Practical Study of Visual Studio Code

Practical Course—Contributing to an Open-Source Project

Michael Plainer
michael.plainer@tum.de

March 16, 2021

Visual Studio Code is a free* and open source* code editor actively developed by Microsoft. After an extensive research period, I have contributed to Visual Studio Code as an external contributor over the past couple of months. During this period, my involvement was continuously documented and assessed. This paper presents and discusses findings of this involvement, especially regarding the aspects of open source development and best practices. Most importantly, those findings include a reflection of my contributions and also dedicate a large portion to the analysis of the project's internal structure.

Contents

1. Introduction

Microsoft develops the free* code editor Visual Studio Code (*VSCode*), that is used by more than 11 million users [1]. In the StackOverflow developer survey from 2019, about 50% of the participants stated that they use VSCode [2], showing how popular this editor has become. The underlying source code is considered free (as in free speech) [3, 4] and referred to as *Code - OSS*. Microsoft uses Code - OSS as a base, slightly modifies it (e.g. adds a marketplace integration for distributing plugins), and releases it with a proprietary license under the name "Visual Studio Code" [5]. Although this custom license makes VSCode technically speaking non-free and non-open source [3, 6], other distributions of Code - OSS are free and contain substitutes for the missing features, for example VSCodium [7]. Further analysis of VSCode and Code - OSS regarding aspects of FOSS (free and open source software) development can be found in the extensive preliminary study [8] that was conducted before the project phase.

1.1. Revisiting the Preliminary Study

Before starting to collaborate as an external contributor, I analyzed the development of VSCode in a wide range of aspects such as: how collaboration and communication works, organization of the team, involvement of the community, etc. [8]. This research was an essential base to contributing, which allows me to more thoroughly compare how the actual processes deviate from the documentation, what aspects should be praised, and allows me to suggest improvements. Researching and writing this preliminary report has set expectations and a general mood—which of those were met and which were not are discussed in the following paragraphs.

At that time, it seemed that most decisions are made in public and that the roadmap and monthly iteration plans communicate the direction with the community rather well [9, 10]. While the latter holds, I was not able to find any discussion where a contributor discusses the general direction of the project with the team directly.

Since the project is used so widely, most information was easy to find. This generally also holds when working with VSCode directly. Regularly needed information is very well documented, such as how to get started, run the project, or debug it [11]. After cloning VSCode, it has sensible default configurations with automatic code linting, support for development containers, suggesting plugins on startup, and so on [11]. On the other hand, finding specific information often becomes overwhelming and difficult. In those cases, the easiest way is to ask a team member. The only downside to this is that while team members are generally very friendly, receiving answers can take a significant amount of time. I have already encountered this in the preliminary analysis when inquiring about the governance structure and team members. Those long response times often made collaboration with the team difficult (see section 6).

Finding information stored inside issues often proves problematic as the project has a lot of issues and GitHub's search functionality is limited [12]. As of now, there are over 100,000 closed issues with more than 4000 open issues, making it difficult to identify duplicate issues or to find out if someone has already faced a similar problem [13]. Often it is easier to use a search engine and limit it to the GitHub repository [14].

2. Tickets

Due to the size and wide use of VSCode, the backlog has become very large over the years. Thus, it is difficult for the maintainers to give each thread the attention it deserves. Furthermore, makes it challenging for external contributors with limited time to keep a high-level overview. And while they perform an issue grooming iteration once a year [15], it would be even more useful to hire someone specifically for this task.

2.1. Situation of Backlog and Tasks for Collaborators

The general guideline for external contributors is that issues marked as *bug* or with *help wanted* are free to be worked on by the community. All other issues, especially those with the label *feature-request* need a discussion whether external contributions will be merged and how the implementation should look like [11]. This gives the impression that the team decides which and how features will be implemented, while the community should only help with rather unimportant tasks. The milestone *On Deck* contains a small number of tasks that may be important soon and of those, there are nearly no issues with the label help wanted [16]. Some tasks that were marked with help wanted get this label removed before entering the On Deck milestone, such as [17].

This makes it difficult for collaborators to find places to help that are also appreciated and relevant in a timely manner. Most of the issues with the label help wanted are currently in the backlog, meaning that they are not needed anytime soon and are merely "nice to have" [18]. As those issues are not crucial, they are often not so well thought out. In some cases, once a collaborator tries to implement them, the issue will be closed without giving thorough reason, as to why the feature is not needed anymore [19, 20, 21, 22]. Relevant issues are usually directly created by or assigned to a maintainer, responsible for implementing this change without ever receiving a help wanted label. Hence, contributors typically do not have the chance (or trust) to submit impactful fixes or features [23, 24].

2.2. Lack of Good First Issues

Finding suitable issues to work on as a contributor can become very tedious. One problem with the current state of the backlog is that there are very few open issues labeled help wanted (265, about 5% of open issues), and nearly none good first issues (15, less than 0.5%) [13]. Especially the lacking number of good first issues is alarming for an open source community that tries to be welcoming. Those issues are the starting points for interested newcomers and help building a welcoming community [25].

