MASTER'S THESIS

Attractiveness to new Contributors of Open Source Software projects on Github

Nieuwhof, B.

Award date:
2021

Attractiveness to new Contributors of Open Source Software projects on Github

by

Ben Nieuwhof

In partial fulfillment of the requirements for the degree of

Master of Science
in Software Engineering

At the Open University, faculty of Management, Science and Technology
Master Software Engineering
To be defended publicly on 6 July 2021 at 1:00 PM

Student number:

Course code:	IM9906	
Thesis committee:	mrs. dr. ir. Fenia Aivaloglou (chairman),	Open University
	mr. prof. dr. Marko van Eekelen (supervisor),	Open University

Contents

1. Abstract

The importance of Open Source Software is increasing rapidly. Open source software projects require newcomers for their continuity. However attracting newcomers can be challenging. In several papers, aspects about attractiveness or barriers for newcomers are analyzed.

In this thesis we explore two aspects that can potentially affect the attractiveness of OSS projects for newcomers, namely, (1) the acceptance of their first pull request, and (2) the existence of different kinds of technical and organizational documentation. The research is done on a snapshot of the Github repository, collected by the ghtorrent project. This project made it possible to do quantitative research on a total of 4.442.209 projects as well as to select 66 projects for which their documentation was manually inspected.

We found out that there was no correlation between the acceptance of a first pull request and the willingness to contribute more to a project by a newcomer. This was an unexpected outcome. The existence of helpful documentation to start contributing, such as ‘how to start’ documents and guidelines appear to be very effective.

2. Introduction

The Open Source Software community is continuously growing. Software development teams are collaborating and sharing their code via open source project repositories such as Github.com and Sourceforge.net. In the beginning March 2019 on Github there were over 30.000.000 registered users and more than 115.000.000 projects. Github is built as a repository for the Git versioning system. Microsoft bought Github in 2018. This has led to a situation that anti-Microsoft OSS-developers move their project to Gitlab or Bitbucket, both based on Git too.

Open source software has helped in mitigating several problems with closed source software. For example, in the Netherlands, a chipcard was introduced for public transport based on the Mifare Chip. The proprietary security algorithm by NXP could relatively easily be hacked [22]. While customers assume to have bought a safe payment system for public transport, the closed source proprietary algorithm is only tested by a limited number of testers and thus are safety guarantees restricted.

Another problem with closed software is the lack of transparency. For example, a customer of some system doesn't know what information of his enterprise is shared with others. Especially in SAAS environments, a customer has limited or no knowledge about data leaving his ERP system or CRM system. The current discussion about 5G in Europe and the doubt of giving access to the Chinese manufacturer Huawei to the public tender is an example of this fear.[1] However, it's not only China being a risk. Large companies like Facebook, Google, Apple and Microsoft get access to so much data that they have the power to manipulate processes in society, like elections, or business[2]. At the same time, software has become that complex that it is hardly doable to do exhaustive black box testing for customers on all aspects. For that reason transparency is required, so software suppliers are requested to prove that their software does not do anything, that the user doesn't know. This requirement demonstrates the importance of Open Source Software.

Developers have also been found to benefit from open source software. When young developers start to work in a closed source environment, their contribution is mostly focused on an existing product. Their job is to extend it and to improve it. The development process is predefined and coding standards exist. These are often company standards that are not necessarily common best practices. On the other hand, when young programmers contribute in Open Source Software, they come in a situation where there is a need for coding standards and process standards because collaboration would be problematic if they are omitted. Ye and Kishida researched the motivation of Open Source Software Developers [28] and found out that contributors are attracted because the product solves a problem for them, but they also signalize that improving developers skills and

[1] Volkskrant 5 december 2019, Kabinet besluit Huawei te weren uit kern 5G-netwerk (Laurens Verhagen en Niels Waarlo)

[2] New York 4 april 2018, Times Cambridge Analytica and Facebook: The Scandal and the Fallout So Far

acquiring a good reputation in the developers community motivates developers to contribute.

Skill transfer is another benefit of the OSS movement. Kuechler [19] interviewed developers in OSS projects and found out that 80% of the developers in OSS projects joined the project to develop new skills, while 68% joined the projects to become a better programmer. The possibilities that OSS projects offer for developers, could certainly attract new developers. When contributors contribute to more than one project, skills they acquired in one project can be passed to another project. It improves the project quality but participating in other projects also gives a chance to meet other standards and to work with more well-crafted software to gain new skills. Both projects and developers therefor benefit by the mobility that the open source movement facilitates.

2.1 Research motivation and scope

The open source community depends on programmers willing to maintain and expand software. To ensure the future of OSS projects, it is important to attract new, preferably young developers. The motivation for this project comes from personal experience. In daily contact with young students in Software Engineering, I noticed that those young people do not tend to contribute to open source software but prefer a job in closed source environments. Almost every graduate starts with a job in a closed source environment. Apparently open source software projects are not very attractive to them.

Prior research has highlighted the problem of this unattractiveness of OSS-projects to potential new contributors. Mereille et al. [20] mentioned structural complexity as a barrier for onboarding. Stol et al. [26] also mentioned architectural complexity and lack of domain expertise.

The main purpose of this research is to find out which barriers prevent new contributors from joining an OSS project and what facilities attract new contributors. Barriers have a negative impact on attractiveness to new contributors in open source projects. The outcome may help to introduce the right projects to students to make them curious and willing to contribute.

One of the potential barriers that we want to explore is the handling of newcomers' pull requests. Van Krogh et al. [10] found that pull requests from newcomers often are rejected because of existing duplicates or requirement changes. The inner circle of project developers communicates project specific developments via private channels, so the newcomer is not aware of the fact that issues already are picked up or obsolete. For the newcomer who is eager to co-operate this is frustrating and it forms a barrier to contribute more. For that reason, this research a correlation between attractiveness and acceptance of first pull requests by new contributors will be researched.

Čubraniç et al. [8] mentioned the absence of documentation, unclear documentation and outdated documentation as a barrier for newcomers. Ho-Quang [14] however, did not find a correlation between the presence of UML and attractiveness. For that reason. research will be done to find the correlation between different kinds of documentation and attractiveness to newcomers.

This research is done on a snapshot of one of the biggest existing OSS-repositories and is related to several other research what will be described in the next chapter.

3. Background and related work

The prior chapter emphasized the need of OSS-developers and declared the motivation for this research. In this chapter the context and related inspiring research is described.

3.1 Context of research

Prior research on the implicit attractiveness of OSS project is mostly based on statistical research where qualitative research finding barriers on this subject is mostly based on a limited number of projects. Finding reasons why developers don't contribute is often researched using interviews with developers or studies on exploring the communication channels such as email.

For this research the research context is a snapshot of the Github repository taken on the 1st of March 2019. This snapshot contained more than 30.000.000 users, over 115.000.000 projects over 30.000.000.000 commits and more than 47.000.000 pull requests. GHTorrent is a Github mirror containing Github metadata and has already been used in several studies, for example by Gousios [11] and Gousios and Diomidis [12].

The work that is most relevant to the topic of this thesis is a study of Igor Steinmacher et al. [16] about attracting, onboarding and retaining newcomer developers. He refers to aspects, what makes project basically attractive, to former research in 4 papers. These 4 papers are discussed in paragraph 3.22. He also performed a systematic literature review on hindering factors that prevent software developers to contribute. This review was based on 19 other researches and is discussed in paragraph 3.3. At the and he proposes a developers joining model. This is discussed in paragraph 3.44.

3.2 Attractiveness of OSS projects to new contributors

There are several studies on attractiveness. In Table 1, OSS project *attractiveness definitions* on page 8, an overview is given on the outcome of these studies.

Carlos Santos et al. [5] defined attraction as popularity amongst users to use the software and sponsors to spent money in further development. He also researched the correlation between attractiveness to sponsors and users and activeness in the contributing community. The influence on attractiveness in this research is correlated to the type of license, the type of user, the application domain and the state of development of the project. He also computed the correlation between the aspects influencing attractiveness and activeness, effectiveness, likelihood of task completion and time for task completion. Because the variety of relationships he used Structural Equation Modelling to compute these correlations. In this research Santos concluded that attractiveness certainly influences the activeness of developers in the project. He also found a confirmation that attractiveness is influenced by the type of license, the type of user, the application domain and the state of development of a project.

Mereilles et al. [20] refers to Carlos Santos et al. but defined attractiveness as *the capacity of bringing users and developers to a project*. He adds Structural complexity and Lines of Code as aspects influencing attractiveness. They researched the influence of

complexity and #LOC (number of lines of code) on attractiveness. They measured attractiveness as the number of downloads for the project and the number of members. They did a statistical research on sourceforge.net. For this, they used an analyzing tool to analyze source code metrics. They started with 11.433 projects but eventually 6.773 projects where analyzed. They concluded that the #LOC has a significant effect on the number of project users and developers because a large #LOC indicates a lot of features. The results of their statistical research indicated that Structural Complexity has a negative impact on attractiveness to developers. In spite of their expectation, they haven't found a correlation between the number of modules and attractiveness.

Ververs et al.[27] studied the influence of certain events on the attractiveness of the Debian OSS project. He traced the logfiles from 2000 to 2011 of the Debian project and manually checked the Debian website for upcoming events. The log-items were categorized and linked to the kind of upcoming events. The events and log-items were persisted in a SQL database and he searched for correlation between the activity in de developers community and the weeks before and after the event using a linear technique. However, he found in only 10.82% of all measurements a weak, moderate or strong correlation. Most events didn't have any influence. Only Cebit, Debian Day and the introduction of new releases or frozen releases influenced the activity more or less.

Chengalur-Smith et al. [7] researched the correlation between project development base size, project age and niches size and its attractiveness. They defined attractiveness as *the ability of the project to attract and retain developer resources*. The project development base size is the number of active developers in a project, while a niche size in this research is the base of potential contributors in the used program language and the used operating system.

They measured the number of new contributors between 2 periods of a year and the activity of contributors in 2 years to measure attractiveness. In this situation there are more influence factors possibly influencing each other. Structural Equation Modelling is used to compute correlation. His research was based on ~2000 projects in sourceforge.net. The conclusion of this research was that development both size, niche size and project age indeed influences a project capability to attract new developers in the future.

3.3 Barriers for newcomers to OSS projects

Igor Steinmacher et al. [25] published a systematic review on barriers faced by newcomers. He categorized this in social interaction, newcomers previous knowledge, technical hurdles, finding a way to start and documentation.

