	2.23	1.7

5.3 Sponsoren

Im dritten Teil der Umfrage sollte herausgefunden werden, welchen Einfluss Sponsoren bei der OSS Wahl spielen. Auch hier wurde mit der 5-Punkte Skala bewertet, mit der Frage: *"Bewerten Sie folgende Aussagen."* Die Aussagen waren:

1. Projekte mit Sponsoren wirken zukunftssicherer
2. Ich bevorzuge es, wenn möglich, Projekte mit Sponsoren zu nutzten
3. Gesponserte Projekte sind meistens qualitativ besser

Mit einer sehr schwachen Zustimmung von durchschnittlich 3,27 gaben die Teilnehmer an, dass sie Projekte mit Sponsoren als zukunftssicherer empfinden. Projekte mit Sponsoren werden tendenziell weder stark bevorzugt (2,92) noch als qualitativ hochwertiger empfunden (2.68).

1 https://insights.stackoverflow.com/trends
2 https://www.npmtrends.com/
3 https://trends.google.de/trends/

Tabelle 5.3 Einfluss von Sponsoren

Aussage:	1.	2.	3.
	3.27	2.92	2.68

5.4 Development

Im vorletzten Teil der Umfrage ging es um den Einfluss des Entwicklungsprozesses bei der Auswahl von OSS. Die Frage war: *"Wie sehr achten Sie auf die folgenden Punkte, wenn Sie ein Projekt wählen?"*

- Anzahl aktiver Maintainer
- Ticket/Issue Verhältnis Open/Closed
- Antwortzeit der Entwickler auf Tickets/Issues
- Regelmäßigkeit der Commits
- Aktualität des letztes Commit

Diese Punkte wurden gewählt, da es leicht einsehbare Daten sind, die den Entwicklungsprozess eines Projektes widerspiegeln.

Ähnlich wie bei der Dokumentation legen die Nutzer einen große Wert auf Aktualität (vgl. Tabellen 5.1) Mit einer durchschnittlichen Zustimmung von 3,94 ist diese das wichtigste Kriterium des Entwicklungsprozesses, gefolgt von Anzahl der Maintainer (3,33), Regelmäßigkeit der Commits (3,28), Verhältnis von offenen und abgeschlossenen Tickets (2,90) und Antwortzeit der Entwickler auf Tickets (2,86).

Tabelle 5.4 Einfluss des Entwicklungsprozesses bei der Wahl von OSS

Aktualität	Anzahl der Maintainer	Regelmäßigkeit	Verhältnis	Antwortzeit
3.94	3.33	3.28	2.90	2.86

5.5 Freie Kategorisierung der Erfolgskriterien

Im letzten Teil der Umfrage sollten die Teilnehmer verschiedene Kriterien per *Drag and Drop* sortieren. Die Fragestellung lautete *Sortieren Sie nach den für Sie wichtigsten Kriterien bei der Auswahl von OSS.* mit folgenden Punkten zum Sortieren

- Gute Dokumentation
- Beliebtheit
- Anzahl von offenen Issues/Tickets
- Trends
- Projekt hat Sponsoren

Aufgrund der Übersichtlichkeit hat sich die Auswahl hier auf fünf beschränkt. Es wurde versucht alle vorherigen Themen abzudecken. Je nach Platzierung, haben die Kriterien Punkte bekommen, 5 Punkte für den ersten Platz, 4 für den zweiten usw. der letzte Platz bekam einen Punkt.

Das wichtigste Kriterium ist Gute Dokumentation mit 1329 Punkten, gefolgt von Beliebtheit (1165), Anzahl der offenen Issues/Tickets (794), Trends (624), Projekt hat Sponsoren (559).

6 Diskussion

Das Ziel dieser Bachelorarbeit war es, zu untersuchen welche Faktoren Einfluss auf den Erfolg eines Open Source Projektes haben. Um die Hypothesen dieser Arbeit zu prüfen wurden 108 Projekte analysiert sowie 308 Studenten und adesso Mitarbeiter befragt. Anhand der Daten wurden sieben Hypothesen belegt und eine wurde verworfen. In den folgenden Unterkapiteln werden diese Ergebnisse diskutiert.

6.1 Der Einfluss von permissiven Lizenzen auf den Erfolg

In Kapitel 2.1 wurde die Hypothese aufgestellt, dass permissive Lizenzen einen positiven Effekt auf den Markterfolg haben. Um diese Hypothesen zu prüfen wurde eine Datenerhebung mit 108 Projekten durchgeführt. Als Metrik des Markterfolges wurden GitHub Sterne verwendet. Wie die Ergebnisse aus Kapitel 4.1 zeigen, haben Projekte mit permissiven Lizenzen 27,4% mehr Sterne zusätzlich sind die Top 32% Projekte permissiv lizenziert. Und das trotz expliziter Suche nach restriktiven Projekten, wie in Kapitel 3.1 beschrieben. Bei der manuellen Datenerfassung war es zusätzlich notwendig explizit nach restriktiven Projekten zu suchen, um den Anteil dieser Projekte zu erhöhen. Das zeigt zum einen, dass permissive Lizenzen allgemein bevorzugt werden und permissiv lizenzierte Projekte auch allgemein erfolgreicher sind. Somit bestätigt sich Hypothese H1. Ursprünglich sollten auch die NPM Downloads als Beliebtheitsmerkmalen verwendet und verglichen werden, allerdings sind nur 4 der 10 restriktiven Projekte auf NPM.

Des Weiteren wurde die Hypothese aufgestellt, das permissive Lizenzen einen positiven Einfluss auf den technischen Erfolg haben. Zum Prüfen der Hypothese wurden die Anzahl der Commits in den letzten 12 Monaten und Anzahl der Mitwirkenden verglichen. Projekte mit permissiven Lizenzen haben im Median sowohl mehr Commits (252%) als auch mehr Mitwirkende (144%). Somit bestätigt sich die Hypothese H2, permissive Projekte sind technisch erfolgreicher.

Dieser Arbeit hat sich nur mit den Projekten beschäftigt, die hauptsächlich in JavaScript/Typescript programmiert sind. Das JS/TS Umfeld ist hierbei dominiert von permissiven Lizenzen, dies hat zur Folge gehabt, das von den 108 Projekten nur 10 Projekte restriktiv lizenziert sind. Diese Ungleichverteilung erschwert generalisierbare Aussagen bezüglich der Lizenzen.

Die Ergebnisse bestätigen die Erkenntnisse von Subramaniam et al. in dem Punkt, dass permissive Lizenzen einen positiven Effekt auf den Erfolg von Projekten haben [Sub09]. Über den Effekt von restriktiven Lizenzen lassen sich hier keine äquivalenten Aussagen treffen, aufgrund kleiner Datenmengen mit restriktiven Lizenzen.

6.2 Der Einfluss einer guten Dokumentation auf den Markterfolg

Im Kapitel 2.2 wurde die Hypothese aufgestellt, dass gute Dokumentationen mehr Nutzer anziehen und somit zu einem höheren Markterfolg führen würde. Um diese Hypothese zu prüfen wurde eine Umfrage durchgeführt. Die Umfrage hatte 308 Teilnehmer. Die Teilnehmenden sollten auf einer Likert Skala Aussagen zustimmen oder ablehnen, hier hatte das Thema Dokumentation die höchste Zustimmung

verglichen mit den anderen Themengebieten der Umfrage, dies zeigt ein allgemein hohes Interesse an der Dokumentation. Diese Aussage wird von der Tatsache untermauert, das wie in Kapitel 5.5 gezeigt wird, die Dokumentation das wichtigste Kriterium, bei der Auswahl von OSS ist. Des Weiteren gaben 81% der Befragten an, aufgrund schlechter Dokumentation schon mal zu einem alternativen Projekt gewechselt zu haben. Das zeigt die Wichtigkeit einer guten Dokumentation und bestätigt somit die Hypothese H3.