Those 15 open issues that are currently marked as good first issues were all labeled by one of three maintainers. Although more members are involved in already closed good first issues, those three assignees are very prominent [13]. Kulla says, "motivating others to join the project and for them to start contributing something meaningful takes time and effort. It needs a lot of guidance, interaction, and hence a significant amount of time from a maintainer until someone feels comfortable enough in the project" [26].

Those three maintainers seem to particularly care about more people joining the community and collaborating. Good first issues usually already have a lot of helpful information to get started with. This typically includes an extensive description of what has to be achieved and also code pointers [27, 28]. But due the low number of issues labeled good first issue (or help wanted) compared to the number of interested contributors, it

is often difficult to find an appealing issue to work on. Labeling more issues for external contributors could possibly help the community to grow even further and quicker.

2.3. Selecting Issues

Since the number of good first issues is manageable, my starting point and the one I would recommend is to go through all those issues and decide on one that sounds interesting and fun. The best way to filter for those is to use the integrated search functionality of GitHub [12, 29]. After contributing to the first issue, it becomes more difficult to find an appropriate issue that is free and has enough information provided. To efficiently navigate through the backlog, I created the search filter

```
is:issue is:open label:"help wanted" interactions:>3 -label:"upstream"
↪  -label:"needs more info" -linked:pr
```

to only list interesting issues. In a following, there is a brief explanation of this filters.

`interactions:>3` Requiring a minimum number of interactions removes issues that are not yet discussed thoroughly. In some cases, help wanted issues are also used internally, when input from another maintainer is requested (i.e. not specifically help from an external contributor) [30]. Both of these cases are typically resolved with this filter.

`-label:"upstream"` Removes issues where the cause is not in this repository, but requires changes in another, upstream repository.

`-label:"needs more info"` In rare cases, there is more info required (typically from the creator) and hence this issue cannot be worked on directly by a contributor.

`-linked:pr` This part filters out issues that have a linked PR (pull request). This removes help wanted issues that already have an associated PR that might fix this problem. But on the other hand, this discards issues where an existing PR was not able to fully solve the issue and further work is still needed, as it happened in [31].

Once an adequate issue has been found, it is suggested to state that you work on this issue, which also follows best practices of FOSS development [11, 32].

3. Summary of Contributions

At the time of writing, I was involved in 11 *issues* (6 of which are now closed) and created 4 *PRs* (where 1 has been merged). Furthermore, I created 1 *PR* in the VS-Code wiki that has been merged. In the following, my involvement will be briefly summarized and in section 4, one of the more ambitious contributions is described (and thus omitted in this list). The following also lists the most important interactions and communications I made with the community. A full up to date list of contributions and communications in the main project can be found at [33].

3.1. Issues contributed to

In this section, my involvement in issues without a corresponding PR will be described. In those, my request whether this issue is available to work on was turned down due to various reasons or I simply tried to help.

3.1.1. Suggest File Names when Creating Git Commits

In issue [21], the author requested the feature that when creating a git commit with only one staged file, the file name is suggested for auto completion. Another person already tried to implement this feature but abandoned it. I looked into how it can be implemented and suggested to implement it similarly to code completion, allowing to complete the name of any staged file. To follow FOSS best practices and since the assignee removed himself from the issue in the meantime, I asked if this issue is still something the team would like to see implemented. The assignee said that now there is an API that allows accessing staged files and this should rather be implemented as an extension and closed the issue.

3.1.2. Changing Launch Configuration only opens once

When investigating on what issues to work on, I found the issue [22]. This describes a bug where, when changing the launch configuration of a program, a popup appears. But once this window is closed, it cannot be opened again. I could not reproduce the issue and suggested closing it (which it was).

3.1.3. Silent Updating

The issue [20] describes the problem that when manually checking if there are any updates for VSCode, the user is only informed when there are no updates. If there are any, it silently updates in the background with no visual indicator. When taking a closer look, I found out that in some cases the user is informed and that this feature was once implemented but the commit reverted. When asked what the problem was and if I could help, the issue was closed with the label *wont-fix* without reasoning. When another user asked as to why, the assignee did not respond anymore and the thread was locked a few weeks later.

3.1.4. Customize the Location of ".vscode" Directory

A user asked in [34] if it would be possible to change the path of the ".vscode" directory (i.e. the path containing configuration and plugins). I suggested a fix, linked to another issue where this has already been discussed, and asked a member of the team whether we should allow for another way to configure this setting. He answered that there are no plans for this and since I have already answered the question closed the issue.

3.1.5. Setting to not track the active File in the Explorer

Another issue where I was involved in was issue [35]. The author could not find a way to modify the behavior of VSCode so that the currently open file is not highlighted in the file hierarchy and hence created a feature request. I pointed to the right configuration and to documentation about this setting. It was then quickly closed.

3.1.6. Question regarding Issue Handling and Transparency

During my involvement in the project, I noticed a few areas where the team could improve. Most importantly, I suggested in [36] to reduce response times, define a process what to do when a contributor does not get a response, and increase transparency. Kai

Maetzel, the engineering manager [37], stated that there is no such procedure in place, briefly sketched reasons on why a specific team member may be unavailable, and asked for problems I got stuck on. As this question was more of a general discussion, I tried to elaborate further, steer the conversation into a more general direction, and stated that such a procedure would be beneficial. So far, there has not been a response from Kai Maetzel anymore.