3.3.1 Social Interaction

Steinmacher references to 7 studies that have been done finding the correlation between the existence of social interaction between newcomers and core members and contributions by newcomers. Steinmacher categorized this as a barrier called Lack of social interaction with project members.

Christian Bird [2] concluded in a linear statistical study that there was a strong correlation between development behavior and the level of importance that participants have in the social network. Bird measured the importance of a newcomer by the number of emails sent and received. Nicholas Ducheneaut [9] also researched the content of divers emails. He found out that perseverance is needed to newcomers to get some status in the community. Bird based his conclusion on a number of bigger projects in sourceforge.net where Ducheneaut researched social interaction in ANT and Python.

Another aspect on social interaction is how quick questions by newcomers are answered. Kuechler et al. [19] found a correlation between not getting timely answers and loss of motivation by newcomers, but he also concluded that far out most questions of newcomers are answered quickly within 1 or 2 days. Kuechler concluded in the chapter: Joining Free/Open Source Software Communities: An Analysis of Newbies' First Interactions on Project Mailing Lists of their thesis that improper answers, as far as the answers are public, are rare. However analyzing a survey with almost 60.000 respondents, he concludes that it may be hard to break into the tight-knit social networks of OSS developers. The conclusion of the survey seems to be contrary to statistical research, but the statistical research only considered public interaction while in surveys, private messages are considered as well.

3.3.2 Newcomers previous knowledge

Out of a survey from Kuechler et al. [19] among contributors in OSS projects, it appeared that 'Extending skills' is an important motivation to join the OSS community for developers. However, most OSS projects are not equipped to teach newcomers and to introduce a technical introduction to newcomers. Zhou and Mockus [21] concluded that lack of technical experience by the newcomer is a barrier to become a contributor This research was done on the Gnome project and the Mozilla project. For the qualitative research they researched the joining process of 20 successful contributors and 20 unsuccessful attempts.

Stol et al. [26] noticed lack of domain expertise as a hindrance. They used the input of student research and extend it with interviews. Van Krogh et al. focused on Freenet [10]. They interviewed core members and also they concluded the lack of domain expertise as a hurdle to join the project. Schilling et al. [23] researched the retention of former Google Summer of Code students and retention in the KDE project. Students are assessed in this project and from interviews it appeared that lack of knowledge of project practices was one of the reasons why newcomers couldn't join the project.

3.3.3 Technical hurdles

A few studies mention technical hurdles. Stol et al. [26] interviewed 12 students and one of the challenges they met was to setup a workspace and have the project to work on compiled and running. Mereilles et al. [20] mentioned source code complexity as unattractive and in the literature reviews this is mentioned as a technical hurdle while Stol et al. [26] concluded that the software architecture complexity can be a hurdle as well, especially when design decisions aren't accurate documented.

3.3.4 Finding a way to start

Van Krogh et al. [10] investigated the content of a lot of email communication on Freenet and encountered email exchanges between members and developers who were eager to contribute but couldn't find an appropriate task to start with. It would be preferable if newcomers could get a mentor. Canfora et al. [4] introduced a mentoring practice called Yoda. In fact very few of the monitored OSS projects in all researches have a mentoring system.

3.3.5 Documentation

Čubraniç et al. [8] wrote a paper called Hipikat: a project memory for software development. He did empirical tests with experienced software developers in the Eclipse project. Hipikat should be a solution for documentation problems such as there are: outdated documentation, unclear code comments and lack of documentation. Unclear code comments points to comments that are only understandable for insiders in the project. Čubraniç faces this has to do with the way developers use to communicate, mutual with a minimum of effort and a maximum of understanding. That makes them unwilling to deliver appropriate documentation usable for newcomers. Ho-Quang et al. [14] found the presence of UML documentation to be helpful for newcomers. However he couldn't find a correlation between hindering factors and absence of UML.

3.4 Neutralizing barriers for newcomers to OSS projects

Steinmacher et al. [16] proposes a developers joining model to attract, onboard and retain new contributors. When an outsider is attracted he should be onboarded and be seduced to contribute and eventually become a member. The minimization of hindering factors is an important aspect in attractiveness especially to newcomers in the world of OSS projects. Therefore Steinmacher proposes an introduction program to newcomers including mentoring.

Steinmacher et al. [17] declared that barriers don't have to be a problem always. They can lead to an improved quality of future contributions. Some barriers can be used as filters. Moreover, research conducted in the OSS domain demonstrated that socialization barriers are useful for maintaining community integration and the quality of the community's product.

Definition of Attractiveness	Influenced by	Resulting in	#projects	Reference
Attractive to users and sponsors	Type of License Type of User Application Domain State of development	Activeness Effectiveness Likelihood of task completion Time for task completion	~4.500	(Carlos Santos 2013)
Attractive to contributors to join	Structural complexity	Onboarding	6773	(Mereilles 2010)
Capacity of bringing users and developers to a project	# lines of Code	Willingness	6773 2772	(Mereilles 2010) (Chengalur-Smith 2010)
	Upcoming events	Willingness	1	(Ververs 2011)
	Project age Niches Size Project Base Size	Developer Attraction User Attraction	2.772	(Chengalur-Smith 2010)
	Contributor's opportunities	Long time contribution	2	(Minhui Zhou 2012)
Attractive to contributors	Social Interaction, Social status in community	Contributing	200-500 2	(Bird 2011) (Ducheneaut 2005)
Attractive to newcomers	Quick email answers Proper email answers Access to community	Motivation	Unknown (Survey 60.000 users)	(Kuechler, Jensen en King 2013)
	Technical experience newcomer	Contributing	2	(Minhui Zhou 2012)
	Domain expertise	Onboarding	Survey 1 (Survey)	(Stol, Avgeriou en Babar 2010) (G. von Krogh 2003)
	Knowledge project practices	Contributing	1 (Survey)	(Schilling, Laumer en Weitzel 2012)
	Architecture complexity	Contributing	Survey	(Stol, Avgeriou en Babar 2010)
	Other technical Hurdles	Onboarding	Survey	(Stol, Avgeriou en Babar 2010)
	Introduction 1st task	Contributing	1 (Survey)	(G. von Krogh 2003)
	Presence Mentor	Contributing	-	(Canfora, et al. 2012)
	Documentation Items: Lack, outdated, incomprehensible	Contributing	1 (Survey)	(Čubraniç, et al. 2005)

Table 1 OSS project attractiveness definitions

The barriers mentioned in the bespoken previous research together with the motivation mentioned in the introduction have led to the research question in the next section.

4. Research Questions

Research discussed in the previous chapter has already highlighted that newcomers face difficulties when they are willing to start contributing to existing OSS projects. Von Krogh et al.[10] noticed that the lack in transparency in communication often leads to rejected pull requests from newcomers. He expected this to be frustrating but he did not research the effect of this rejection on future project participation for new coming developers. Čubraniç et al. [8] found that documentation issues influence attractiveness and Von Krogh et al. [10] again noticed a problem with newcomers that they don't know where to start.

4.1 Questions and hypotheses

Taking into account the indications of prior research on potential barriers to newcomers to OSS projects, the aim of this thesis is to answer the following research question:

Is the attractiveness of mature OSS projects for new contributors on Github significantly affected by the degree of acceptance of their initial contribution and the existence of technical documentation or a how-to-start page?

Attractiveness in this research is defined as *the ability of the project to attract and retain developer resources*. Attractiveness is in this research measured as: *The number of new contributors on a project performing a minimum of 2 pull requests.* The number of contributors that perform only one contribution is significant but they don't bring sustainability to a project.

This research question is decomposed into 2 sub questions.

Rq1. How strong is the correlation between the degree of acceptance of newcomers' first contribution on mature projects on Github and the attractiveness to newcomers?

Rq2. How strong is the correlation between the existence of technical documentation or a how-to-start page on mature projects on Github and the attractiveness to newcomers?

Associated to these research questions there are 2 hypotheses which we formulate as follows:

H1. A strong correlation is expected between the acceptance of a newcomers first contribution on a Github project and attractiveness of this project to newcomers

H2. A strong correlation is expected between the existence of technical documentation or a how-to-start page on Github and the attractiveness of this project to newcomers

As described in former research and in observations it is expected that absence of easy accessible documentation is a hurdle and rejected pull requests are demotivating.

In the next section the research methodology is described for this research.

5. Methodology

The research question in the previous chapter was divided into 2 sub questions both needing a different research method.

The quantitative research to answer RQ1 took place on a snapshot of the GHTorrent [11] database. For the quantitative part of the research on RQ2 the wiki pages from Github were downloaded to a local storage. The qualitative research for RQ2 took place directly on Github itself and the websites of the projects.

5.1 Data collection

For this research the metadata of Github as retrieved by the GHTorrent Project [11] are used and analyzed along with data from Github project data from the dataset.

5.1.1 GHTorrent Dataset

The GHTorrent project builds a documented database of Github's metadata [12]. The GHTorrent project is supported by TU Delft and Microsoft. A network of contributors query the Github database and the results are merged and persisted at ghtorrent.org. The research has taken place on a snapshot of this GHTorrent database taken from ghrorrent.org the 1st of March 2019.

The relational model of the GHTorrent is found in Appendix A. Most tables are related to the User table and the Project table. Further for each pull request the history is persisted.

5.1.2 Github wiki pages

For RQ2 Github is queried to collect the wiki pages from projects. The number of wiki pages for each project, part of the investigation in RQ1, is determined. This resulted in a collection 61.113 project wikis.

Github doesn't offer a web API to count the number of wiki pages. Therefore, the wiki pages for each project are cloned and counted.

To clone the wiki, the URL of the project is taken and reformatted. https://api.github.com/repos/loginid/projectname is reformatted to https://github.com/*loginid/projectname*.wiki.git.

A system call to execute the cloning resulted in an error message, in case there is no wiki present for the project. If a Wiki is present then a folder named *projectname.wiki* is created. In this folder 5 subfolders exist. Some metafiles, having a name starting with a dash, exist. For the counting, the folders and the meta files were not relevant. The remaining files contain the wiki pages and the hyperlinks to other wiki pages. The number of these files plus the number of unique hyperlinks within these files is equal to the number of wiki pages as part of the project. This number is persisted with the project.

Then it is evaluated if the number of wiki pages is correlated with the attractiveness of the project to newcomers. This step gave an indication about the number of projects that should be reviewed in the next step.

5.1.3 Data collection from project websites

While most collaborating projects on Github also present themselves on a project website, those websites are explored manually to find

supplementary documentation. The reference to this website is found on Github.