Die Freitextfelder haben gezeigt das die Kriterien der geschlossenen Fragen, die es zu bewerten galt nicht vollständig ware. Vor allem Aktualität und Vollständigkeit sind weitere wichtige Eigenschaften der Dokumentationen, die die Befragten angaben. Für weitere Forschung wäre daher sinnvoll alle als wichtig bewerteten Kriterien, zu untersuchen. Ein weiterer Aspekt der in Bezug auf die Dokumentation von den Befragten häufig erwähnt wurde, war die UI/UX. Auch diese Kriterien könnten für weitere Untersuchungen relevant sein.

In einer Umfrage der Open Source Survey aus 2017 stellte sich heraus, dass unvollständige bzw. verirrende Dokumentation als größtes Problem in Open Source gelten [Git17]. Diese Arbeit bestätigt diese Ergebnisse, und erweitert sie. Neben der Komplexität und Unvollständigkeit sind die wichtigste Punkte einer guten Dokumentation, Code Beispiele und Aktualität.

6.3 Der Einfluss vom Code of Conduct und Contributing Guide auf den technischen Erfolg

Im Kapitel 2.3 wurden Hypothesen H4 und H5 aufgestellt, basierend auf den Open Source Guide von GitHub [Gita]. Die Hypothese H4 beschäftigt sich mit dem Effekt des *Code of Conducts* auf den technischen Erfolg. Hypothese H5 hingegen beschäftigt sich mit dem Effekt des *Contributing Guides* auf den technischen Erfolg. Zum Prüfen beider Hypothesen wurden die Daten aus der GitHub Datenerhebung verwendet. Ein Code of Conduct haben 53% der 108 analysierten Projekte, einen Contributing Guide 86%. Gemessen wurde der technische Erfolg anhand der Commits und Anzahl der Mitwirkenden. Projekte mit einem Code of Conduct haben deutlich mehr Commits (717%) und Mitwirkende (201%). Projekte mit einem Contributing Guide haben 463% mehr Commits und 412% mehr Mitwirkende als Projekte ohne. Damit sind beide Hypothesen belegt.

Eine mögliche Erklärung für die Größe der Effekte, kann damit begründet werden, dass es sich für kleine Projekte nicht lohnt, eine Code of Conduct einzuführen, da die Größe der Teams zu klein ist. Im Fall des Contributing Guides, könnte die ungleiche Verteilung in den Daten die Ursache sein.

Es zeigt sich allerdings deutlich, dass die Empfehlungen von GitHub auch durchgesetzt werden, vor allem in großen Projekten.

6.4 Der Einfluss der Sponsoren auf den Erfolg

Im Kapitel 2.4 wurde die Hypothese aufgestellt, dass Sponsoren einen positiven Einfluss bei der Wahl von OSS hätten. Die Hypothese wurde mittels Umfrage getestet. Wie in Kapitel 5.2 bereits gezeigt wurde, achten die Nutzer sehr wenig auf das Vorhandensein von Sponsoren. In Kapitel 5.3 ist ein möglicher Grund dafür erkennbar. Gesponserte Projekte werden weder bevorzugt noch als qualitativ hochwertiger betrachtet, das heißt Nutzer haben wenig Anlass um auf Sponsoren zu achten. Hypothese 6 wurde somit verworfen.

Bezüglich Sponsoren wurde zusätzlich noch die Hypothese aufgestellt, dass Sponsoren sich positiv auf den technischen Erfolg auswirken würden. Diese Hypothese wurde mittels der GitHub Datenerhebung geprüft. Wie in Kapitel 4.4 dargelegt, haben Projekte mit Sponsoren 72% mehr Commits und 86% mehr Mitwirkende als Projekte ohne Sponsoren. Entsprechend wird die Hypothese 7 belegt.

Zwar hat die Umfrage gezeigt, dass die Nutzer nicht auf die Sponsoren achten, allerdings achten sie auf die Aspekte des technischen Erfolgs, welcher wiederum durch die Anwesenheit der Sponsoren unterstützt wird. Die Beziehung zwischen all diesen Faktoren sollte in künftigen Studien untersucht werden, die genaue Effektstärke der Sponsoren im Projekt aufzuklären.

6.5 Der Einfluss von Beliebtheit auf den Markterfolg

In Kapitel 2.5 wurde die Hypothese aufgestellt, dass Beliebtheit zu mehr Markterfolg führt. Die Hypothese wurde als Teil der Umfrage getestet. Die Beliebtheit ist laut Umfrage das zweit wichtigste Kriterium bei der Wahl von OSS. Insgesamt gaben 46% der Befragten an, aufgrund der geringer Popularität eines Projekts es nicht genutzt zu haben. Es wurde erwartet, das die Nutzer die Beliebtheit höher gewichten würden. Dennoch führt die geringe Popularität bei fast der Hälfte der Befragten dazu ein Projekt nicht zu verwenden. Angesichts diesen hohen Anteils gilt die Hypothese H8 als belegt.

Dass die Nutzer die Beliebtheit als nur zweitwichtigstes Kriterium eingestuft haben kann vor allem durch höhere Gewichtung der Dokumentation in einem Projekt erklärt werden. Weitere Forschung könnte die Unterkategorien der Beliebtheit umfangreicher analysieren, um genauer feststellen zu können welche Aspekte der Beliebtheit für die Nutzer größere Rollen spielen.

6.6 Weitere Beschränkungen der Arbeit

Zu den wesentlichen Limitationen der Untersuchung zählt die mangelnde Überprüfung der Beziehung der einzelnen Erfolgskriterien zueinander. Es wurde nicht aufgeklärt ob es mögliche Überlappungen zwischen den jeweiligen Variablen gibt. Des Weiteren wurden nur Projekte in zwei Programmiersprachen (JS/TS) zur Analyse verwendet. Aus diesem Grund wäre es für die weitere Forschung betrachtenswert auch Projekte in anderen Programmiersprachen zu untersuchen und mit Erkenntnissen dieser Arbeit zu vergleichen.

7 Fazit

Das Ziel dieser Arbeit war es den Einfluss von **Lizenzen, Dokumentation, Code of Conduct, Contributing Guide, Beliebtheit und Vorhandensein von Sponsoren** zu erforschen. Um dieses Ziel zu erreichen wurden insgesamt acht Hypothesen aufgestellt. Mithilfe einer Umfrage mit 308 Teilnehmern sowie einer Datenerhebung von 108 Github Projekten konnten sieben diese Hypothesen belegt werden.

Die Umfrage hat gezeigt, dass Dokumentation das wichtigste Entscheidungskriterium der Nutzer bei der Auswahl von Open Source Software ist. Die wichtigsten Aspekte einer guten Dokumentation sind hierbei Übersichtlichkeit, Code Beispiele sowie eine Getting Started Seite. Weitere Eigenschaften sind Aktualität und Vollständigkeit. Die Umfrage kam auf ähnliche Ergebnisse wie [Git17], und bestätigt die Aussagen von [Sco18] Die Umfrage hat auch gezeigt, dass Beliebtheit das zweit wichtigste Entscheidungskriterium ist. Anzahl der Downloads und die der GitHub Sterne werden hierbei am häufigsten verwendet. Diese Erkenntnis deckt sich mit der von [Mid12]. Die Datenerhebung der 108 GitHub Projekte hat gezeigt, dass permissive lizenzierte Projekte einen höheren Markt- als auch technischen Erfolg zu verzeichnen haben. Aufgrund der starken Ungleichverteilung der Lizenzen Gruppen, lässt sich allerdings keine allgemeingültigen Schlussfolgerungen ziehen. Wie die Datenerhebung und Umfrage gezeigt haben, sind Projekte mit Sponsoren technisch erfolgreicher, als Projekte ohne. Allerdings haben Sponsoren keinen Einfluss auf den Markterfolg. Sponsoren werden von potenziellen Nutzern fast nicht beachtet.

Weitere Forschung ist nötig, um den Zusammenhang der von Lizenzen und Erfolg zu erforschen. Hierbei muss darauf geachtet werden, dass restriktive und permissive Projekte besser gleichverteilt sind. Da Dokumentation sich als wichtigstes Kriterium des Open Source Erfolgs erwiesen hat, wäre es empfehlenswert diesen Erfolgsfaktor in weiterer Studien zu untersuchen.