3.1.7. Inquiry about Contributor Survey

In December 2020 a developer survey aimed towards external contributors of VSCode was conducted. The incentive was to evaluate the satisfaction of active collaborators and areas where to improve. The survey said that the results will be made public soon. But those results were never published, and I created issue [38] to ask when they will. Until now, neither have I received an answer nor were the results made publicly available.

3.2. Description of my PRs

The following sections summarize the PRs I have created and the corresponding underlying issues. Those descriptions document the features and implementation but also include most communications with maintainers.

3.2.1. Read Terminal Names by Narrator

VSCode includes the functionality to open terminal windows directly in the editor. Those terminals can have titles, either automatically generated (e.g. powershell) or set by the user. Issue [31] describes that renamed terminal windows are not read out by text-to-speech applications, causing accessibility problems. This issue has been ongoing for a while and a proposed fix has not resolved the problem yet. As this was my first issue, I said so and asked for a code pointer. My question was left unanswered, but I was able to resolve the issue and created PR [39]. The PR was assigned to a member of the team that even after several pings never responded in any form. Fortunately, another user involved in the underlying issue [31] offered to help and pinged a different maintainer. He then reviewed my code and pointed out an edge case I did not consider. After a fix, the PR was merged on the same day.

3.2.2. Allow empty Repositories to be synced

Issue [40] is about the included git integration in VSCode. After syncing an empty repository it cannot be synced via the integrated UI. Meaning, that when changes are committed via a different mean (e.g. GitHub web interface), those commits cannot be fetched anymore. As this problem seemed difficult, I investigated it before communicating with the team. Although one should not develop in private [41], I wanted to get more familiar with the code base to have a proper discussion. After becoming comfortable enough I listed one of the changes I believe to be necessary and asked a few questions regarding the code. The assignee seemed to not have read the issue text properly and claimed that the current behavior works. So I clarified the problem. Leaving all my questions unanswered, the assignee then said we can experiment in a PR. After I resolved the issue, I created the PR [42] and answered my own questions in the issue for the case that someone else is looking for them. To make sure that my changes work

properly, the assignee of the PR asked git maintainers what they think of my changes in [43]. We decided to use my implementation as a fallback, but I have not received a code review yet and several pings were ignored.

3.2.3. Do not update disabled Extensions

Extensions in VSCode are automatically updated in the background, but the community would prefer that disabled extensions are not updated automatically, as described in issue [44]. I asked whether this feature was still available to work on and promptly received a code pointer. I tried to discuss the implementation I had in mind and asked how to properly test this feature. I have not received an answer but was able to implement the changes myself and created a PR [45]. After several pings, I received a thorough code review with an additional request to make the setting also available in the UI. I have implemented most of the requested changes but due to some local build issues, it will take a little more time before requesting a new review.

3.2.4. Rename Master Branch in Documentation

Because the term "master" has a negative literal meaning in the context of slavery, it is considered inappropriate [46]. Hence, GitHub decided to change the default branch to main [47] and the VSCode team followed [48]. But since the documentation was not updated, I created a PR [49]. In there, I updated links and git commands in the pages most relevant to newcomers (e.g. how to contribute). The change received two reviews and was merged within a day.

4. Case Example: Recursively expand Folders when Moving

Now let us investigate how a contribution might look like once one has already become familiar with the project and its etiquettes. In the following, we will take a look at at issue [50] and the associated PR [51]. In [50] the author describes that when moving a folder into a directory where a folder with the same name already exists, the new folder overwrites the old one. For example, imagine moving the folder `main` into a directory `src/`. But `src` already contains a folder named `main`. VSCode will now completely overwrite `src/main` with the content of the other, new folder. Most operating systems on the other hand, (e.g. Windows, Ubuntu, ...) will combine the contents of the two folders `main` and `src/main`.

The team acknowledged that in such a case, the folders should be combined. After a brief discussion by the team, the feature was investigated internally. After about three months, the assignee (@isidorn) roughly sketched how it could be implemented and gave a code pointer. He also warned that this might be difficult to implement. I investigated the cause of the issue and stated what I believed was the best way of achieving this new behavior. I then documented important information regarding this issue in my response and was advised to create a draft PR and discuss the implementation there.

I implemented the core behavior in a draft PR [51] and discussed a few aspects of the implementation. When merging folders, file conflicts can occur, so @isidorn suggested to implement a popup asking the user to "Replace All", "Skip All", or to "Decide Individually". After a brief discussion, the required changes were implemented, and the PR marked ready for review. After a few days, @isidorn stated his concern about the

fact that so many lines of code were introduced, making it difficult for him to maintain it in the future. Thus, I suggested to make the code more compact without impacting readability and justified that a portion of the code was introduced for the UI.