5.2 Data selection and preparing for analysis

Having gathered all the data, the dataset is cleaned first. For this some filters are applied on it. Otherwise it would have resulted in too many projects that aren't really software projects or projects that are not created as collaborative OSS work.

The following filters are applied for the selection of projects:

- Language: To ensure that the subjects of research were only software projects, there is a filter on language. Appendix B contains a list of program languages that are part of the research. These are not only popular programming languages but also less popular used program languages;
- Lifetime: only 'mature' projects are investigated, projects are subject of research from the moment they exist at a minimum of two years;
- Size: Projects with only 1 contributor are excluded from this research because these projects are not meant to collaborate;
- Forks: If a project is forked from another project, the original project starting date is taken in account. After this fork the project can be split up into an enterprise developed edition and a Free open edition like Open Office/Libre Office [18];
- Pull requests: Projects not working with pull requests are excluded.

The results of these imports and filtered lead to the following figures:

Table 2: Results import and filters pull request.

Kind of data	# initial import	# after filtering
Users	30.600.306	30.600.306
Projects	116.167.204	4.442.209
Pull requests	47.844.942	19.146.023
Pull request history	82.851.530	42.338.628

In the §5.2.1 until §5.2.4 the process of importing and filtering is described. In §5.2.5 an interim analysis is done to determine if sufficient data exist for performing the quantitative research. In §5.2.6 until §5.2.10 the projects are ranked on their ability to attract newcomers.

5.2.1 Import of user data

At first the Users Table is imported. As demonstrated in Appendix A, this table has no dependencies. The GHTorrent file contains 30.618.798 user lines. During import 18492 lines did not contain valid user data. The disambiguating algorithm to filter multiple id's of one user could not be applied because the structure of the GHTorrent database in real lacked the name and email address in spite of the documentation provided. Although Github is an American site, GHTorrent is situated in Europe and has to respect European privacy rules as discussed by Engelfriet [29]. So this information is removed from this data collection for privacy reasons.

After import, the following results occur:

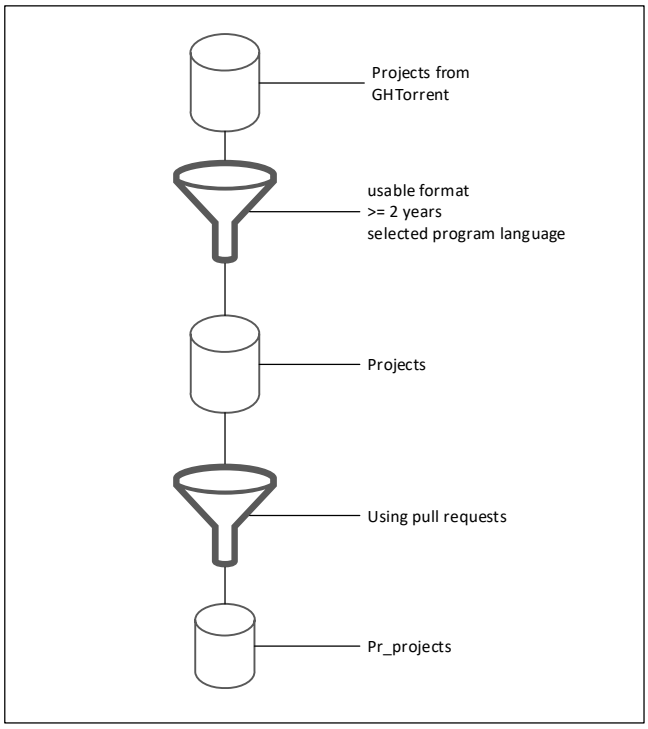
Table 3: User Import

Action/Filter	Start#	Correction#
Initial	0	+30.618.798
Invalid Id	30.618.798	-18.492
Disambiguating	30.600.306	0
Result #users	**30.600.306**	

A qualitative look on the logging showed that the rejected lines had prosaic text in the id field, that should be numeric. It appeared to be an overflow from the previous line containing a lot of text.

5.2.2 Import of project data

[i]Filtering projects before the eventual analysis is done according next scheme:



Scheme 1: import and filter projects.

During the import of the projects, the first filtering took place. Projects are filtered on age and program language.

Table 4: Project Import

Action/Filter	Start#	Correction#
Initial	0	+116.168.027
Wrong format	+116.168.027	-823
Younger 2 years	+116.167.204	-24.222.685
Wrong language	+91.944.519	-64.013.934
Result #projects	**+27.930.585**	

Again a qualitative look at the logfiles learned that lines in wrong format not only were malformed but also contained data not concerning software projects.

Since Github is not that old it was expected that about 20% of the projects in Github were too young to meet the requirements of this research.

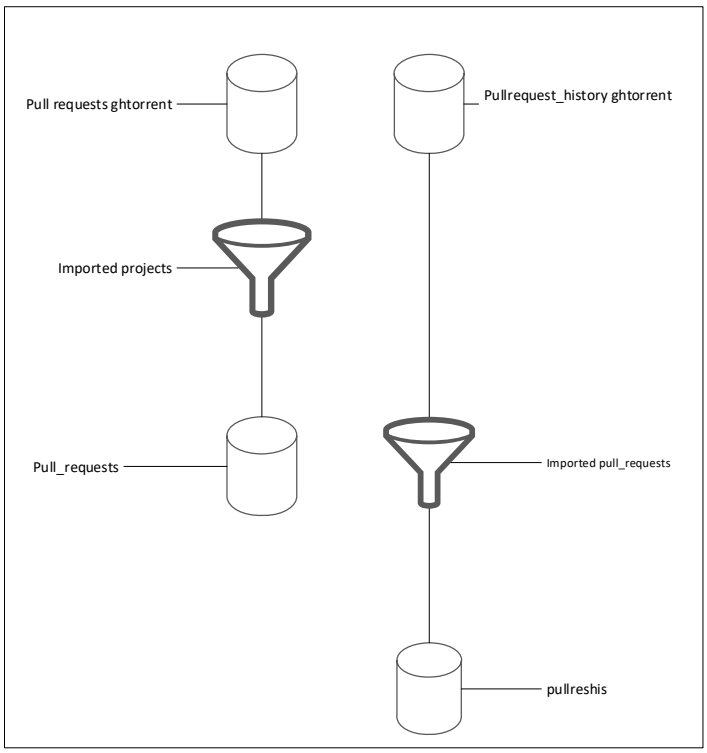
More surprising was the fact that more than half of the projects on Github weren't written in one of the program languages that are part of this research, as listed in Appendix B.

The language in which a project is written, is mentioned in the project file. However sometimes the language mentioned is not equal to the language actually used. Having reviewed random projects, not one of those appeared te be in another language. For this research this satisfies because the programming language itself is not part of the research.

Investigating the shake out because of this, it appeared that many were Unix/Linux shell script projects and also HTML and CSS projects. These projects are out of scope because we aim to focus on software development project. The popularity of the GO language however, was not foreseen. Projects written in GO are not part of the research plan and thus not part of the outcome.

5.2.3 Import of pull-request data

Pull requests and it's history are imported according next scheme 2.



Scheme 2: import filter pull requests and pull request history

Only pull requests related to the imported projects were imported to avoid unnecessary data in the dataset:

Table 5: Pull request import

Action/Filter	Start#	Correction#
Initial	0	+47.844.942
Format error	+47.844.942	-252.150
Project. Unknown	+47.592.792	-28.446.769
Result #pr	**+19.146.023**	

The format error was caused by null values in project ids.

The projects unknown are the projects, filtered in the previous step because they were created less then 2 years ago.

After having imported the pull requests the pullrequest_history table should be imported.

Table 6: Pullrequest_history import

Action/Filter	Start#	Correction#
Initial	0	+82.851.530
Format error	+82.851.530	-231.649
n.e. pull request	+82.619.881	-40.281.253
Result #prhist	**+42.338.628**	

The format errors again concern null values in identifying fields. However the number of them is restricted. The non-existing pull requests concern the pull requests that aren't imported in the previous run.

5.2.4 Filtering out projects without pull requests

A lot of projects on Github are not suitable for this research. Since the research is restricted to projects working with pull requests, an extra filter on this is applied on the project table. To keep this traceable a new table, pr_projects, is created with only the projects with pull requests. This is simply done with a SQL Query:

Query 1 : Removing irrelevant projects

```
select * into pr_projects
       from projects
       where id in
              (select base_repo_id
                     from pull_requests)
       or id in
              (select head_repo_id
                     from pull_requests)

(4442209 rows affected)
```

This filter doesn't affect the outcome of the research but it offers a significant increase on the performance since the number of projects is reduced to 4.442.209. During this filter, an additional check is performed on the consideration in the Import of project paragraph that Linux Shell projects are not considered to be software projects. Therefore, an additional import has been performed to import those projects. However for none of them exists a single pull request. This confirms that this consideration was acceptable and has no effect on the research results.

In paragraph 5.2 it was mentioned that from forked projects the age of the original project should be taken in account. To ensure this a program is executed that investigates these chains of projects. In the pull request the base_repo_id is set to the oldest project id in the chain, so this project will be taken in account to find the maturity of the project.

5.2.5 Interim analysis eliminating risk

The number of projects, subject of this research decreased so quickly during filtering that some interim analysis was requested to ensure that the risk of a too small number of projects wouldn't occur.

At this moment in the research all out of scope data is removed. In the research proposal, the risk was mentioned that, because of the relative short existence of Github, maybe not enough data would be present for the research. If the amount of data is reduced to a number that doesn't meet the demands of statistical research, the minimum age of the projects should be reduced.

For this reason a quick analysis on the filtered imported data is performed.

Before proceeding it should be clear how many pull requests are opened for mature projects by different potential contributors.

The chosen number of different users (> 1, > 2 ..etc.) have no special meaning. This analysis is only done to find out if there are sufficient objects to perform a valuable research.

Table 7 shows the number the result of this analysis. The first column shows the number of different users, having opened a pull request from the moment the project is older than 24 months. The second column shows the number of projects meeting this number of different users. A third column contains the number of follow up pull request actions for the projects on the second column. This might give an indication if there is any correlation between the number of contributors and the number of users involved with the follow up on a pull request.

Table 7: Users active on mature projects

Different users	Projects	Follow Up
>1	119.473	85.235
>2	78.811	94.636
>3	71.027	50.253
>4	60.316	51.416
≥10	36.129	32.872
≥20	21.246	22.290

≥30	15.162	17.367
≥40	11.773	14.369
≥100	4.661	7.750
≥200	2.069	4.263
≥300	1.185	2.891
≥400	785	2.150
≥1000	209	644

Table 7 shows a percentual increase of users involved with the follow up when the number of different contributors increases. The tables also indicates that the number of projects, having many contributors involved, is sufficient for the planned research.