Literaturverzeichnis

[Ban13] W. Bangerth und T. Heister. What Makes Computational Open Source Software Libraries Successful? *Computational Science & Discovery*, 6(1):015010, Nov. 2013.

[Bra] About Branches. https://ghdocs-prod.azurewebsites.net/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-branches. Zuletzt aufgerufen am 07.06.2022.

[Cry] Cryptpad.org. https://cryptpad.org/. Zuletzt aufgerufen am 05.06.2022.

[Dag10] B. Dagenais und M. P. Robillard. Creating and Evolving Developer Documentation: Understanding the Decisions of Open Source Contributors. In *Proceedings of the Eighteenth ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE '10, S. 127–136. Association for Computing Machinery, New York, NY, USA, Nov. 2010.

[Del03] W. H. Delone und Ephraim R. McLean. The DeLone and McLean Model of Information Systems Success: A Ten-Year Update. *Journal of Management Information Systems*, 19(4):9–30, 2003.

[GHa] Why Are My Contributions Not Showing up on My Profile? https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-profile/managing-contribution-graphs-on-your-profile/why-are-my-contributions-not-showing-up-on-my-profile#commit-was-not-made-in-the-default-or-gh-pages-branch. Zuletzt aufgerufen am 18.05.2022.

[Gita] Building Welcoming Communities. https://opensource.guide/building-community/. Zuletzt aufgerufen am 02.04.2022.

[Gitb] Git - Gitglossary Documentation. https://git-scm.com/docs/gitglossary. Zuletzt aufgerufen am 07.06.2022.

[Gitc] Starting an Open Source Project. https://opensource.guide/starting-a-project/. Zuletzt aufgerufen am 07.04.2022.

[Gitd] Your Code of Conduct. https://opensource.guide/code-of-conduct/. Zuletzt aufgerufen am 07.04.2022.

[Git17] Open Source Survey. https://opensourcesurvey.org/2017/, 2017. Zuletzt aufgerufen am 03.04.2022.

[Ler02] J. Lerner und J. Tirole. Some Simple Economics of Open Source. *The Journal of Industrial Economics*, 50(2):197–234, 2002.

[Lik32] R. Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[Mar15] D. Margan und S. Čandrlić. The success of open source software: A review. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, S. 1463–1468. 2015.

[Mid12] V. Midha und P. Palvia. Factors Affecting the Success of Open Source Software. *Journal of Systems and Software*, 85(4):895–905, Apr. 2012.

[Net] Understanding the Network Effect. https://www.investopedia.com/terms/n/network-effect.asp. Zuletzt aufgerufen am 23.04.2022.

[Ope07] The Open Source Definition | Open Source Initiative. https://opensource.org/osd, März 2007. Zuletzt aufgerufen am 15.03.2022.

[Ope15] Open Source License Usage on GitHub.Com. https://github.blog/2015-03-09-open-source-license-usage-on-github-com/, März 2015. Zuletzt aufgerufen am 30.05.2022.

[Ope22] Open Source Journey for Global Companies 2018-2021. https://www.statista.com/statistics/1245975/worldwide-open-source-process-organizations/, Jan. 2022.

[Pul] About Pull Requests. https://ghdocs-prod.azurewebsites.net/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests. Zuletzt aufgerufen am 07.06.2022.

[Rea] React Tutorial. https://reactjs.org/tutorial/tutorial.html. Zuletzt aufgerufen am 07.06.2022.

[Sco18] A. Scott. The Eight Rules of Good Documentation. https://www.oreilly.com/content/the-eight-rules-of-good-documentation/, Apr. 2018. Zuletzt aufgerufen am 06.06.2022.

[Sta] Stack Overflow Developer Survey 2021. https://insights.stackoverflow.com/survey/2021/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2021. Zuletzt aufgerufen am 20.03.2022.

[Ste06] K. J. Stewart, A. P. Ammeter und L. M. Maruping. Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects. *Information Systems Research*, 17(2):126–144, 2006.

[Sub09] C. Subramaniam, R. Sen und M. L. Nelson. Determinants of Open Source Software Project Success: A Longitudinal Study. *Decision Support Systems*, 46(2):576–585, Jan. 2009.

[Top] Top GitHub Projects by Stars. https://github.com/search?l=&o=desc&q=stars%3A%3E100+stars%3A%3E1&s=stars&type=Repositories.

[Tur21] J. Turner. Open Source Has a Funding Problem. https://stackoverflow.blog/2021/01/07/open-source-has-a-funding-problem/, Jan. 2021. Zuletzt aufgerufen am 22.04.2022.

[W3T22] Usage Statistics and Market Share of Web Servers, February 2022. https://w3techs.com/technologies/overview/web_server, Febr. 2022. Zuletzt aufgerufen am 11.02.2022.

A Anhang

A.1 Erhobene GitHub Daten

Die erhobenen GitHub Daten werden aufgeteilt auf 6 Seiten. Teil 1 bis Teil 3. Wobei jeder Teil nochmal in 2 Subparts unterteilt ist, da die Tabellen mit allen Attributen sehr breit sind. Die *Parts* ergänzen die Tabelle um Spalten, die *Teile* ergänzen die Tabelle, um Zeilen.

Würde man die Tabellen nebeneinander legen wäre sie wie folgt zu lesen:

Teil 1, Part 1	Teil 1, Part 2
Teil 2, Part 1	Teil 1, Part 2
Teil 3, Part 1	Teil 3, Part 3

Teil 1, Part 1

	GitHub_Project	NPM_package	Documentation	Sponsors	BackedBy	Category	created_at	stars
0	axios/axios	axios	2	1	0	Library	2014-08-18T22:30:27Z	93419
1	mrdoob/three.js	three	2	1	0	UI	2010-03-23T18:58:01Z	82050
2	angular/angular	@angular/core	2	0	2	Framework	2014-09-18T16:12:01Z	81406
3	mui/material-ui	@mui/material	2	1	0	UI	2014-08-18T19:11:54Z	78499
4	storybookjs/storybook	@storybook/core	2	1	0	Utility	2016-03-18T04:23:44Z	71018
5	atom/atom		1	0	2	Application	2012-01-20T18:18:21Z	57548
6	expressjs/express	express	1	0	1	Framework	2009-06-26T18:56:01Z	57033
7	chartjs/Chart.js	chart.js	2	0	0	UI	2013-03-17T23:56:36Z	56951
8	lodash/lodash	lodash	2	0	1	Library	2012-04-07T04:11:46Z	53221
9	jashkenas/underscore	underscore	1	1	0	Library	2009-10-25T18:31:06Z	26424
10	ElemeFE/element	element-ui	2	1	0	UI	2016-09-03T06:19:26Z	52095
11	elemefe/element-react	element-react	2	0	0	UI	2016-10-18T06:56:20Z	2718
12	element-plus/element-plus	element-plus	2	1	0	UI	2020-07-21T06:51:19Z	15561
13	facebook/react	react	2	0	2	Library	2013-05-24T16:15:54Z	188137
14	chalk/chalk	chalk	1	1	0	UI	2013-08-03T00:20:12Z	18514
15	reduxjs/redux	redux	1	1	0	Library	2015-05-29T23:53:15Z	58052
16	vitejs/vite	vite	2	1	0	Utility	2020-04-21T05:03:57Z	41817
17	meteor/meteor	meteor	2	0	2	Framework	2012-01-19T01:58:17Z	42888
18	nestjs/nest	@nestjs/core	2	1	0	Framework	2017-02-04T20:12:52Z	47005
19	mermaid-js/mermaid		2	1	0	Application	2014-11-01T23:52:32Z	47280
20	serverless/serverless	serverless	1	0	2	Framework	2015-04-21T03:48:40Z	42759
21	juliangarnier/anime	animejs	2	0	0	UI	2016-03-13T21:37:45Z	42291
22	microsoft/vscode		1	0	2	Open-Core	2015-09-03T20:23:38Z	131809
23	vercel/next.js	next	1	0	2	Framework	2016-10-05T23:32:51Z	86860
24	coder/code-server		1	0	2	Application	2019-02-27T16:50:41Z	53554
25	apache/echarts	echarts	2	0	1	UI	2013-04-03T03:18:59Z	51078
26	sequelize/sequelize	sequelize	1	1	0	ORM	2010-07-22T07:11:11Z	26145
27	wwayne/react-tooltip	react-tooltip	2	0	0	UI	2015-04-07T13:15:04Z	2752
28	d3/d3	d3	2	0	0	UI	2010-09-27T17:22:42Z	101304
29	recharts/recharts	recharts	2	1	0	UI	2015-08-07T06:50:27Z	18239
30	plotly/plotly.js	plotly.js	2	1	2	UI	2015-11-05T23:27:17Z	14668
31	typeorm/typeorm	typeorm	1	1	0	ORM	2016-02-29T07:41:14Z	28251
32	moment/moment	moment	1	0	1	Library	2011-03-01T02:46:06Z	46534
33	fastify/fastify	fastify	1	1	1	Framework	2016-09-28T19:10:14Z	23098
34	Unitech/pm2	pm2	1	0	2	Utility	2013-05-21T03:25:25Z	37005
35	moment/luxon	luxon	1	0	1	Library	2015-11-30T12:48:48Z	12466