After reducing the size, @isidorn briefly reviewed the code for the first time and said that we should remove the "Decide Individually" feature he requested before, where users can decide for each conflicting file which one should be used. Another feature was then requested to make it more versatile, which required some further discussion. But since I did not receive an answer and changing it would be nearly no effort, it was implemented the way I believed to be better.

While the code was ready in February, the PR was first assigned to the Backlog, then pushed into the February milestone, pushed back into the backlog, and was put into the March milestone. After being in this milestone for a few weeks, it was once again pushed back into the April milestone without any notice. I am currently awaiting a review regarding the latest changes.

I perceived the communication very friendly but a more thorough analysis before suggesting the feature could have spared me from implementing a change that was not desired after all. And as with other communications, response times tended to be high. This can have a negative impact on the motivation but this will be discussed in section 6.

5. Communication

Interaction and communication between collaborators are a crucial part in every development process. As seen world-wide with the current pandemic, communication becomes especially difficult when the team is not used to asynchronous means such as chats [52]. When trying to maintain an active FOSS community, a transparent, authentic, and open communication can be the key to success [53, 54]. In the following sections, the general means of communication and my interactions with the community are analyzed and discussed.

5.1. Involvement in Communication

As described in the preliminary analysis [8], in the development of VSCode all public communications are issue-centered, meaning that they either happen inside an issue or a PR. Although there are IRCs for the community, active participation of maintainers in those channels is usually limited [55]. Apart from some messages in those IRC channels, my communication with the maintainers was solely in issues. Highlights and important aspects of those communications are described in section 3 on a per issue base. In most cases, this communication was directly about features or discussing the code on some levels (e.g. [31, 40, 50]). But in other cases, I tried to discuss procedures [36], inquire some unpublished information [38], add valuable information by referencing relevant issues [34], or by answering questions [35]. During the whole communication I tried to adhere to the FOSS development best practices by being polite, open to other ideas, quick in responding, and keeping the conversation on-topic [32].

5.2. Communication in General

I perceived collaborators and communication very friendly. Maintainers seemed to appreciate the effort that is put in and are happy to see a growing community [22, 31, 36,

39]. While responses never seemed unfriendly, they were in some cases so short, that they missed the point [20, 45]. Sometimes, only the outcome is described without giving reason and further questions are even ignored in some cases [20, 34]. In other cases, the communication partner did not take the time to fully read the message (or the history leading to the message) [36, 40].

Furthermore, while maintainers seem to trust the opinion of their coworkers and regularly ask for input from them [44, 50], they generally seem skeptical towards suggestions from the community [20, 21, 34]. The lack of trust is generally bad for an open source community, but this may be an inherent problem with a large community: many suggestions, especially feature requests, are subjective, not well researched, or simply not fully thought through [56]. On the other hand, I am not aware of any occurrence in the project, where a well thought out technical issue with a concise description is actively turned down without the possibility for discussion. It seems like generally, there is always room for technical discussions in the project, which is crucial for a FOSS community.

6. Analysis of Response Times

One of the main reasons for discussions to die down is the unresponsiveness of team members. Discussions about specific code sections and highly technical ones as well as more fundamental issues, such as accessibility problems, are often left unanswered. In those and many other cases, the issue will remain in the backlog for the foreseeable future. Pings in those issues are usually ignored [20, 57, 58]. The unresponsiveness of (certain) team members is one of the major problems I have encountered during my collaboration.

Some team members, such as @isidorn or @Tyriar, stood out for excellent response times [22, 34, 39, 51]. Others only responded after (several) pings, if at all [39, 42, 45]. Figure 1 illustrates the response time of maintainers to my messages. Independent right-censoring is assumed meaning that the end date and hence the duration might be censored [59, p. 135]. This is the case when an answer would not make sense anymore (e.g. a question that I answered myself) [31, 44] or when there was no answer until the cut-off date of this report [36, 38, 42]. The full data collected is listed in Appendix A.

It seems that issues are typically answered more quickly than PRs. Intuitively, this would make sense as PRs involve code reviews and more technical topics that might be more time consuming. But both, a logrank test [61] and a Wilcoxon signed-rank test [62] yield a p-value > 0.4, making the difference (at least with the data presented) statistically highly *insignificant*. Meaning that the difference in response times between issues and PRs cannot be shown.

Hence from now on it is assumed that the response time does not differ regardless where communication happens. The mean response time is about 7.5 days which seems too high to keep a discussion alive and relevant. The median on the other hand is 45 hours, which seems reasonable to receive an answer within two days. This shows that in most cases, messages are responded to in a reasonable time frame (especially by those aforementioned maintainers) [22, 34, 35, 39, 50, 51], but in other cases, it takes unjustifiably long [36, 38, 39, 42, 45]. Those outliers render communication often difficult and Figure 1 shows that in PRs there are many instances where responses take very long.