Not shown in Table 7 is that on 88.047 projects only 1 user opened 1 or more pull requests that didn't result in any follow up.

So a rough analysis is done on the number of newcomers. The analysis is rough, because it doesn't take account of the passed time meanwhile newcomers are welcomed. So the number of newcomers could be welcomed in a period of 2 weeks but also within 4 years. It is also rough because newcomers in a project are considered to be new on Github could be active on other project since the researched project became mature.

However the outcome, not being part of the research itself give an indication towards the expected number of projects becoming part of the statistical part of this research.

Two queries have been performed on the current dataset. The first query detects the number of newcomers on projects that haven't performed any pull request on Github before the date the researched project became mature; the second yields in the number of newcomers in a project, independent if they contributed in some other project before.

Table 8: New Contributors on Github

#Newcomers	#Projects
≥1	23.441
≥100	112

As we can see in table 6 the number of brand-new contributors on Github is restricted in mature projects. If we only look at newcomers in the project itself we get other results:

Table 9: New contributors for project

#New Contributors	#Projects
≥1	119.274
≥100	1.656

The numbers exposed here are officially not part of the research. However the research focusses on the figures in Table 9. With more

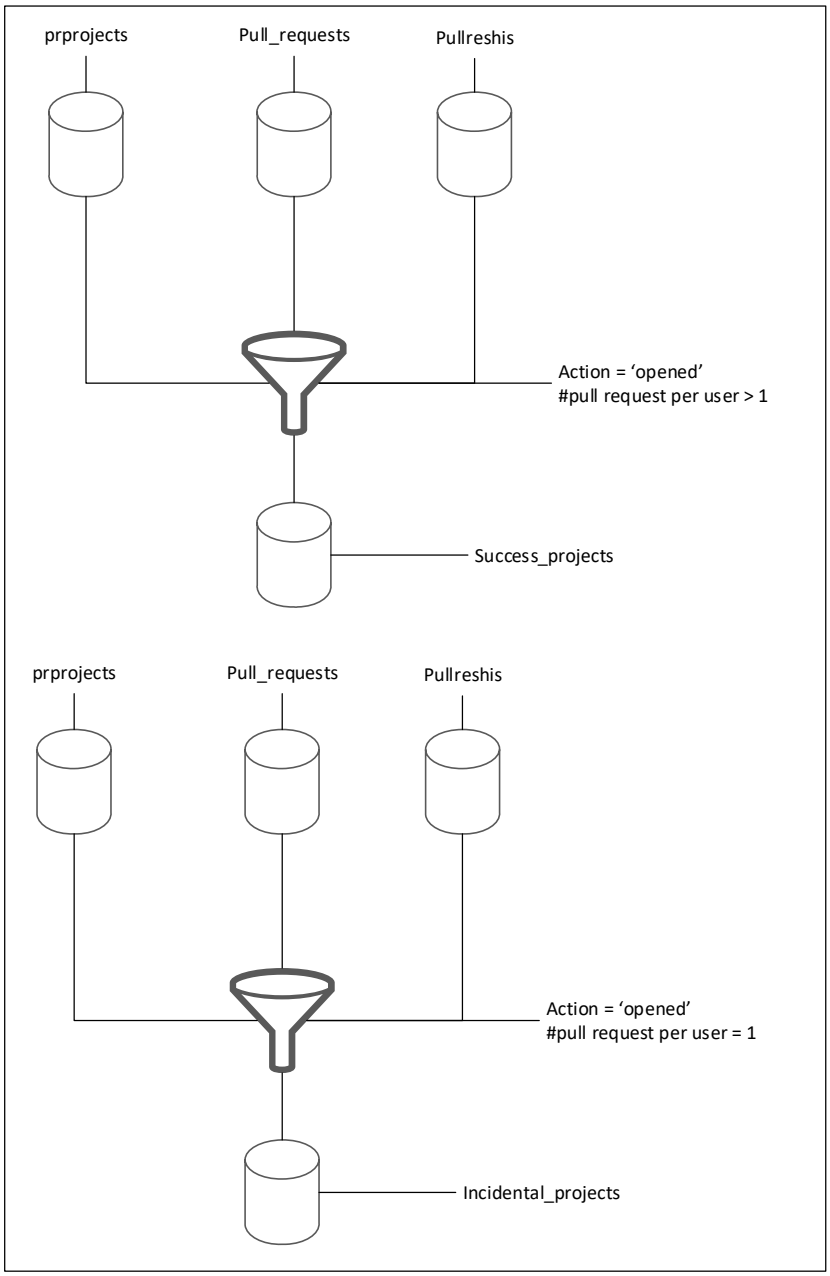
than 1.500 projects attracting more than 100 new contributors it's clearly indicated that sufficient data exist for the statistical part of this research.

5.2.6 Ranking projects successful in attracting new contributors

To find an answer on both research questions, projects should be ranked by their success on attracting new contributors. To achieve this, all pull requests, having a request date 2 years or more after the project creation date, are evaluated using this criterion:

Is this the first pull request by this user for this project?

- When Yes : Is the pull request eventually merged;
- When No : Is there another pull request from this user in this project that was his first pull request.



Scheme 3: Filter projects succeeding in attracting newcomers

After this evaluation, a ranking is made of projects that attract most newcomers per 6 months having performed more than 1 pull request. The successive pull requests do not have to be within the same 6 months but it should be performed once. When no successive pull request is found and the date of the first pull request is less than 9 months before the snapshot date we consider this situation as unknown.

In this research projects are considered to be successful if they manage to attract recurring contributors. The next step now is to find those projects and after that ranking them in the number of attracted recurring contributors per half year.

Paragraph §5.2.7 describes the process illustrated by *scheme 3 while* §5.2.8 describes the process how the ranking on the outcome is created.

5.2.7 Finding successful projects

The more recurring contributors the more successful a project is. Because the total number of new contributors is depended on the age of the project, the average number of newcomers per 6 months is determined. The choice for 6 months is arbitrarily but intentionally chosen because a shorter period will result in very small numbers and a longer period would possibly result in unreliable figures for projects that only shortly have the status 'mature'.

To find out the average number of new contributors per 6 month, a view is created that finds the projects being successful in attracting more at least 1 new recurring contributor after 2 years. Contributors are considered as new if the have not contributed within the first 2 years.

The view is defined as follows:

Query 2 : Create view successful projects

```
CREATE VIEW success_projects as
        SELECT p1.id projectid, prh1.actor_id actorid,
                count(prh1.pullrequest_id) nrofpullreqs
        FROM pullrequest_history prh1
                INNER JOIN pull_requests pr1 ON prh1.pullrequest_id = pr1.id
                INNER JOIN pr_projects p1 ON pr1.base_repo_id = p1.id
                  AND prh1.action = 'opened'
                  AND DATEDIFF(Month, p1.created_at, prh1.created) >=24
                  AND prh1.actor_id NOT IN
                    (SELECT prh.actor_id FROM pullrequest_history prh
                    INNER JOIN pull_requests pr on prh.pullrequest_id = pr.id
                    INNER JOIN pr_projects p ON pr.base_repo_id = p.id
                    WHERE p.id = p1.id
                   AND DATEDIFF (Month, p.created_at, prh.created) < 24)
        GROUP BY p1.id, prh1.actor_id
        HAVING COUNT(prh1.pullrequest_id) > 1
```

This view creates a virtual table with all actors that performed more than 1 pull request on a project older than 24 months that did not perform this during the first 24 months of existence of this project on this project. From the 4.442.209 projects only 94.491 projects succeeded in attracting 1 or more new contributors performing more than 1 pull request after 2 years of its creation. Totally 329.690 new contributors were attracted. Those are not all unique contributors. Contributors are counted for every project they contribute on.

With this outcome, there should be a lot of contributors/project combinations having opened only one pull request and thus no successors. To find out, Query 2 is slightly modified. The last line of it is changed to 'HAVING COUNT(prh1.pullrequest_id) = 1'. This view is also created and called incidental_projects.

This results in 185.256 projects with 745.766 incidental contributors. If we exclude the projects that, apart from incidental contributors, also attract recurring contributors, 126.289 projects remain only attracting incidental contributors so far since they exist for two years or longer.

On projects not occurring in one of these 2 views there have been no pull requests 2 years after the creation date

5.2.8 Ranking successful projects

To generate a ranking in the successful projects some calculations in querying this view are performed:

Recurring contributors are contributors having contributed more than once. To ensure this, this query is executed on the 'success_projects' view, so only recurring contributors are part of the result.

Query 3 : Number of attracted recurring contributors per project per half year

```
SELECT p1.id, p1.name, p1.owner_id, p1.created_at,
   COUNT(sp.actorid) nrOfNewComers,
   (DATEDIFF(quarter,p1.created_at,'2019-03-01')/2) 'Half Years',
   Cast(Cast(COUNT(sp.actorid) as float) /
   Cast( (DATEDIFF(quarter, p1.created_at, '2019-03-01')/2) as float) as float)
       Gemiddeld
FROM success_projects sp
INNER JOIN pr_projects p1 ON sp.projectid = p1.id
   GROUP BY p1.id, p1.name, p1.created_at, p1.owner_id
   ORDER BY CAST (count(sp.actorid) as float) /
       CAST((DATEDIFF(quarter, p1.created_at, '2019-03-01')/2) as float) DESC;
```

This Query resulted in a list of 94.491 mature projects attracting a minimum of 1 recurring contributor varying from 128,63 to 0.05 new recurring contributors per half year

On the second view, a query is performed sorting the result descending on the number of actors opening only 1 pull request.

Query 4 : Find projects successful in attracting incidental contributors

```
SELECT projectid, p.name, p.created_at, COUNT(actorid) FROM incidental_projects
       INNER JOIN pr_projects p ON projectid = id
       GROUP BY projectid, p.name, p.created_at
       ORDER BY COUNT(actorid) desc
```

This Query resulted in 185.256 projects. Matching the results from Query 3 and Query 4 showed 58.967 projects occurring in both result sets and 4.221.429 projects not occurring in one of those 2 result sets..

The results of these queries are presented in the next table:

Table 9: Summarizing results

# Projects working with pull requests	4.442.209
# Projects attracting new recurring contributors after 2 years	94.491
# Projects only attracting incidental contributors after 2 years	185.256
# Projects without any pull request after 2 years	4.221.429
# Projects with incidental AND recurring contributors after 2 years	58.967

Performing a checksum 4.442.209 -/- 94.491 -/- 185.256 -/- 4.221.429 resulted in 58.967. So apparently no errors are in these Queries.