Teil 1, Part 2

	code_of_conduct	contributing	license	commits	contributors	last12MonthsCommits	usedBy	downloads	dependants
0	1	1	mit	1199	367	151	6333706.0	30140843	77979
1	1	1	mit	40103	1625	1685	0.0	628367	2359
2	1	1	mit	24510	1547	2500	1913102.0	3163072	12467
3	1	1	mit	19695	2513	2596	867211.0	1271064	1478
4	1	1	mit	39819	1494	0	3438.0	3973351	108
5	1	1	mit	38496	489	151	0.0	0	68
6	1	1	mit	5734	293	110	0.0	25466973	61798
7	0	1	mit	4190	411	266	519078.0	2088541	2335
8	1	1	mit	8005	310	0	15508835.0	44787010	155392
9	1	1	mit	2797	280	38	1491035.0	9144831	22303
10	0	1	mit	4532	569	83	268187.0	311656	8664
11	0	1	mit	1236	48	0	2233.0	2560	33
12	1	1	mit	3462	313	2352	19233.0	81650	744
13	1	1	mit	14979	1557	795	10052259.0	15117154	86909
14	1	1	mit	332	53	15	15166164.0	175939084	0
15	1	1	mit	3530	907	186	2183285.0	7698164	16113
16	1	1	mit	4113	558	1491	212749.0	954460	683
17	1	1	mit	32320	660	1288	64.0	2539	0
18	1	1	mit	10642	306	1392	153201.0	1411654	3044
19	0	1	mit	4766	320	819	9396.0	0	68
20	1	1	mit	15396	959	1262	23412.0	946080	337
21	0	0	mit	736	52	0	31513.0	133994	565
22	1	1	mit	96652	1637	12870	4.0	0	68
23	1	1	mit	11233	2134	2825		2663012	2156
24	1	1	mit	3291	174	546	11.0	0	68
25	1	1	apache-2.0	8665	163	663	178279.0	486021	3635
26	1	1	mit	9355	1011	540	455059.0	1330691	5086
27	0	1	mit	702	104	18	40853.0	1160650	1286
28	0	0	isc	4333	123	35	248041.0	1760874	4483
29	0	1	mit	1910	209	56	78304.0	1000211	958
30	1	1	mit	23973	178	1237	10696.0	144086	171
31	0	1	mit	5002	824	409	158552.0	1076460	2749
32	0	1	mit	3969	592	8	3119830.0	19671551	57545
33	1	1	mit	3220	492	383		519217	1281
34	0	1	agpl-3.0	4970	266	54	62535.0	1383283	1231
35	0	1	mit	1113	184	127	121886.0	3331104	2957

Teil 2, Part 1

	GitHub_Project	NPM_package	Documentation	Sponsors	BackedBy	Category	created_at	stars
36	date-fns/date-fns	date-fns	1	1	0	Library	2014-10-06T10:24:22Z	28756
37	iamkun/dayjs	dayjs	1	1	0	Library	2018-04-10T09:26:44Z	38954
38	vector-im/element-web		2	1	0	Application	2015-07-22T05:32:15Z	8326
39	hexojs/hexo	hexo-cli	1	1	0	Framework	2012-09-23T15:17:08Z	34795
40	Leaflet/Leaflet	leaflet	2	0	0	Library	2010-09-22T16:57:44Z	34581
41	facebook/jest	jest	1	1	2	Test-Framework	2013-12-10T00:18:04Z	39005
42	vercel/hyper		0	1	2	Application	2016-07-01T06:01:21Z	38526
43	cypress-io/cypress	cypress	2	1	2	Test-Framework	2015-03-04T00:46:28Z	38467
44	microsoft/playwright	playwright	1	0	2	Test-Framework	2019-11-15T18:32:42Z	37880
45	styled-components/styled-components	styled-components	2	1	0	UI	2016-08-16T06:41:32Z	36580
46	gatsbyjs/gatsby	gatsby-cli	1	1	2	Framework	2015-05-21T22:43:05Z	52880
47	mochajs/mocha	mocha	1	1	1	Test-Framework	2011-03-07T18:44:25Z	21371
48	preactjs/preact	preact	2	1	0	Library	2015-09-11T02:40:18Z	31668
49	webpack/webpack	webpack	1	1	1	Utility	2012-03-10T10:08:14Z	61066
50	babel/babel	@babel/core	2	1	0	Utility	2014-09-28T13:38:23Z	40861
51	naptha/tesseract.js	tesseract.js	2	1	0	Library	2015-06-24T02:49:52Z	26871
52	niklasvh/html2canvas	html2canvas	2	0	0	Library	2011-07-16T01:05:58Z	25970
53	chakra-ui/chakra-ui	@chakra-ui/react	1	1	0	UI	2019-08-17T14:27:54Z	25993
54	react-hook-form/react-hook-form	react-hook-form	2	1	0	UI	2019-03-05T23:47:10Z	28262
55	caolan/async	async	1	0	0	Utility	2010-06-01T21:01:30Z	27557
56	vuejs/vuex	vuex	2	1	0	Library	2015-07-16T04:21:26Z	27554
57	ReactiveX/rxjs	rxjs	1	0	0	Library	2015-03-15T06:17:10Z	27036
58	webtorrent/webtorrent	webtorrent	1	1	0	Library	2013-10-15T08:16:40Z	26356
59	kenwheeler/slick	slick-carousel	2	0	0	UI	2014-03-24T02:10:05Z	27485
60	markedjs/marked	marked	1	0	0	Library	2011-07-24T13:15:51Z	27545
61	nocodb/nocodb		1	1	0	Application	2017-10-29T18:51:48Z	27548
62	postcss/postcss	postcss	1	1	0	Utility	2013-09-24T23:06:48Z	26255
63	typicode/husky	husky	1	1	0	Utility	2014-06-23T12:14:21Z	26511
64	remy/nodemon	nodemon	1	1	0	Utility	2010-10-03T12:50:52Z	23968
65	hammerjs/hammer.js	hammerjs	2	0	0	UI	2012-03-02T12:58:28Z	23021
66	balena-io/etcher		1	0	2	Application	2015-10-27T16:53:23Z	22949
67	NativeScript/NativeScript	nativescript	2	1	0	Framework	2015-03-01T09:47:08Z	21226
68	appwrite/appwrite		1	1	2	Utility	2019-04-08T16:36:25Z	22074
69	chenglou/react-motion	react-motion	1	0	0	UI	2015-06-11T07:38:23Z	20860
70	sghall/react-move	react-move	1	0	0	UI	2017-03-20T15:38:13Z	6439
71	pmndrs/react-spring	react-spring	2	1	0	UI	2018-03-07T15:39:32Z	23117