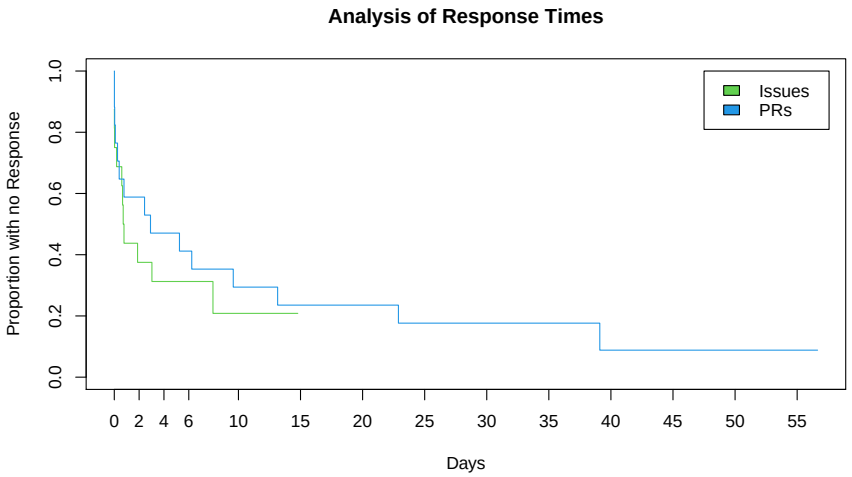

Figure 1: This graph shows the survival function of messages and their answers [60], differentiating between response times in issues and PRs. With this interpretation, a death occurs once a message has been answered (hence survival is not desired). For a given day on the x-axis, the proportion of messages that have not yet been responded to are shown. It considers that currently, data is censored, meaning that the date where all messages have been answered ($y = 0.0$) is unknown.

6.1. Length of my Messages

Since the data used for this analysis only includes communications with me, it is biased. After investigating the data further, it can be seen that my messages are longer compared to maintainers' responses (as seen in Figure 2). Now, one (myself included) might be wondering if my messages are too long, too detailed, or too complicated and if writing shorter messages would lead to lower response times overall. Generally, concise messages are preferred in FOSS development [32] but often a lot of context is needed, making messages longer.

Calculating the Pearson correlation coefficient [63] between the length of my messages and the answer rate, yields $r = 0.061$. This strongly suggests that there is no correlation between the length of messages sent by me and how quickly an answer was received. An accompanying scatterplot illustrated in the Appendix under Figure 3, shows also no obvious direct relationship. This lets me conclude that if I am the reason for long response times, it has nothing to do with the length of my messages but is more fundamental.

7. Openness

Both terms free and open source refer to properties of the software but are not concerned with how the product is developed [4, 64]. OpenStack lists terms that allow FOSS to be better classified [65]. In addition to the widespread term open source, three others are defined: open design, open development, and open community. The meaning of those terms and whether Code - OSS meets those are discussed in the following based on my experiences.

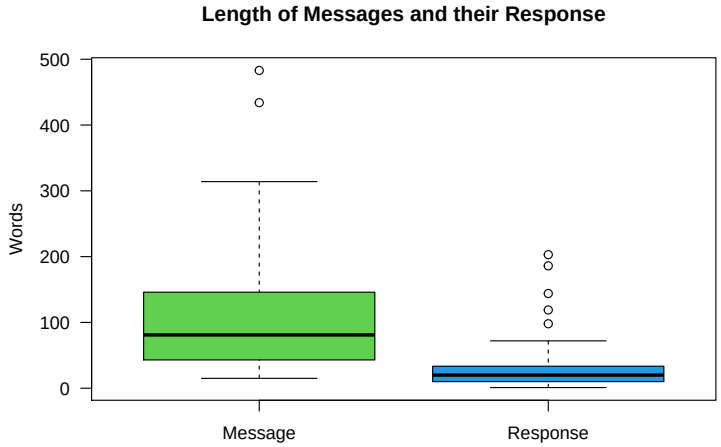

Figure 2: A boxplot comparing the number of words in my messages, and the length of maintainers' responses. It can be seen that my messages are almost always longer than those received by team members.

Open Design means that the design process is open for every member of the community. The aim is to produce a roadmap or an iteration plan together and open to the community. In Code - OSS, the community can vote on feature requests by upvoting issues on GitHub [66]; however, an individual opinion is often not cared about, for example see [20]. The roadmap and iteration plan are publicly available and updated regularly [9, 10]. Meaning that, while the community is not able to drastically alter the direction the maintainers take nor the path that lead to decision are made public, the results and decisions are.

Open Development describes the state in which all development happens in public and in which code reviews as well as the project's goals are known to the community. As this project is hosted on GitHub, all development happens there, and the direction is publicly announced, Code - OSS meets all requirements to qualify for open development.

Open Community is achieved once all processes are transparent, open, and every member of the community can interact with and influence the process at any point. In its complete form, this is only achieved when: 1) all communications and meetings are public, 2) the design and implementation process is governed by the community, and 3) the community can decide what people operate in which (lead) roles. This is where Code - OSS lacks the most much of communication happens internally, and most aspects are decided without involving the community. The README already sets the tone that this repository is where Microsoft develops VSCode, not the community [67].