The #projects without any pull request after 2 years, are the projects working with pull requests but not occurring in the 2 queries mentioned above.

The complete result set of this is to be found in Appendix A (RankingWith2OrMorePullrequests.xlsx). The list with projects having incidental contributors (only performed 1 pull request) is to be found in Appendix B (ProjectsSinglePullRequest.xlsx)

A top 30 shows some particular characteristics. A top 15 is very successful in attracting new contributors, from 16 to 26 the difference between the following ranks is small, making a jump from 22,67 to 18,80.

Rank	Project_ID	Project Name	Owner ID	Original Creation Date	#New Contributors	#demi-years passed	Avg
1	6866209	homebrew	27039	20-5-2009 19:38	2444	19	128,63
2	9570147	homebrew-cask	2876023	5-3-2012 02:05	1382	14	98,71
3	4230805	DefinitelyTyped	1597482	25-6-2013 02:37	986	11	89,64
4	1334	rails	8137	11-4-2008 02:19	1694	21	80,67
5	5520	Specs	19222	11-9-2011 11:47	1028	15	68,53
6	7301975	patchwork	2016667	10-1-2014 00:00	677	10	67,70
7	9808223	cgm-remote-monitor	3999983	22-5-2014 00:32	540	9	60,00
8	1229	homebrew	7165	20-5-2009 17:38	1121	19	59,00
9	1142	salt	6936	20-2-2011 20:16	907	16	56,69
10	1992097	framework	20944	10-1-2013 21:27	648	12	54,00
11	1486	ansible	8571	6-3-2012 14:58	657	14	46,93
12	11250	package_control_channel	4009	5-8-2011 03:56	693	15	46,20
13	634	symfony	4808	4-1-2010 14:21	785	18	43,61
14	37	angular.js	159	6-1-2010 00:34	709	18	39,39
15	992	Spoon-Knife	6321	27-1-2011 19:30	551	16	34,44
16	1226	django	7161	28-4-2012 00:47	414	13	31,85
17	1321133	DefinitelyTyped	83128	5-10-2012 16:39	375	12	31,25
18	6824363	spacemacs	1086156	17-12-2012 21:34	351	12	29,25
19	6	cocos2d-x	31	18-11-2010 23:17	467	16	29,19
20	4708601	bootstrap	1106238	29-7-2011 21:19	427	15	28,47
21	20096	yii2	12139	13-2-2012 15:32	395	14	28,21
22	1920	three.js	2532	23-3-2010 18:58	453	18	25,17
23	17694	ceph	22729	1-9-2011 21:41	374	15	24,93
24	5219	scikit-learn	23655	17-8-2010 09:43	407	17	23,94
25	34896252	home-assistant	11394549	17-9-2013 07:29	255	11	23,18
26	202	gaia	1653	3-9-2011 01:38	340	15	22,67
27	8196280	spark	13369	25-2-2014 08:00	188	10	18,80
28	27329	pandas	85274	24-8-2010 01:37	319	17	18,76
29	13294	zeroclickinfo-goodies	6732	6-8-2011 13:26	276	15	18,40
30	3905191	react	2156	24-5-2013 16:15	202	11	18,36

Table 10: 30 most successful projects

5.2.9 Consolidating result sets contributions

The 2 result sets created here are the base input of the next research step to find a correlation between the acceptance of the first pull request and the attractiveness of the project to new contributors. For convenience of further research a new table is created (Requesters) with a Primary Key projectid and actorid in which the output of both views (success_projects and incidental_projects) is imported grouped by projectid and actorid. In fact this offers an extra check on the queries before, because it's not allowed that duplicate keys occur. If a combination actorid-projectid would occur in both views there would be an error.

It results in a table:

Table 11: #contributors

# new incidental contributors	745.766
#new recurring contributors	329.690
#total new contributors	1.075.456

New Contributors in this case means that the contributor started to contribute after the project is mature. Contributors can occur in all lines of this table and even in 1 line a contributor counts for every project he contributes on.

Having found out all new contributors, in next paragraph the aim is tot restrict the research data to only new contributors.

5.2.10 Compressing pull requests and pull request history

In the pull request history, there are still al lot of data not being relevant for this research. We only want to search the pull request of new contributors but in the current tables all contributors are present.

The main benefit of this is an increased performance because *Table 8 shows that* it eliminates over 90% of the complete pull request history.

To do so this new table is populated in 2 steps. The first step is to insert all the ‘opened’ pull requests by new contributors and the second step is to add all other actions on those pull requests.

Query 5 : Adding all relevant open actions from pull requests

```
SELECT prh1.* INTO pullreshis
FROM pull_requests pr1
INNER JOIN pullrequest_history prh1 ON pr1.id = prh1.pullrequest_id
INNER JOIN projects p1 ON pr1.head_repo_id = p1.id
AND prh1.action = 'opened'
AND DATEDIFF(Month, p1.created_at, prh1.created) >=24
AND prh1.actor_id NOT IN (
        SELECT prh.actor_id FROM pullrequest_history prh
        INNER JOIN pull_requests pr ON prh.pullrequest_id = pr.id
        INNER JOIN projects p ON pr.head_repo_id = p.id
        WHERE p.id = p1.id
        AND DATEDIFF(Month, p.created_at, prh.created) < 24)
```

Query 5 filters all pull requests from actors on a mature project who did not open any pull request during the first 2 year of existence of this project. The found requests, with action is ‘open’ are written into a new file pullreshis. With this action, all pull request that are not relevant for this research are eliminated. For reason of leaving a trail this history is copied into a new table, pullreshis, so this action is easy to reproduce afterwards.

When all open requests for pull requests are inserted, all other actions on these pull requests are inserted in the same pullreshis table.

Query 6 : Adding other actions on opened pull request

```
INSERT INTO pullreshis
 SELECT DISTINCT prh1.*
 FROM pullrequest_history prh1
      INNER JOIN pullreshis ph1
        ON ph1.pullrequest_id = prh1.pullrequest_id
      WHERE prh1.action <> 'opened'
```

After executing these queries the number of pull request history rows in the pullreshis table is 2.998.231. The original pullrequest_history table contained 42.338.628 records. This means a significant reduction of the number of records, so an far quicker performance is expected in the next steps.

The next step, to make analysis more simple, is to extract all unique new contributors who have opened a pull request. Therefor a table is created named new contributors.

Query 7 : Filtering only new contributors

```
INSERT INTO newcontributors
 SELECT distinct actor_id
 FROM pullreshis
 WHERE action = 'opened'
```

At this moment all data, not relevant for the research en RQ1 is filtered out. In the next paragraph is explained how the analysis on this data is executed.

5.3 Data analysis for RQ1: Correlation between accepting first contribution and attractiveness to newcomers

For each new contributor is investigated for each project on which he contributed if his first pull request eventually is merged and how many successive contributions are opened after the moment the merging took place. This moment is chosen for causality reasons. If a contributor opens a new pull request before his first one is accepted, the acceptance of the first pull request has no influence on the existence of this new pull request.

To facilitate data analysis, we built a Java program to analyze the contributions per newcomer on a project. The basic algorithm is shown in Append C. A new table is created, called firstcontributionsalt. This new table is designed as follows:

Table 12: firstcontributionsalt

Field Name	Type
Actorid	Int (pk)
Projectid	Int (pk)
Pullrequestid	Int
Merged	Boolean
successors	Int
Successorsmerged	Int

For each newcomer it is persisted if his first pull request eventually is merged and the number of successive pull requests. Also is persisted the number of successors that is merged eventually.

The first attempt to contribute in a Github project is when a new coming developer submits a pull request.

Correlation between acceptance of the first pull request and attractiveness can only be done on projects where newcomers have performed at least 1 pull request. All other projects aren't ranked at all. Those projects will be out of scope for the rest of the research. When nobody has ever done one attempt, the project was probably not meant for open collaboration.

The linear correlation is distilled between the acceptance of the first pull request and whether this user performs a second pull request. In this case we have five options:

1. The first pull request is merged and successors exist;
2. The first pull request is merged but no successor exists;
3. The first pull request is rejected but successors exist;
4. The first pull request is rejected and no successor exists;
5. The first pull request is performed the 1st of July 2018 or later and no successor exists.

If a correlation must be presented between A and B then A ←→ B must be true and implicitly ¬A ←→ ¬B.

So, if 1 or 4 is true, it confirms this correlation. If 2 or 3 is true, the correlation is denied. If 5 is true, the outcome is unsure so it will be ignored. That means the this pull request is removed from the *firstcontributionsalt* table during execution of the analyzing software

This is implemented is following Query:

Query 8 : Examen Relationship

```
select count(*) Pros from firstcontributionsalt where (merged = 0 and successors =
0) or (merged = 1 and successors > 0);
select count(*) Cons from firstcontributionsalt where (merged = 1 and successors = 0)
or (merged = 0 and successors > 0);
```

There should have been significantly more confirmations than denials to conclude that there is a serious correlation between de degree of

acceptance of a first pull request and attractiveness. If the outcome is purely co-incidental, there would be as much instances confirming the hypothesis as instances denying the hypothesis for this research question.

Because the existence of other reasons the number of confirmations should be at a minimum 3 times the number of denials to come to this conclusion.

For linking the result to individual projects another additional request is provided:

Query 9 : Query answering RQ1 per project

```
select a.projectid [Project], p.name, a.actors [Accepted Successors],
       b.actors [Accepted None],
       c.actors [Refused Successors], d.actors [Refused None] ,
a.actors + d.actors [Confirmation], b.actors + c.actors [Denial],
(a.actors + d.actors) - (b.actors + c.actors) [Difference]
from acceptednext a
       inner join acceptednot b on a.projectid = b.projectid
       inner join refusednext c on a.projectid = c.projectid
       inner join refusednot d on a.projectid = d.projectid
       inner join projects p on a.projectid = p.id
order by 9 desc
```

The results of this research are demonstrated in section 6, analyzed and discussed.

5.4 Data analysis for RQ2: Qualitative research on project documentation

To answer RQ2, the project wikis had to be qualitatively evaluated. The technical documentation stored in WIKI pages is investigated. Technical documentation and the quality of an introduction paper (like readme.txt) is subject of manual research because the kind of documentation supplied cannot be retrieved from the metadata. It was the intention to restrict the research to the contents of the wiki pages on Github but many projects have their own project website with more documentation then supplied on Github. This information was therefore also taken in account.