Teil 2, Part 2

	code_of_conduct	contributing	license	commits	contributors	last12MonthsCommits	usedBy	downloads	dependants
36	0	1	mit	1874	368	186	1350202.0	14020395	10890
37	0	1	mit	1403	290	63	894211.0	10179207	7046
38	0	1	apache-2.0	11667	486	798	0.0	0	68
39	1	1	mit	3529	162	57	101393.0	21007	9
40	1	1	bsd-2-clause	7276	731	372	132819.0	652454	1742
41	1	1	mit	6213	1356	740	5362219.0	16700955	10484
42	0	1	mit	3155	271	616	0.0	0	68
43	1	1	mit	16641	347	896	438142.0	4032758	379
44	1	1	apache-2.0	7796	236	3038	11432.0	638430	279
45	1	1	mit	3310	295	55	1084349.0	4173510	17149
46	1	1	mit	20200	3862	1742	422343.0	434726	128
47	1	1	mit	3582	477	99	1568144.0	6510061	9138
48	1	1	mit	5032	287	107	105234.0	1407929	1363
49	1	1	mit	15107	722	850	9929393.0	23133348	25327
50	1	1	mit	15198	1021	592	152875.0	40577342	18356
51	0	0	apache-2.0	627	56	2	6181.0	35650	124
52	0	1	mit	1066	50	85	64361.0	956310	1497
53	1	1	mit	8092	511	1108	51244.0	289480	594
54	1	1	mit	3188	214	556	162663.0	1973757	1530
55	0	0	mit	1864	231	68	11529402.0	47338681	32002
56	1	1	mit	1201	312	29	957344.0	1864332	10794
57	1	1	apache-2.0	5010	504	206	6986873.0	37651544	27449
58	0	1	mit	3133	164	201	2878.0	28926	131
59	0	1	mit	921	147	0	0.0	693480	974
60	1	1	mit	2730	144	277	764807.0	5654308	6672
61	1	1	agpl-3.0	3008	90	2224	21.0	0	68
62	0	1	mit	3733	362	240	9864955.0	66378258	8624
63	0	1	mit	838	97	63	723459.0	7498407	2314
64	1	1	mit	1231	150	59	2806696.0	4848158	3645
65	0	1	mit	1199	87	0	361734.0	1027475	1968
66	0	1	apache-2.0	2985	71	68	0.0	0	68
67	1	1	mit	6906	222	308	4389.0	7769	1
68	1	1	bsd-3-clause	10934	247	3805	0.0	0	68
69	0	0	mit	809	72	0	47570.0	705665	1137
70	0	1	mit	524	11	14	4907.0	106796	89
71	0	1	mit	3255	141	137	84565.0	789818	1491

Teil 3, Part 1

	GitHub_Project	NPM_package	Documentation	Sponsors	BackedBy	Category	created_at	stars
72	vercel/pkg	pkg	1	0	2	Utility	2016-08-08T19:41:59Z	21114
73	reactjs/react-transition-group	react-transition-group	1	1	2	UI	2016-11-20T19:29:24Z	9186
74	vercel/serve	serve	1	0	2	Utility	2016-04-27T03:58:25Z	7751
75	tailwindlabs/tailwindcss	tailwindcss	2	0	2	UI	2017-10-06T14:59:14Z	56935
76	mholt/PapaParse	papaparse	2	0	0	Library	2013-10-07T20:33:21Z	10521
77	cheeriojs/cheerio	cheerio	1	1	0	Utility	2011-10-09T04:23:20Z	25070
78	thlorenz/parse-link-header	parse-link-header	1	1	0	Library	2013-06-07T11:01:54Z	285
79	node-fetch/node-fetch	node-fetch	1	1	0	Library	2015-01-26T07:29:26Z	7617
80	motdotla/dotenv	dotenv	1	0	0	Library	2013-07-05T18:25:05Z	15393
81	npkgz/cli-progress	cli-progress	1	0	0	Utility	2015-12-13T13:52:36Z	679
82	fent/node-ytdl-core	ytdl-core	1	0	0	Library	2014-06-29T22:06:35Z	3210
83	mozilla/pdf.js		2	0	1	Application	2011-04-26T06:32:03Z	38613
84	videojs/video.js	video.js	2	0	2	UI	2010-05-14T18:45:10Z	33312
85	alvarotrigo/fullPage.js	fullpage.js	2	1	0	UI	2013-09-20T11:58:29Z	33575
86	supabase/supabase		1	1	2	Application	2019-10-12T05:56:49Z	33404
87	google/zx	zx	1	0	2	Utility	2021-05-05T05:50:01Z	31183
88	gorhill/uBlock		1	0	0	Application	2015-04-01T17:51:11Z	30414
89	prisma/prisma	prisma	1	1	0	ORM	2019-06-20T13:33:47Z	22831
90	ovity/octotree		1	0	2	Open-Core	2014-05-09T18:15:20Z	21982
91	usablica/intro.js	intro.js	2	0	2	UI	2013-03-10T15:12:45Z	21209
92	jsdelivr/data.jsdelivr.com		1	1	1	API	2017-07-12T19:14:58Z	145
93	grid-js/gridjs	gridjs	2	1	0	UI	2020-04-15T17:41:40Z	3400
94	puppeteer/puppeteer	puppeteer	1	0	2	Utility	2017-05-09T22:16:13Z	77974
95	lovell/sharp	sharp	1	1	0	Utility	2013-08-19T20:24:24Z	22279
96	redux-saga/redux-saga	redux-saga	1	1	0	Library	2015-11-29T16:58:12Z	22191
97	GoogleChromeLabs/squoosh		1	0	2	Application	2018-03-07T22:42:03Z	16399
98	jasmine/jasmine	jasmine	1	0	0	Test-Framework	2008-12-02T23:46:37Z	15347
99	jgraph/drawio		0	0	2	Application	2016-09-06T12:59:15Z	29307
100	GoogleChrome/lighthouse	lighthouse	1	0	2	Utility	2016-03-08T01:03:11Z	24599
101	uxsolutions/bootstrap-datepicker		2	0	0	UI	2012-03-17T01:11:40Z	12512
102	RobinLinus/snapdrop		2	1	0	Application	2015-12-18T15:44:18Z	12156
103	spicetify/spicetify-cli		1	1	0	Application	2018-12-01T19:55:18Z	11326
104	shelljs/shelljs	shelljs	1	0	0	Utility	2012-03-02T17:04:47Z	13221
105	shelljs/shx	shx	1	0	0	Utility	2016-02-14T19:53:03Z	1354
106	ether/etherpad-lite		1	1	0	Application	2011-03-26T13:09:02Z	12756
107	pa11y/pa11y	pa11y	1	0	0	Test-Framework	2013-03-05T09:51:11Z	3339