7.1. Transparency and Community Involvement

While Code - OSS qualifies as free software, it is not considered open regarding the design and development. Thus, it is often difficult for external contributors to understand why some changes occur or other aspects are prioritized. One step the team could take is to make meetings and IRCs public. Although this might sound difficult for a company to achieve, other companies such as the Open Stack Foundation or Zulip

improve openness towards the community with this approach while still being profitable [65, 68]. More drastic approaches, such as going as far as letting the community decide on responsibilities and roles of maintainers can stand in conflict with a company-oriented business. As Code - OSS is developed by Microsoft, this could potentially cause a conflict since solely Microsoft pays for the development and salaries. Nevertheless, even by publicly announcing and giving reason for each hire (or making the hiring process open) would be a large step towards the right direction.

This lack of open community makes the project seem non-transparent in some cases. Decisions, especially those regarding the roadmap, seem like business decisions with only little interaction with the community. This lack of transparency has an impact on multiple parts. Those vary from not knowing why there is no response for a long time as in [36, 38, 39, 57, 58] or as to why a feature is not pursued (anymore) [20, 34]. In the spirit of openness, transparency, and engagement with the community, I tried to discuss with the team whether there is a system in place that allows to involve other team members if the current discussion partner does not seem to be properly involved in the discussion [36]. One important question was how to handle cases in which the partner does not answer and how long to wait for a ping or before involving someone else. The answer was along the lines that there is no guideline on those topics and stating reasons as to why a maintainer may not be reachable. The answer seems political and does not seem to have any profound intention to change anything regarding transparency.

This lack of transparency and openness towards general discussions can be found in multiple areas. For example, in December a contributor survey was conducted. It was announced that the results will be published. Those results were not released and an inquiry by me about the results was ignored [38]. Another example is an open issue from over a year ago regarding better inclusivity of people with disabilities with no intention to being solved, and no response from a maintainer for more than a year [58]. Or not reacting to discussions on how the team tries to address issues with accessibility in general [57]. In those and many other instances, transparently stating the current situation would be a good way to address them properly.

8. Retrospective

In the development of VSCode (or better said Code - OSS), many things concerning openness are definitely done right, especially since the size of the community is so large [8]. While contributing to Code - OSS, it has become apparent that the development is steered by the team and pursues the vision of Microsoft. Most maintainers highly appreciate external contributors and are generally happy to help community members. But this drive for open source varies between maintainers, where some are more willing or capable of investing time than others.

When revisiting the terms and definitions of openness, it is worth highlighting that the team fully adheres to open development, which is great. The publishing of the roadmap and iteration plan helps the community to stay up to date and helps the community to understand the current focus of the team [9, 10]. As the project has over 1300 external contributors, the community is definitely thriving. But a large portion of those collaborators may be attributed to the size and wide use of VSCode. As already discussed in the preliminary report, most contributors leave the project within a year [8].

8.1. What and how to improve

While the community is steadily growing, there are still changes that could foster the FOSS design and development principles and help getting the community more involved. Although some points have already been mentioned, the following summarizes some of the findings and build upon those. It lists some concrete ways on how the project could become more transparent, more open, more involving, or simply to improve how quickly newcomers find their way into the project in the near future.

More Labeling First of all, the number of issues marked with "help wanted" and especially those with "good first issue" should be drastically increased. The team annually schedules issue grooming iterations where they dedicate a complete iteration to cleaning up and taking a look at the backlog [69]. Apart from correctly labeling new issues, those iterations could be used to label older issues and nice-to-haves and make them available for the community. This requires awareness and dedication from the team. For issues with those labels, directly adding code pointers would require even more effort but may be similarly beneficial.

Improved Labels The difficulty of issues varies significantly, and it can become very tedious to asses the workload, especially for inexperienced contributors. Due to the long response times discussed earlier, reaching out to team members is not a satisfying option. Thus, I would suggest adding three labels of difficulties for "help wanted" issues such as easy, medium, and hard. While these additional labels might be very subjective and require (at least in the beginning) more time from maintainers, I believe that this would help newcomers a lot.

Public IRC The topic transparency and community involvement is crucial where a lot can be improved. The first step that I would suggest is to make the internal IRCs public. This not only allows the community to make more sense of the direction the team is taking but would also make it easier to see which maintainers are currently available. It would also give a better overview of the (available) maintainers and their responsibilities. Additionally, a public map containing maintainers and their responsibilities would be useful when a community member has a question, but does not know whom to contact.

Code Architecture Although in my opinion the code is structured and written rather well, a general high level overview of the structure and architecture could be useful for newcomers. As of now, I could not find any architecture overview. Hence, I would suggest a more advanced documentation structure such as the C4 model, where the first three levels would probably be sufficient in the beginning [70].

8.2. Reflection of my Contributions

While contributing, I tried to follow the FOSS development best practices as well as possible [32, 41]. Amongst others, those include to never develop privately, announce and ask before working on an issue, try to be open-minded and friendly, and to reply quickly. Most importantly, I tried to be concise while still giving necessary context. In subsection 6.1 it was discussed that my messages are longer than their responses, and although the length did not correlate with the answer time, this may be a general indicator that my messages might not be concise enough.