5.4.1 Criteria selection for research

For this research the following criteria are formulated:

- Does a ‘How to Start’- document exist?
- Are contact data for newcomers supplied?
- Does a project website exist?
- Is supplementary documentation provided on the project website?
- Is technical documentation present?
- How is technical documentation characterized?
- Do installation instructions exist?
- Do deployment instructions exist?
- Are development guidelines provided?

The comparison in this aspect is linear. For each group (attractive, medium and unattractive) the results are cumulated because the answer on the questions are binary. The answers are collected in an Excel spreadsheet together with the references to the answers.

5.4.2 Selection of projects

Bird et al. [3] selected projects for qualitative research from Sourceforge.net based on ranking in Stars [1]. Projects on Sourceforge earn their stars based on reviews. In this research for RQ2 the most successful projects in attracting new contributors are selected. This has led to another set of projects being subject of this qualitative research. This qualitative research is done on the 22 most attractive projects. They are compared with the same number of projects that are most unsuccessful in attracting newcomers. A third control group with the same size is selected for demonstrating a causal link. So 66 projects are examined. Originally it was meant to do the qualitative research on the top 26 attractors from the ranking. This cut was made because the between the number 26 and 27 there was a serious gap in attractiveness from 22 newcomers to 18 newcomers per half year. From there further downward the sorted list didn't contain a serious gap.

From this top 26, investigating the contents, it appeared that 2 projects were deleted or ended years ago. Two project weren't meant for development but only to learn working with Github and to learn working with pull requests. This makes the project very popular in terms of number of pull request but these pull request have nothing to do with contributing to the project. So only 22 top attractors remain.

These top 22 attractors are researched. The 26 worst attractors are selected but three of those projects were deprecated. These three projects are not taken in account. From the control group of the same size there was 1 project deprecated and 1 was archived. This control-group is selected splitting up the ranking in equal parts and select a project from every part.

Having gathered all this information, an analysis is done to conclude what aspects, or combination of aspects, cause attractiveness.

6 Research results

Executing the research plan, described in the previous section, the results are described for each research question.

6.1 Result for RQ1: Effect of the acceptance of the first pull requests on the successors

The hypothesis in this was:

A strong correlation is expected between the acceptance of a newcomers first contribution on a Github project and attractiveness of this project to newcomers.

The following rules should confirm this hypothesis:

The first pull request is merged and successors exist;

The first pull request is rejected and no successor exists;

The following rules should reject this hypothesis:

The first pull request is merged but no successor exists;

The first pull request is rejected but successors exist;

The results on this are :

Table 13: Correlation

Confirmations	120.172
Denials	116.255

Rq1. How strong is the correlation between the degree of acceptance of newcomers first contribution on mature projects on Github and the attractiveness to newcomers?

The linear correlation between these factors is approximately +0,017. So the exact answer on this question is a clearly no.

If we take a closer look to the figures and look at the partial results, there are some more significant differences.

Table 14: Detailed overview

Accepted	Successors	#Newcomers
Yes	Yes	54.432
Yes	No	81.575
No	Yes	35.031
No	No	65.407

Attractiveness is also affected by the number of contributions a newcomer is willing to do. This is examined too with the following result:

Table 15: Successors and acceptance

Accepted	# avg Successors	#avg Accepted successors
Yes	8,5	6,6
No	11,6	6,1

No correlation was therefore be found between the acceptance of a first pull request and the existence of successors. There is also no positive correlation between the number of successive contributions and the acceptance of the first pull request.

6.1.1 The effect of code bots

The results so far where unexpected and some more detailed analysis where performed on the data. It appeared that a few users where responsible for thousands of incidental contributions.

The top 4 of them lead to over 3000 projects. To find out who those contributors were, the user information was requested. It appeared that a number of codebots generate pull requests. They were identified by their name and the behavior of generating many pull requests for different projects. So they are not really developers.

To finetune the statistical results all results generated by obvious codebot users should be removed from the research population. For this, contributions from users having a username starting or ending with ‘bot’ will be removed from the population. After this the results will still be affected by some bots but for statistical purpose it isn’t too relevant.

After repeating the analysis on the remaining data it yields in a result of 117.735 confirming the hypothesis and 114.076 denying this hypothesis. So this did not lead to significant difference. The effect of codebots appeared to be very restricted.

6.1.2 Result on hypothesis 1

For this research question a hypothesis was formulated:

H1. A strong correlation is expected between the acceptance of a newcomers first contribution on a Github project and attractiveness of this project to newcomers

The correlation coefficient was calculated. This resulted in 0.08. So, there is no correlation. Even the detailed views do not lead to more then a very week correlation. We must conclude that this hypothesis is not true.

6.2 Results for RQ2: Investigating the project wiki and documentation

The second part of this research includes qualitative research on a selection of projects. The results are separated int three categories: Best Attractors. Worst Attractors and a Control group.

The research took place on aspects that could help to start contributing for newcomers like the existence of a “How to start” document, technical documentation and guidelines. However, a lot of projects do not offer this information on GitHub itself but use their own project website for this. So those websites are visited and information on it is researched.

Table 17: Number of projects meeting the aspects to help newcomers to get started

Aspect	Best Attractors (22)	Worst Attractors (22)	Control (23)
Existence of a 'How to start contributing'-document	22	1	7
Presence of contact data for new contributors	16	1	4
Existence of a project website	22	8	5
Supplementary documentation on project website	15	0	4
Existence of technical documentation	17	0	4
Installation instructions	21	12	10
Deployment instructions	21	4	11
Existence of programming guidelines	21	1	4

Each aspect is described below. In the table you can see that the results in the control group are near to the worst attractors. For this it's good to know that the top attractors attract 23,18 to 128,63 new contributors per 6 months in average. The control group attracts 0.06 to 1.00 new contributors in average per 6 months and the worst scoring group comes to 0.05 new contributors per 6 month. So the attractiveness of the control groups is more near to the bad attractors then to the top attractors. For this reason, while calculating the correlation between attractiveness and aspects, the control group is considered to be a bad attractor. The results are presented in *Table 18.* They are restricted to 2 decimals without rounding. The interpretation of these results are considered to be weak positive when > 0,30, moderate positive when > 0.50 and strong positive when > 0.70.

Aspect	Best vs. Worst	Best vs Worst & Control group
Existence of a 'How to start contributing'-document	0.95	0.76
Presence of contact data for new contributors	0.68	0.67
Existence of a project website	0.63	0.61
Supplementary documentation on project website	0.68	0.67
Existence of technical documentation	0.84	0.73
Installation instructions	0.40	0.31
Deployment instructions	0.77	0.52
Existence of programming guidelines	0.90	0.82

Table 18: Correlation between aspects to help newcomers and attractiveness of projects

Table 18 demonstrates moderate to strong correlation between a number of helping aspects and the attractiveness of projects.

6.2.1 Existence of how to start documentation

All best attractors contain a 'How to start' document. In these documents potential contributors are stimulated to start on all levels such as submitting issues, translating documentation and picking up issues.

In the worst attracting projects only 1 project (Selenium) has a how to start document.

In the Control group there were only 7 projects with such a document. Not all documents in this control group were inviting. Most of them only described the process on how to contribute and those processes are often complex and thus not very inviting, especially when a relative small community of developers want to keep control. A typical example for this is the *embed* project. If one wants to add new functionality, he should first discuss this in the community and when he at least does a pull request a small core group decides whether this contribution is acceptable. There are several criteria mentioned on the site but none of them is described in a way that a potential contributor is able to find out if his contribution meets those criteria.

As demonstrated in *Table 18* out we can say, there is a strong positive correlation between the existence of a helpful 'How to Start' document and attractiveness to new contributors.

6.2.2 Existence of contact processes

16 of the 22 most attractive projects share contact data for new contributors. The contact data aren't supplied via Github but via a project website. Contact data are mostly discussion groups via Google, Discord, Slack and some other platforms. The projects, not supplying contact data, have a discussion possibility within the Github project itself. An email address is rarely supplied.

In the worst performing group again Selenium is the only project supplying contact data. In the control group there are 4 discussion platforms from which 2 are marked as private.

Table 18 demonstrates a moderate correlation between the presence of contact data and attractiveness to new contributors.

6.2.3 Existence of a project website and supplementary documentation

Not all information for new contributors can be found on Github. Many projects have their own website with supplementary documentation. This can be about the way they are organized, a list of desired contributions and information on how to contribute but also some personal profiles of the contributing crew.

Every project classified as part of the best attractors have its own website. All those 22 projects have websites that contain information for potential contributors including supplementary documentation.

From the worst attractors there are 8 projects with a website but 3 of them lead to erroneous results. The others don't supply information on how to contribute.

The control group only has 5 projects with a project website although 4 of them supply supplementary information. So in *Table 18* a trend is visible that the correlation between the existence of this website and attractiveness to new contributors is moderate positive but increases when this website contains supplementary documentation for potential contributors.

6.2.4 Existence and character of technical documentation

The next aspect is the existence and character of technical documentation. From the best attractors in 17 projects exists technical documentation while none of the worst attractors supply technical documentation and in the control group only 4 projects supply this.

From the top attractors this documentation is in 11 cases mostly explaining text and API documentation. Only Ceph supplies UML diagrams with examples. Cocos2d and home-assistant supply an architectural drawing and examples. The other projects only supply API documentation.

In the control group 3 of 4 projects only supply API documentation. Only buildbot supplies written documentation, examples together with API documentation.

As *Table 18* demonstrates there is a strong correlation between the existence of technical documentation and the attractiveness to new contributors. The effect of the kind of this technical documentation could not be measured.

6.2.5 Availability of installation and deployment instructions

Most projects supply installation and deployment instructions. The difference between de best attractors and others is not as big as on the other aspects while there is hardly a difference between the worst attractors and the control group. The installation and deployment instructions mostly offer information for an administrator on how to install and deploy. Information on how to setup a development environment is rare.

The character of this documentation and the small differences between the groups of research show that these instructions do not significantly contribute to attractiveness.

6.2.6 Provision of guidelines for contributing

Most attractive projects (21) supply guidelines for programming while only 1 bad attractor an 4 from projects the control group offer these. From the worst attractors, only Selenium offers guidelines.

The guidelines in the control group are mostly poor and in one case (buildbot) they only tell what you should not do.

Guidelines for contributing are often an extension on a 'how to start' document and technical documentation. In *Table 18* you can see that the correlation between the existence of these guidelines and the attractiveness for new contributors is very strong.