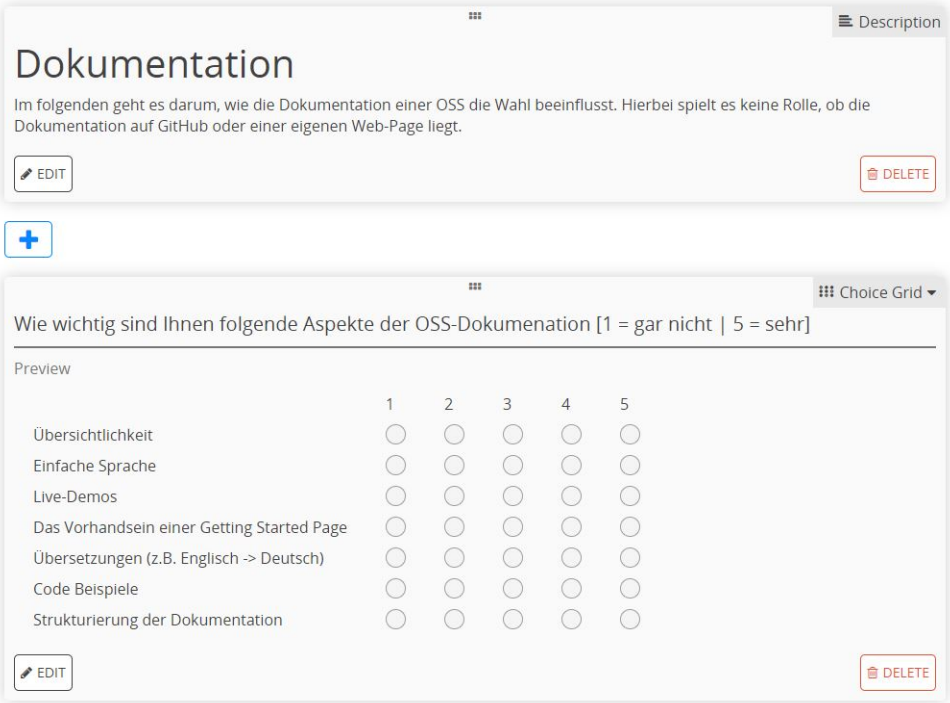
Teil 3, Part 2

	code_of_conduct	contributing	license	commits	contributors	last12MonthsCommits	usedBy	downloads	dependants
72	0	0	mit	1108	59	72	8902.0	131187	190
73	0	0	bsd-3-clause	386	78	42	1208888.0	9709535	7012
74	0	1	mit	729	84	14	419347.0	837011	587
75	1	1	mit	4627	229	719	1406008.0	2941526	1749
76	0	1	mit	478	121	15	39000.0	1255055	1133
77	0	1	mit	1959	126	556	887123.0	7160234	13672
78	0	0	mit	25	6	2	18931.0	726580	295
79	1	1	mit	622	107	62	4707650.0	33082039	25481
80	0	1	bsd-2-clause	470	57	129	7251584.0	25098833	27964
81	0	1	mit	134	18	13	49079.0	1977763	1478
82	0	0	mit	1045	80	18	0.0	61275	583
83	1	1	apache-2.0	15693	359	867	165.0	0	68
84	1	1	apache-2.0	3821	417	140	27540.0	417597	951
85	0	1	gpl-3.0	1990	138	67	7130.0	14487	23
86	1	1	apache-2.0	7035	385	3317	0.0	0	68
87	1	1	apache-2.0	280	33	180	1451.0	157594	148
88	0	1	gpl-3.0	9769	98	979	0.0	0	68
89	1	1	apache-2.0	7959	171	1380	51557.0	451782	132
90	0	0	agpl-3.0	718	66	8	2.0	0	68
91	1	1	agpl-3.0	1188	101	183	3444.0	72180	79
92	0	0	osl-3.0	462	5	82	0.0	0	68
93	1	0	mit	1252	17	177	579.0	12913	12
94	0	1	apache-2.0	2625	418	452	204655.0	3197765	4980
95	0	1	apache-2.0	1627	162	197	333865.0	1789940	2801
96	0	1	mit	1906	325	11	246140.0	1080559	2900
97	0	1	apache-2.0	1425	61	137	59.0	0	68
98	1	1	mit	2532	218	193	2254668.0	1860643	978
99	1	0	apache-2.0	1774	31	184	0.0	0	68
100	1	1	apache-2.0	5238	310	596	10089.0	601188	287
101	1	1	apache-2.0	1674	343	1	82458.0	0	68
102	0	0	gpl-3.0	370	35	11	0.0	0	68
103	0	0	lgpl-2.1	713	56	298	0.0	0	68
104	0	1	bsd-3-clause	814	84	19	1936008.0	7285272	14691
105	0	1	mit	143	11	8	71859.0	426146	258
106	0	1	apache-2.0	7986	278	723	39.0	0	68
107	0	1	lgpl-3.0	561	59	38	4228.0	139010	57

A.2 Umfrage und Ergebnisse

A.2.1 Dokumentation

Für die Umfragen wurde das Open Source Tool *cryptpad.fr* verwendet Beispiel einer Frage aus dem Fragebogen:



Wie wichtig sind Ihnen folgende Aspekte der OSS-Dokumentation [1=garnicht | 5 = sehr]

	1	**2**	**3**	**4**	**5**
Übersichtlichkeit	0	6	26	124	149
Einfache Sprache	29	59	100	87	29
Live-Demos	30	89	89	66	29
Das Vorhandensein einer Getting Started Page	204	54	34	6	6
Übersetzungen (z.B. Englisch ->Deutsch)	5	22	56	109	109
Strukturierung der Dokumentation	3	4	28	11	157

Die folgenden Punkte würden mich von der Nutzung eines OSS-Projekts abhalten oder mich möglicherweise dazu veranlassen, nach Alternativen zu suchen. [1 = Stimme ich garnicht zu | 5 = Stimme ich voll und ganz zu]

	1	2	3	4	5
Keine Code Beispiele	5	26	45	91	137
Schlechte Struktur	2	25	68	110	100
Komplexe Doku	12	55	109	85	43
Fehlende Ubersetzung	227	51	18	4	4
Fehlende Getting Started Page	22	77	91	76	37

Hat sie eine schlechte Dokumentation jemals dazu gebracht, ein alternatives Projekt zu wählen?

249 haben für *"Ja"* gestimmt, und 58 für *"Nein"*.

A.2.2 Beliebtheit

Im folgenden Teil geht es um die Aspekte der Beliebtheit und wie diese Einfluss bei der Wahl eines Projektes haben.

Wie sehr achten Sie bei der Auswahl von OSS auf... [1 = garnicht | 5 = sehr]

Die Frage bezüglich Trends bezeiht sich auf Trends wie sie auf z.B. StackOverflow, npm trends Google Trends oder ähnlichen Plattformen zu finden sind.

	1	2	3	4	5
Anzahl der Downloads	33	49	67	118	40
GitHub/GitLab Sterne	46	60	96	83	22
Anzahl Contributor	54	82	100	51	20
Trends	105	84	69	40	9
Anzahl Sponsoren	192	103	57	17	1
Anzahl von StackOverflow Fragen/Antworten	89	50	83	64	21

Hat die geringe Popularität eines Projekts Sie jemals davon abgehalten, es zu nutzen?

140 haben für *"Ja"* gestimmt, und 167 für *"Nein"*.

A.2.3 Sponsoren

Im folgenden geht es, um Sponsoren und dessen Einfluss bei der Wahl eines OSS-Projekts. Da einige Open Source Projekte, wie Android von Google, von Unternehmen entwickelt werden sind diese nicht von Sponsoren nicht abhängig und spielen für die folgenden Fragen daher keine Rolle. Die Fragen beziehen sich daher auf Projekte, die von der Community entwickelt werden und auf Sponsoren angewiesen sind. Beispielsweise VueJS oder PostgreSQL.

Bewerten Sie folgende Aussagen [1 = stimme NICHT zu | 5 = stimme voll und ganz zu]

	1	2	3	4	5
Zukunftssicherer	26	34	11	110	26
Projekt mit Sponsor wird bevorzugt	62	82	117	40	6
Gesponsert ist besser	45	69	135	47	11

A.2.4 Devlopment

Welchen Einfluss hat der Entwicklungsprozess bei der Auswahl eines Projektes.

Wie sehr achten Sie auf die Folgenden Punkte, wenn Sie ein Projekt wählen? [1 = gar nicht | 5 = sehr]

	1	2	3	4	5
Anzahl der Maintainer	24	55	71	111	46
Ticket/Issue Verhältnis Open/Closed	44	62	97	89	15
Regelmäßigkeit der Commits	24	47	95	102	39
Aktualität des letzten commits	11	21	49	121	105
Antwortzeit der Entwicklera auf Tickets/Issues	33	74	113	78	9

Priorisierung der Kriterien

In der letzten Frage sollten die Teilnehmer Kriterien sortieren nach Priorität sortieren. Die Aussage war: *Sortieren Sie nach den für Sie wichtigsten Kriterien bei der Auswahl von OSS.* Das Tool hat je nach Platzierung die Kriterien bepunktet. Für Platz 1 gab es 5 Punkte, für Platz 2 gab es Platz 4 gab es 4 Punkte usw.