As the response times were rather long, in many cases I had to ping the threads multiple times. Some FOSS guidelines suggest to ping a thread after one week with no response [32]. In the beginning, I continuously pinged a question after every week, up to four times [42]. I could not find guidelines on how to handle such cases where there is no response after multiple pings, and an inquiry to the maintainer team did not suggest a procedure either [36]. While I did increase the time between pings, in hindsight it would be probably better to increase it sooner and only ping once every 1–2 months after 2–3 pings.

When contributing, my starting point was to search for issues in the backlog. In the beginning, I started with a good first issue and then investigated help wanted issues and continued from there. I believe that this is a common and solid approach. But in this specific instance with so much time at my disposal, it might have been a good idea to directly message a maintainer and ask whether I could get assigned issues. Those issues might would have been relevant for this iteration, or at least soon. This could have reduced the response time and review time in PRs, as the changes are desired by the team. Since there is no list of team members with their technical focus available, it would have been difficult to find the right person to contact. And even if this were public information, contacting a maintainer in a transparent way is difficult as there is no direct communication channel available.

Apart from those aspects, I believe that my contributions followed the general guidelines very closely, and hence most things went rather well [34, 36, 39, 42, 51]. I believe that my changes and the time I have invested were all appreciated, and I was able to contribute relevant and import changes [34, 35, 39, 42, 45, 51] (once they are merged). I would have loved to get continuous feedback on my code and discuss various aspects of implementations, but unfortunately, in most cases, this was not possible [31, 40, 42]. Because the community is generally very friendly and supportive, I plan to continue contributing to VSCode, and especially give my best that my remaining open PRs will be merged [42, 45, 51].

9. Conclusion

While VSCode, or better said Code - OSS, adheres rather well to FOSS design principles, there are still some areas where there is potential for improvement. Both the development process and design process are considered as very open [65]. All development happens in public and a lot of the short-term (i.e. iteration plans) and long-term visions (i.e. roadmaps) are made available to the community [9, 10]. Even if it is does not seem to be in Microsoft's interest, much has to be done to make VSCode an open community project [67]. Considering those aspects and the lacking commitment of making VSCode open source and free (including the extension store) shows that, while allegedly "Microsoft loves open source" [71], they are not fully embracing the community and their wishes. Microsoft is still a business, not your friend. But despite those facts, VSCode has achieved a helpful, open, friendly, and thriving community with many people committed to contributing to this project—so much that I believe that even without Microsoft's support, the community would be capable of forking the project and continue working on it in a meaningful way.

References

[1] The VS Code Team. *The next Phase of the Go Experience*. Visual Studio Code. June 9, 2020. URL: https://code.visualstudio.com/blogs/2020/06/09/go-extension (visited on Mar. 1, 2021).

[2] Stack Overflow Team. *Stack Overflow Developer Survey 2019*. Stack Overflow. 2019. URL: https://insights.stackoverflow.com/survey/2019#development-environments-and-tools (visited on Mar. 1, 2021).

[3] Free Software Foundation. *Various Licenses and Comments about Them*. GNU Operating System. Jan. 31, 2021. URL: https://www.gnu.org/licenses/license-list.en.html#SoftwareLicenses (visited on Mar. 12, 2021).

[4] Free Software Foundation. *What is free software?* Feb. 3, 2021. URL: https://www.gnu.org/philosophy/free-sw.en.html (visited on Mar. 5, 2021).

[5] Microsoft. *License - Visual Studio Code*. VSCode. URL: https://code.visualstudio.com/license (visited on Mar. 5, 2021).

[6] Opensource.org. *Licenses & Standards*. Open Source Initiative. URL: https://opensource.org/licenses/ (visited on Mar. 12, 2021).

[7] VSCodium. *Vscodium*. VSCodium, Mar. 11, 2021. URL: https://github.com/VSCodium/vscodium (visited on Mar. 12, 2021).

[8] Michael Plainer. *Study of Visual Studio Code*. Dec. 1, 2020. URL: https://www21.in.tum.de/teaching/osp/WS20/assets/pr-plainer-vscode.pdf (visited on Mar. 1, 2021).

[9] Microsoft. *Roadmap*. VSCode Wiki. Jan. 13, 2021. URL: https://github.com/microsoft/vscode/wiki/Roadmap (visited on Mar. 6, 2021).

[10] Microsoft. *Iteration Plans*. VSCode Wiki. Feb. 8, 2021. URL: https://github.com/microsoft/vscode/wiki/Iteration-Plans (visited on Mar. 6, 2021).

[11] Microsoft. *How to Contribute*. VSCode Wiki. Feb. 24, 2021. URL: https://github.com/microsoft/vscode/wiki/How-to-Contribute (visited on Mar. 11, 2021).

[12] GitHub. *Searching issues and pull requests*. GitHub Docs. Dec. 7, 2020. URL: https://docs.github.com/en/github/searching-for-information-on-github/searching-issues-and-pull-requests (visited on Mar. 8, 2021).

[13] Microsoft. *Issues*. GitHub. Mar. 6, 2021. URL: https://github.com/microsoft/vscode/issues (visited on Mar. 6, 2021).

[14] Google. *Refine web searches*. Google Search Help. URL: https://support.google.com/websearch/answer/2466433?hl=en (visited on Mar. 6, 2021).