6.2.7 Result on hypothesis 2

For this research question the following hypothesis was formulated:

H2. A strong correlation is expected between the existence of technical documentation or a how-to-start page on Github and the attractiveness of this project to newcomers

The research results make clear that this correlation is very strong. The top attractors all have a how to start document while from the worst attractors only 1 has such a document and the controlgroup indeed has a number in between. Almost the same you can say about technical documentation and guidelines. Since guidelines are an extension on the mentioned documents, they are taken in account here. A correlation on a how-to-start document is 0.95, on technical documentation it is 0.74 and on guidelines it is 0.80.

This research didn't only consider the existence of the documentation or guidelines but as well the kind and usefulness of it. The documentation of attractive projects was far more usefull than the documentation of worse performing projects.

So the conclusion is that this hypothesis is true.

7 Discussion

As demonstrated in the research results there is no significant correlation between the acceptance of the first pull request of a newcomer and attractiveness to new contributors. However between helpful startup documentation and attractiveness for new contributors there is a moderate to strong positive correlation. Especially a 'How to start'-document, technical documentation and contribution guidelines are strongly correlated to this attractiveness.

7.1 Discussion on RQ1: Pull request acceptance

Van Krogh et al. [10] suggested that the refusal of a first pull request for a new contributor has a negative impact on his willingness to get involved further. They found out that project members have their private communication channels that made the project less accessible for newcomers. However in this research there is no serious correlation found between the refusal of this first pull request and the fact if this contributor gives it a second try. On the other hand the acceptance of this first pull request does not seem to be an important motivation to do more. Perhaps this is to be explained by Igor Steinmacher et al. [15] in his study about quasi contributors. Apparently there are a lot of people scanning projects and performing a small contribution. It would be interesting when those quasi contributors will be eliminated from the statistical research.

The question remaining is why do so many contributors, restrict themselves to 1 pull request, when this request is merged eventually. Pinto et al. [13] performed an in-depth study of casual contributors. There are many contributions that only correct typos. A pull request with corrected typos is mostly accepted and eventually merged, but is not the best starting point for becoming a recurring contributor.

Some other contributors are discussed by Steinmacher et al.[15]. Those are contributors performing pull requests over and over without having one merged.

To filter out those contributors, a more qualitative research is necessary on a selection of projects.

After all it's still important to evaluate the behavior of the core developers in a project according to newcomers. It's not a part of this research but it could be interesting to find out what outcome this research will have if the focus is moved to the projects themselves.

7.2 Discussion on RQ2: Documentation

In this research the number of wiki pages on Github is determined. However, a lot of projects have wiki pages on their own website. The most successful project (homebrew) doesn't have any wiki pages at all but has exhaustive documentation on it's own site (brew.sh). Some of the successful projects have wiki pages but they are not editable for everybody. The number of wiki pages was no indication at all for attractiveness for mentioned reasons.

Steinmacher et al. [17] described social barriers for new contributors. They mention a task to start with as an aspect to start contributing as well as a clear 'how to contribute' document. It's clear that successful projects all have such a document and the worst attractors don't. So

this is confirmed by this research and thus it needs no further discussion.

In the same paper, Steinmacher et al. [17], mentioned lack of a mentor and email contact as a barrier for newcomers. In this research I can not find anything of this. In this research all kind of contact data is collected, but email addresses are rare in this. It's not always wise to publish email addresses on public boards because, if easily accessible, it will be part of a spam database very quickly.

Despite the importance that Canfora et al. [4] grants to a mentor, successful projects do not explicitly offer mentoring to new contributors. So this research can't confirm if this really contributes to attractiveness.

Kuechler et al. [19] mention the need of quick and proper email answers as an aspect of motivation. However, in common, contact takes place via discussion groups and Slack nowadays. Answers on questions there are given most of the time by different project members and not by a specific mentoring member. What we have found is that in the control group those discussion groups more often are private while the discussion platforms of the successful attractors are accessible for anyone.

Communication among developers nowadays does not takes place on individual basis via email but via media like Discord, Slack. Discord is a popular social network for gamers while Slack is a secure chat solution mostly in use by companies. This research does not contradict the need for quick and proper answers. Only the contact data for the projects are on another platform.

Čubraniç et al. [8] mention the importance of actual documentation. This research confirms this. The worst attractive projects have an absolute lack on documentation. Succesfull projects are well documented. However, the kind of documentation is not as expected. One would expect architectural design in a formal language, but most supplied documentation appears to be informal prozaic language supplemented with API documentation and source code comments. Čubraniç also mentioned in his research that too much documentation, often incomprehensive, works as a barrier as well as lack of documentation. It's the same as the development in closed software projects since agile development is more common then waterfall projects where documentation was part of the process because every phase had to be accepted by a principal. The agile manifest considers working software more important than exhaustive documentation. Because of agile software development is far more transparant to the customer, this customer gets knowledge about the costs of good conserved design documents, so the customer can accept to have less technical documentation. So, in a community of developers the most important documentation should help a new contributor find a way to start. This confirms the conclusion of von Krogh et al.[10] that an introduction to the first task is an important aspect for onboarding new contributors.

However, Ho-Quang [14] concluded that it's not the absence of UML that forms a barrier for contributing. In daily practice I observed that developers need documentation to start, but don't feel the need of formal documentation and certainly don't like to produce this.

The existence of guidelines is common in successful projects. Schilling et al. [23] mentioned the influence of knowledge about project practices as an aspect for contributing. Because all successful projects have these guidelines seriously described, it might also be very helpful in onboarding new contributors. After all it can be very demotivating not to know if ones contribution meets project requirements. You can say that guidelines are an extension on a ‘how to start’ document.

8 Limitations and threats to validity

There are some limitations and vulnerabilities in this research. A part of them are in the quantitative analysis of pull request acceptance and some of them in the qualitative analysis of the documentation.

8.1 Limitations of quantitative analysis of pull request acceptance

One of the limitations is the aspect that the contribution done by contributors doesn't have to be a software contribution. With products containing content the contribution can also be content. After manual inspection we found that the contributions to the most successful project, homebrew, are in fact installation scripts used by homebrew. It would be an idea to eliminate this project but in that case all 94491 projects should be inspected by hand which is beyond the time limits for this research.

As already mentioned in section 7, the research on attractiveness in case of accepting the first pull request is influenced by quasi contributors. It is possible to eliminate this influence, but in that case a study to the behavior of quasi contributors is required.

The number of wiki pages on Github is not always precisely calculated. This is because of not all projects follow the same standards for it. For example, some projects don't have their index in an outline file but repeat the complete index on every page. This leads to a much higher number of wiki pages in this algorithm. Very few of the projects that were part of the qualitative research, use the Github wiki facility.

8.2 Limitations of qualitative analysis of the documentation

67 projects were selected for qualitative analysis which means that 94424 projects aren't inspected. It could be, if the research would be finer grained, the results change. To ensure a reliable outcome, apart from the best attractors and the worse attractors a control group was selected. All selected projects were inspected on their value for this research. Projects without value for the research were removed as described in section 6. There was a consequent development that the best attractors scored far out the best on these aspects, the worst attractors scored worst on these aspects and the control group in between as expected. A bigger population however could give a better view on the way the technique is documented.

A part of the qualitative research was subjective. The usefulness of a how to start document is a personal judgement based on inspection if the document would provide sufficient information for me to pick up an item. However, this did not affect the conclusion that RQ2 can be confirmed.

9 Conclusion

For this research the following research question was formulated:

Is the attractiveness of mature OSS projects for new contributors on Github significantly affected by the degree of acceptance of their initial contribution and the existence of technical documentation or a how-to-start page?

This question was divided into two sub questions:

Rq1. How strong is the correlation between the degree of acceptance of newcomers first contribution on mature projects on Github and the attractiveness to newcomers

Rq2. How strong is the correlation between the existence of technical documentation or a how-to-start page on mature projects on Github and the attractiveness to newcomers

For Rq1 a hypothesis was formulated:

H1. A strong correlation is expected between the acceptance of a newcomers first contribution on a Github project and attractiveness of this project to newcomers

This hypothesis is concluded not to be true.

For Rq2 the following hypothesis was formulated:

H2. A strong correlation is expected between the existence of technical documentation or a how-to-start page on Github and the attractiveness of this project to newcomers

This hypothesis is concluded to be true.

Returning to the main research question, we can not conclude that the answer on is an undoubtable confirmation. The confirmation on H2 however is very convincing while the negative outcome on H1 is less convincing because some questions are left open for further research on this.

10 Further research

This research highlights several direction for future work. The aspect of the quasi contributors should be researched further. It would be interesting to research their behaviour, so they can be filtered out. Quasi contributors should not be mixed with casual contributors. The same other view on data could exist if we not only consider newcomers on a project but newcomers on Github.

experienced project members and newcomers in successful projects. In many papers, bad habits in communication are considered a barrier. How is this communication initiated since email is not the standard on this anymore? Has this led to better communication between new contributors and experienced project members? Is there a difference between Free Open Source Software Projects and Enterprise Managed Open Source Projects? Does a proper way of communication affect attractiveness?

The third potential research direction is the level and the character of documentation that is required for onboarding newcomers. We would expect some clear component diagrams and class diagrams to document the structure of the software, so that an engineer can more quickly find his way. In the researched OSS projects these are hardly present and most technical documentation is in prosaic text, and it would be interesting to research how do newcomers experience this way of documenting.

11 References

[1] B Ray, D Possnett, V. Filkov, and P. Devanbu. “A large scale study of programming languages and code quality in github.” *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engeneering FSE.* 2014. 155-165.

[2] Bird, Christian. “Sociotechnical Coordination and Collaboration in Open Source Software.” *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance.* Washington DC USA: IEEE, 2011. 568-573.

[3] C. Bird, A. Gourley, P. Devanbu, M. Gertz, A. Swaminathan. “Mining email social networks.” *Proceedings of the 2006 International Workshop on Mining Software Repositories.* 2006. 137-143.

[4] Canfora, Gerardo, Massilimilliano Di Penta, Rocco Oliveto, en Sebastiano Panichella. “Who is going to Mentor Newcomers in Open Source Projects.” *Proceedings of the ACM SIGSOFT 20th International Symposium on the foundations of Software Engeineering.* New York: ACM, 2012. 44:1-44:11.

[5] Carlos Santos Jr., J Michael Pearson, Fabio Kon. “Attractiveness Of Free And Open Source SoftwareProjects.” *ECIS 2010 Proceedings.105.* 2010. 1-12.