Folgende Auflistung hat sich entsprechend ergeben:

- **[1329]** Punkte für Gute Dokumentation
- **[1165]** Punkte für Beliebtheit (Anzahl der Downloads, GitHub Stars etc.)
- **[794]** Punkte für Anzahl von offenen Issues/Tickets
- **[624]** Punkte für Trends (z.B. auf StackOverflow oder GoogleTrends etc)
- **[559]** Punkte für Projekt hat Sponsoren

A.2.5 Freitextfeld: Gute Dokumentation

Anmerkung: Aufgrund des Dateiformats welches das Umfragetool ausgegeben hat, könnte es sein, dass Aussagen der gleichen Person über zwei Stichpunkte verteilt ist, bzw das mehrere Teilnehmer in einem Stichpunkt zusammengefasst sind. Da die Daten einzeln manuell kategorisiert worden sind spielt dies jedoch keine weitere Rolle.

- Aktualität
- Offensichtliche Verlinkungen der Lizenzen
- "Ladezeit nicht zu langsam falls im web; angenehm lesbare Schrift und Layout"Wozu den Dokumentation? Die meisten Antworten findet man im Quellcode direkt! Ein klarer Vorteil gegenüber proprietärer Software, welche in den Projekten idR. immer dekompiliert werden muss, um Fehler in dieser zu finden. Changelog/Release History, Contribution Doku, Gut organisiertes Issue Tracking
- Vollständigkeit und Aktualität
- FAQ, potentielle Anwendungsfälle, Roadmap für weitere future changes, Wortwahl wie man sie in der Praxis wiederfindet (erleichter Suchen in der Doku)
- Gute Navigation und Verlinkung
- Gute Verfügbarkeit (z. B. lokal als Man-page)
- Codebeispiele die relevant sind, nicht nur realitätsferne Beispielprojekte.
- Ist aktuell und nicht veraltet
- YouTube Videos zum Tool
- Aktualität
- Architekturschaubild, Aktivitäten im Projekt (Häufigkeit, letzte Änderungen)
- Installationsanleitung für Windows und mehr Linux-Distributionen als nur Ubuntu.
- Aktualität der Dokumentation. Es muss klar sein, für welche Version sie gilt. Inhaltliche Vollständigkeit und länge sind wichtig.
- Referenzierung aus StackOverflow und anderen Helpforen
- Vollständige API Dokumentation mit Code Beispielen; Transparenz bezüglich Abhängigkeiten zu anderen OSS Modulen / Bibliotheken
- Formatierung
- Formatierung
- Muss laufend aktualisiert werden
- Vollständigkeit - Nicht nur einen Teil der Klassen/Methoden dokumentieren.
- Use Case Beschreibung
- Abstufung der Dokumentation von Konzeptionell zu Konkret
- Aktualität, Korrektheit
- Sie wird auch gepfelgt.
- Vollständigkeit

- Vollständige und aktuelle Dokumentation
- Vorbedingungen werden genannt, evtl Workarounds werden gut beschrieben
- Versionierung, Aktualität, Korrektheit
- Aktualität -> sollte permanent auf aktuellem Stand gehalten werden
- Ërklärung komplexer Beispiele; Gute und durchgreifende Erklärung von Grundbausteinen"
- Kurz, auf den Punkt, suchbar. Optimalbeispiel: https://poi.apache.org/components/spreadsheet/quick-guide.html
- Aktualität
- Vollständigkeit.
- Aktualität, Version auswählbar bzw. sichtbar für welche Version und Möglichkeit zur richtigen Doku für die verwendete Version zu wechseln
- Zielgruppen-Orientierung (Admin, Dev, User...)
- Vollständig, aktuell.
- Sollte gut darlegen, was die OSS kann und wie sie sich von der Konkurrenz abhebt
- "Tutorials, die über das übliche Getting Started hinaus gehen."
- Aktualität (=passt der aktuelle Stand der Dokumentation zu den Funktionen der Software)
- Vollständigkeit. Nichts ist schlimmer, als wenn manche bereiche sehr gut dokumentiert sind aber DIE eine Methode die man braucht gar nicht
- detailierte beispiele als reference siehe die dokumentation von Django
- Versionierung / Bezug zu Releases & Codebase (z.B. Github etc.)
- Gute Suche/ Verschlagwortung.
- Vollständigkeit
- Sie sollte existieren. Das ist nicht immer gegeben.
- Syntaxhighlighting, Querverweise, Volltext-Suchfunktion
- Gepflegte und aussagekräftige Release Notes. Es sollte leicht erkennbar sein, welche Features ab welcher Version zur Verfügung stehen und Breaking Changes dokumentiert sein.
- Eine hohe Abdeckung des Funktionsumfangs
- API / Schnittstellen Dokumentation
- nicht zu lang, man kann gleich ausprobieren. Keine Überraschungen wie zum Beispiel bei Log4J
- Die Software Live, bspw. mit Codepen, ausprobieren zu können und Minimalstbeispiele für den Einsatz zu evaluieren.
- Aktualität (oftmals sind Beispiele noch für alte Versionen), Vollständigkeit, Vermittlung von zugrundeliegenden Prinzipien
- Kkleines Demoprojekt
- Versionierung und Changelog
- gute Pflege * Gute API Dokumentation * Konsistentes Wording, klare Terminologie,... * Bei Tools: Cheatsheets sind echt hilfreich. Bsp: git cheat sheet * Bücher: zb Cookbooks die Standardanwendung für OSS ausführlich erklären.
- Ëinstufung / Klassifikation - z.B. 5 Minuten Überblick, 60 Minuten Tutorial usw."
- - Umgang mit Issues und Feature-Requests - How to participate
- Vorhanden sein einer Suche, Verwendung von Standardisierung für gängige Seiten, Versioniert, aktuell, Trennung zwischen User Doku und Developer Doku.
- GRundsätzlich verständlich und vollständig
- Vollständigkeit und Aktualität
- Vollständigkeit (Beschreibung aller Funktionen/Optionen)
- Immer auf dem aktuellen Stand
- Doku alter Versionen verfügbar
- Eine gute und vollständige JavaDoc
- Vollständigkeit für sowohl Nutzer*innen als auch Entwickler*innen
- Möglicherweise Nutzer Feedback
- Dokumentation auf aktuellem Stand, möglichst vollständige Dokumentation aller Features, direkter Link von der Dokumentation eines Moduls zu dessen Code
- Real-world Beispiele, die über die Basisbeispiele hinausgehen.
- Die Existenz einer solchen (= großer Minuspunkt für ein OSS-Projekt, falls keine vorhanden ist). Mehr Inhalt als eine bloße Auflistung von Commands oder ähnlichen, d.h., auch höhere Konzepte, die der Software zu Grunde liegen werden erklärt.
- - Migrationsguides zwischen Versionen. - Doku zu breaking changes zwischen Versionen. - Konsistente Code-Beispiele - Gute Verlinkung zwischen zusammenhängenden Kapiteln/Themen - sichtbares Aktualisierungsdatum - möglichst zeitnahe Updates, idealerweise immer kombiniert mit Codeanpassungen/Feature-Releases
- technische Erläuterungen
- Wenn es sich um eine Bibliothek handelt, ist die Wichtigkeit der Doku davon abhängig wie intuitiv die Nutzung und der Code sind. Wenn das Programm stabil läuft und die wichtigen externen Methoden in einer Klasse stehen, reicht ein Getting Started.
- Getting started und Minimalbeispiele statt seitenlange Texte, die ja sowieso keiner liest.
- Eindeutigkeit, vollständige Abdeckung des Projektes
- Verlinkung zwischen Inhalten, Definitionen zu Fachbegriffen
- - Onboarding - Contribution - Roadmap
- Verlinkung zwischen Inhalten, Definitionen zu Fachbegriffen
- Gute Navigierbarkeit, d.h. man findet schnell von Punkt A zu B, wenn diese zusammenhängen. Gegenbeispiel wäre eine Reihe von Wiki Pages ohne Links zueinander.
- Aktualität.
- Sichtbarkeit auf einen Blick, welche Bedingungen gelten und wie und wann und von wem die OSS eingesetzt werden darf.

- Einsteigerfreundlichkeit, komplexer werdende Inhaltsstruktur d.h. am Begin der Dokumentation werden die Dinge eher einfach gehalten und werden zum Ende zunehmend komplexer
- WikiSeite/Doku
- Einfach einbindbar in Gebrauchskontext
- Einfach findbar (z.B. über übliche Repositories)
- Funktionaltität/Gebrauchsspektrum ist einfach erkennbar/überschaubar
- online Verfügbarkeit inklusive guten Suchmöglichkeiten und Querverlinkungen
- Changelogs / Release Notes
- Roadmaps
- Ausblicke, wie lange die Software auf jeden Fall noch gepflegt werden wird
- Ausblicke, wann das nächste Update geplant ist
- bei guter doku sind kein Codebeispiele nötig, obwohl diese das verständis meist verbessern
- Eher Development, aber saubere, atomare Commits bzw. Commit-Historie
- aktuelle Dokumentation
- Vollständigkeit
- Eigentlich nichts was icht oben genannt wird
- Q&A
- Aktualität
- Versionierung, Trennung zwischen Konzept und API, Aktualität
- Ümfang und Vollständigkeit. Aktualität, und dass es nicht zu abstrakter standards jargon ist."

A.2.6 Freitextfeld: Schlechte Dokumentation

Anmerkung: Aufgrund des Dateiformats welches das Umfragetool ausgegeben hat, könnte es sein, dass Aussagen der gleichen Person über zwei Stichpunkte verteilt ist, bzw das mehrere Teilnehmer in einem Stichpunkt zusammengefasst sind. Da die Daten einzeln manuell kategorisiert worden sind spielt dies jedoch keine weitere Rolle.

- Verstreut über viele Unterprojekte. Veraltete Doku wird zuerst gefunden. Beispiel: Spring framework
- + Codebeispiele die nicht funktionsfähig sind oder einen anderen als den dokumentierten Softwarestand verwenden. + Pseudocode ohne Codebeispiele in für den Anwendungszweck gängigen Programmiersprachen + Fehlende Auflistung von Third-Party-Modulen
- - Keine API Dokumentation - Intransparenz bezüglich der Abhängigkeiten zu anderen OSS Modulen / Bibliotheken
- Weitere Aspekte einer schlechten OSS Dokumentation:
- Fehlende Versionierung
- Gar keine Dokumentation ist ein Zeichen von fehlendem Engagement/Einsatz - nicht vertrauenswürdig
- Überhaupt keine Dokumentation vorhanden
- Beschreibung veralteter Methoden, die so nicht mehr unterstützt werden
- Wortwahl die zwar das Gleiche ausdrückt aber so nicht oft verwendet wird z.B. to read from database VS to read saved content from XYZ
- Copy-paste Ausschnitte aus dem Code (z.B. aus Headern)
- keine vorhanden
- Veraltet, unvollständig
- Installationsanleitung für Apple (wenn es etwas für Apple gibt wird meiner Erfahrung nach mehr auf hip und fancy anstatt auf Qualität geachtet)
- Überladende (ausschweifende) Dokumentation sind wenig hilfreich,
- Ungenügende Pflege
- Codebeispiele funktionieren nicht
- Unvollständigkeit, besonders wenn MethodenParameter nicht ausreichend beschrieben werden.
- Abwesenheit von Dokumentation
- Wenn die Doku aus der Sicht geschrieben ist, das eh schon alles klar ist.
- Outdated
- Unverständliche Formulierungen, zusammenhangslose Beispiele, inkonsistenter Codeing Style.
- Unvollständigkeit, schlechte Übersicht, schlechte Suche (wenn Methoden oder Attribute ähnlich heißen sollten diese trotzdem Teil des Suchergebnis sein)
- Kein Überblick, nur Javadoc
- nicht aktuell
- Aufeinander aufbauende Beispiele
- Geschwafel, 50-seitenlange Abhandlung über Grundkonzepte.
- Ünvollständigkeit. ;-)"
- Nicht einheitliche Dokumentation(React)
- Setzt zu viel Vorwissen voraus.
- Unvollständig, veraltet, fehlerhaft
- Wenn man sie nicht finden kann, outdatet ist, Einrichtung nur für ein OS
- s.o.
- fehlerhafte code beispiele
- Keine Änderungschronik, kein Bezug zu Codebase und Release
- Veraltet, falsch oder auf irreführendeweise unvollständig.
- schlechte Code Beispiele, keine Beispiele für schwierige Use Cases
- Dokumentation so veraltet dass man die aktuelle Version damit nicht verwenden kann

- Öffene Fragen zbs in Foren zu einer Software. Je mehr Fragen gestellt werden, könnte ein Indiz für nicht so gute Doku sein ;)"
- schlechtes UX/UI, Schreibfehler
- - Nicht aktuell oder inkorrekt - Bad Practices in Beispielen
- Veraltet, fehlende Funktionen, keine Erklärung des Konzepts
- API / Schnittstellen Dokumentation
- Ünvollständigkeit, unpräzise Erklärungen, zu simple Beispiele (ich bekomme Ausschlag, wenn ich in einer Doku Hello world lese)"
- Verlatete, Fehlerhafte oder lückenhafte Dokumentation bei wesentlichen Aspekten OSS.
- Obskure Bedientoberfläche (GUI) oder Bedienkonzepte die schlecht oder gar nicht erklärt werden in der Doku.
- Dokumentation Version stimmt nicht mit der aktuellen Software überein, bzw. ist nicht klar, auf welche Version der Software sich die Dokumentation bezieht.
- Veraltet oder unvollständig
- Nicht aktuell, fehlerhafte Doku, keine explizite User Doku, nicht einhalten von standartdts,
- Wichtige Aspekte fehlen und/oder sind veraltet
- Widersprüchliche Informationen an unterschiedlichen Stellen der Dokumentation (= kein Vertrauen in Konsistenz), unbenutzbares Interface (Design).
- - Immer outdated
- - Code-Beispiele gehen (nicht) mehr
- unverständlich, da Kontextinformationen fehlen / hergeleitet werden müssn
- Wenn man den Code nicht selbst bearbeiten, sondern nur nutzen möchte ist das Wichtigste an eine Doku die Beschreibung der öffentlichen Methoden und Parameter. Diese sollte zwingend vollständig sein und kann aus Kommentaren im Code generiert werden.
- Wenn die/eine Sprache nicht Englisch ist, scheidet sie aus (zB gibt es in letzter Zeit zunehmend viele Repos mit vielen Sternen in Chinesisch).
- Lücken,
- -Nicht aktuell, keine Unterscheidung zwischen Versionen
- veraltete / nicht aktuell gepflegte Dokumentation, fehlende Informationen über die aktuellen Features aus neuen Releases des Projektes
- Keine Suchfunktion
- Veraltete Dokumentation, die nicht als solche markiert ist
- Veraltete Dokumentation
- zu alt bzw. zu weit entfernt von der Aktualität
- keine, falsch, veraltet
- schwierige Sprache, schlechtes Engisch/Deutsch, eine OSS die nicht englisch oder deutsch ist
- Abwesenheit einer Doku
- Abwesenheit von Überblickseiten
- Abwesenheit einer Installationswalkthrough
- veraltet, fehlerbehaftet
- Alles ist auf einer Website enthalten. Dadurch kann man schlecht navigieren, die Übersichtlichkeit leidet und es ist schwieriger auf jeweilige Passagen zu verlinken
- -
- Eher Development, aber unordentliche, nicht-atomare Commits bzw. chaotische Commit-Historie
- Am schlimmsten sind Dokumentationen die noch einen alten Stand haben und Code nutzen der als Deprecated makiert ist. Dies macht die OSS oft komplett unbrauchbar
- Uneindeutiger Sprachgebrauch
- Fehlendes Menü an der Seite, sodass man vor einer gigantischen Wall of Text steht, nur um dann immer wieder nach oben scrollen zu müssen und da dann den nächsten Punkt auszuwählen.
- Beispiele nicht minimal, Dokumentation nicht aktuell
- zu abstrakt, ohne greifbare Beispiele