[6] Carlos Santos, George Kuk, Fabio Kon, John Pearson. “The attraction of contributors in free and open source software projects.” *Journal of Strategic Information Systems*, 2013: 26-45.

[7] Chengalur-Smith, IN. Sidorova, A. and Daniel, S.L. “Sustainability of Free/Libre Open Source Projects: A longitudina Study.” *JAIS Vol. 11*, 2010: 657-683.

[8] Čubraniç, Davor, Gail C. Murphy, Janice Singer, en Kellogg S. Booth. “Hipikat: A Project Memory for Software Development.” *Transactions on Software Engineering*, 2005: 446-465.

[9] Ducheneaut, Nicolas. “Socialization in an Open Source Software Community: A Socio-Technical Analysis.” *Computer Supported Cooperative Work* (Springer), 2005: 323–368.

[10] G. von Krogh, S. Spaeth, K. Lakhani. “Community, joining, and specialization in open source software innovation: a case study .” *Sciencedirect*, 2003: 1217-1241.

[11] Gousios, Georgios. “The GHTorrent dataset and tool suite.” *Proceedings of the 10th Working Conference on Mining Software.* San Francisco, CA, USA: IEEE Press, 2013. 233-236.

[12] Gousios, Georgios, en Diomidis Spinellis. “GHTorrent: Github's data from a firehose.” *Working Conference on Mining Software Repositories (MSR).* Zurich, Switzerland: IEEE, 2012.

[13] Gustavo Pinto, Igor Steinmacher, Marco Aurélio Gerosa. “More common than you think: An in-depth study of casual contributors.” *23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER).* Osaka: IEEE, 2016. 112-123.

[14] Ho-Quang, Truong, Regina Hebig, Gregorio Robles, Michel R.V. Chaudron, en Miguel Angel Fernandez. “Practices and Perceptions of UML Use in Open Source Projects.” *ICSE SEIP*, 2017: 203-2015.

[15] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, Marco Aurélio Gerosa. “Almost there: A study on quasi-contributors in open-source software projects.” *IEEE/ACM 40th International Conference on Software Engineering (ICSE).* IEEE, 2018. 256-266.

[16] Igor Steinmacher, Marco Aurélio Gerosa, David Redmiles. “Attracting, onboarding, and retaining newcomer developers in open source software projects.” *Workshop on Global Software Development in a CSCW Perspective*, 2014.

[17] Igor Steinmacher, Tayana Conte, Marco Aurélio Gerosa, David Redmiles. “Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects.” *Proceedings of the 18th ACM conference*

on Computer supported cooperative work & social computing. Hong Kong: ACM, 2015. 1379-1392.

[18] Jonas Gamalielsson, Björn Lundell. "Sustainability of Open Source software communities beyond a fork: How and why has the Libre Office project evolved?" *The Journal of Systems and Software*, 2014: 128-145.

[19] Kuechler, J. Victor, Carlos Jensen, en Scott King. *The Emergent Qualities of Diversity in Free and Open Source Software Communities: A Critical Review and Theoretical Discussion.* Thesis, Oregon State University, 2013.

[20] Mereilles, P.,Santos, C., Mranda, J. Kon, F.,Terceiro, A. and Chavez, C. "A study of the relationship between source code metrics and attractiveness is free software projects." *Brazilian Symposium on Software Engineering.* 2010. 11-20.

[21] Minhui Zhou, Audris Mockus. "What make long term contributors: Willingness and opportunity in OSS community." *International Conference on Software Engineering.* 2012. 518–528.

[22] Morbitzer, M. (2012). *The Mifare Hack*. Visited on 06 10, 2020, from http://proxmark.nl/files/Documents/13.56%20MHz%20-%20MIFARE%20Classic/The_MIFARE_Hack.pdf

[23] Schilling, Andreas, Sven Laumer, en Tim Weitzel. "Who Will Remain? - An Evaluation of Actual Person-Job and Person-Team Fit to Predict Developer Retention in FLOSS Projects." 45th Hawaii Internationa Conference on System Sciences. IEEE, 2012. 3446-3455.

[24] Shah, Sonali K. "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development." *Management Science*, 2007: 1000-1014.

[25] Steinmacher, I., Graciotto Silva, M.A., Gerosa, M.A. Redmiles, D.F. "A systematic literature review on the barriers faced by newcomers to open source software projects." *Information and Software Technology*, 2015: 67-85.

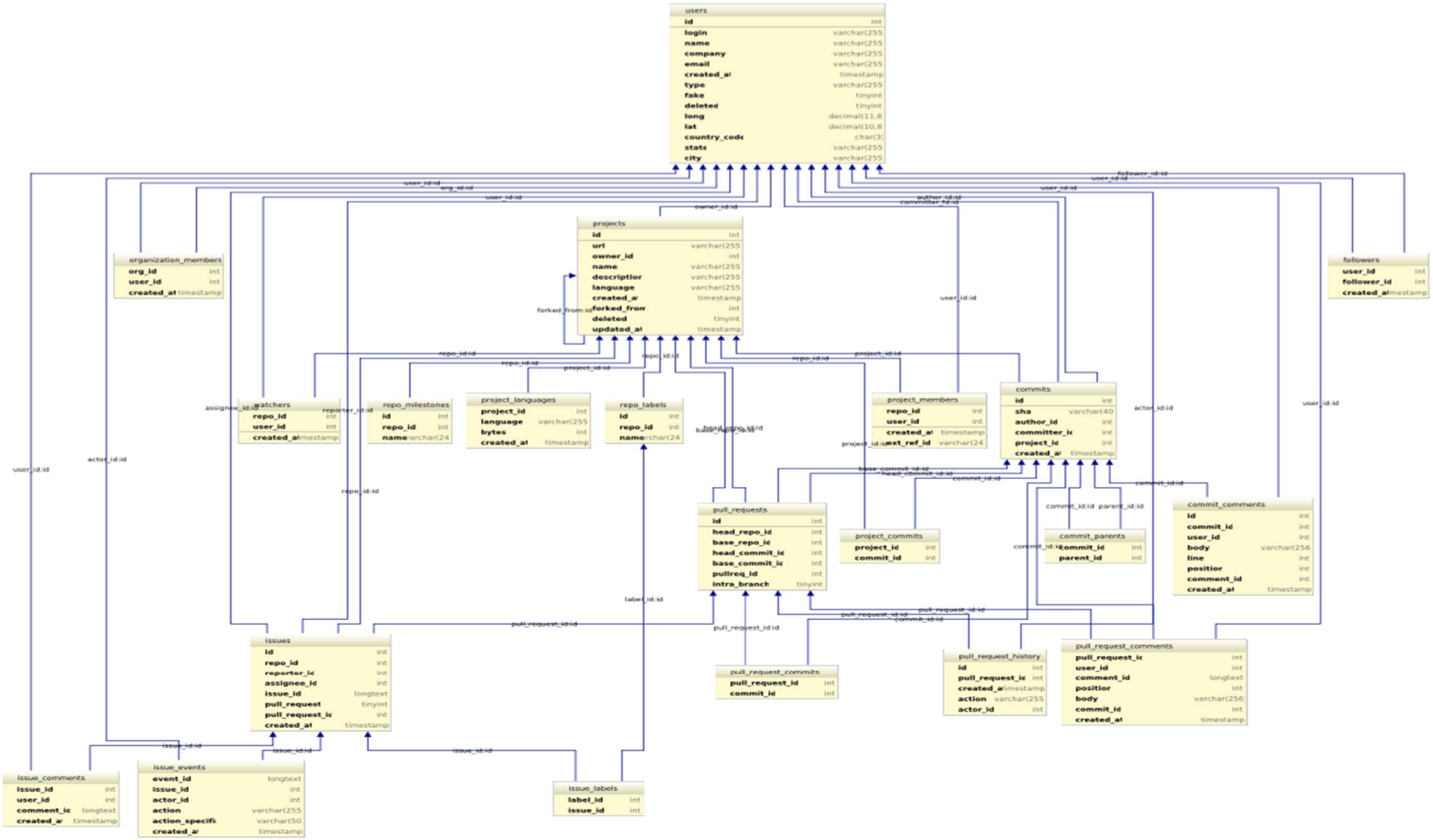
[26] Stol, Klaas-Jan, Paris Avgeriou, en Ali Muhammed Babar. "Identifying Architectural Patterns Used in Open Source Software: Approaches and Challenges." *Proceedings of the 14th International Conference on Evaluation and Assessment in Software Engineering.* Swintin UK: British Computer Society, 2010. 91-100.

[27] Ververs, E., Bommel, R., and Jansen, S. "Influences on developer participation in the Debian software ecosystem." *Intl. Conference on management of Emergent Digital Ecosystems.* San Francisco(California): ACM, 2011. 89-93.

[28] Yunwen Ye, Kouichi Kishida. "Toward an understanding of the motivation Open Source Software developers." *ICSE '03 Proceedings of the 25th International Conference on Software Engineering* . Portland: IEEE Computer Society Washington, DC, USA ©2003, 2003. 419-429.

[29] Engelfriet, Arnout. *Is it legal for GHTorrent to aggregate Github user data?* 28 February 2016. https://legalict.com/2016/02/28/is-it-legal-for-ghtorrent-to-aggregate-github-user-data/ (visited december 12, 2020).

Appendix A : Relational Schema of GHTorrent dataset



Appendix B: Programming Languages taken into account

Smalltalk	Bison	PowerBuilder	Clojure
UnrealScript	COBOL	Assembly	Haskell
Yacc	CoffeeScript	JavaScript	Logos
YAML	Delphi	Kotlin	NewLisp
BlitzBasic	Elixir	Lex	Objective-C++
Brightscript	Elm	Processing	AppleScript
F#	Objective-C	Ruby	FORTRAN
HyPhy	Perl	C	Rascal
JFlex	Perl6	C#	Java
Objective-J	Prolog	Swift	LiveScript
Pascal	Scala	Visual Basic	Lua
PHP	C++	Ada	Groovy
REALbasic	PureBasic	ActionScript	ABAP
TypeScript	Python	Lisp	

Appendix C: Java Source Code for import and analysis

ImportUsers.java

ImportProjects.java

ImportPullRequests.java

ImportPullRequestHistory.java

ImportWiki.java

FirstPRAnalyzer.java (Program to analyse the existence results of first pull requests and successors)

Additionally needed JPA-files and utilities

This Appendix is to be found in the file Appendix_C_Java Source_Code.zip

Appendix D: Results of qualitative research on documentation

Spreadsheet with outcome and references research on websites for RQ2.
This Appendix is to be found in the file Appendix_D_qualitative_research.ods