[15] Kai Matzel. *2020 Issue Grooming - Review, Categorization, Assignment*. GitHub. Oct. 10, 2020. URL: https://github.com/microsoft/vscode/issues/108472 (visited on Mar. 19, 2020).

[16] Microsoft. *Milestone: On Deck*. GitHub. Mar. 6, 2021. URL: https://github.com/microsoft/vscode/milestone/27 (visited on Mar. 6, 2021).

[17] @hsdk123. *Allow to change the font size and font of the workbench*. microsoft/vscode. Nov. 24, 2015. URL: https://github.com/microsoft/vscode/issues/519 (visited on Mar. 6, 2021).

[18] Microsoft. *Milestone: Backlog*. GitHub. Mar. 6, 2021. URL: https://github.com/microsoft/vscode/milestone/9 (visited on Mar. 6, 2021).

[19] Christian Marx. *Code detects repositories when used as a difftool in Git Bash*. microsoft/vscode. July 31, 2018. URL: https://github.com/microsoft/vscode/issues/55428 (visited on Mar. 6, 2021).

[20] Adam Byrd. *No feedback when choosing "Check for Updates..."* microsoft/vscode. June 12, 2020. URL: https://github.com/microsoft/vscode/issues/100016 (visited on Mar. 6, 2021).

[21] @derkallevombau. *Commit message text box: Suggest file name if just one file has been staged*. microsoft/vscode. May 13, 2020. URL: https://github.com/microsoft/vscode/issues/97732 (visited on Mar. 6, 2021).

[22] @kpge. *Do not update disabled extensions*. microsoft/vscode. Aug. 3, 2019. URL: https://github.com/microsoft/vscode/issues/80244 (visited on Mar. 6, 2021).

[23] Allan Oricil. *[1210/111511.666:ERROR:registration_protocol_win.cc(103)] CreateFile: The system cannot find the file specified. (0x2)*. microsoft/vscode. Dec. 10, 2020. URL: https://github.com/microsoft/vscode/issues/112216 (visited on Mar. 6, 2021).

[24] Johannes Rieken. *Install prompt after "restart to update"*. microsoft/vscode. Feb. 8, 2021. URL: https://github.com/microsoft/vscode/issues/116086 (visited on Mar. 6, 2021).

[25] GitHub. *Building Welcoming Communities*. Open Source Guides. Dec. 16, 2020. URL: https://opensource.guide/building-community/ (visited on Mar. 6, 2021).

[26] Stephan Kulla. *Motivation in Open Source Projects*. personal communication. Dec. 8, 2020. URL: http://kulla.me/en/ (visited on Mar. 8, 2021).

[27] @laggingreflex. *External terminal not logging Electron logs*. microsoft/vscode. Oct. 23, 2020. URL: https://github.com/microsoft/vscode/issues/109216 (visited on Mar. 8, 2021).

[28] Edward Brey. *External terminal not logging Electron logs*. microsoft/vscode. May 17, 2019. URL: https://github.com/microsoft/vscode/issues/73879 (visited on Mar. 8, 2021).

[29] GitHub. *Using search to filter issues and pull requests*. GitHub Docs. Jan. 15, 2021. URL: https://docs.github.com/en/github/managing-your-work-on-github/using-search-to-filter-issues-and-pull-requests (visited on Mar. 8, 2021).

[30] Alexander Riccio. *vscode-typescript temp files still not deleted properly*. microsoft/vscode. Mar. 4, 2021. URL: https://github.com/microsoft/vscode/issues/118150 (visited on Mar. 14, 2021).

[31] Daniel Imms. *Include terminal name in terminal textarea title*. microsoft/vscode. June 2, 2020. URL: https://github.com/microsoft/vscode/issues/99072 (visited on Mar. 5, 2021).

[32] GitHub. *How to Contribute to Open Source*. Open Source Guides. Feb. 19, 2020. URL: https://opensource.guide/how-to-contribute/ (visited on Mar. 8, 2021).

[33] Michael Plainer. *Issues*. GitHub. Mar. 8, 2021. URL: https://github.com/microsoft/vscode/issues?q=involves%3Aplainerman (visited on Mar. 8, 2021).

[34] @xcetos. *Customize the location of ".vscode" directory*. microsoft/vscode. Feb. 21, 2021. URL: https://github.com/microsoft/vscode/issues/117161 (visited on Mar. 5, 2021).

[35] Craig E. Shea. *Provide Setting to NOT Track the Active Item in the File Explorer*. microsoft/vscode. Feb. 16, 2021. URL: https://github.com/microsoft/vscode/issues/116787 (visited on Mar. 5, 2021).

[36] Michael Plainer. *Community Question regarding Issue Handling*. microsoft/vscode. Feb. 15, 2021. URL: https://github.com/microsoft/vscode/issues/116726 (visited on Mar. 5, 2021).

[37] Microsoft Developer, director. *Ask the Team: Visual Studio Code — COM01*. May 25, 2020. URL: https://youtu.be/VRiaDG--A4k?t=159 (visited on Mar. 8, 2021).

[38] Michael Plainer. *Results from Contributor Survey*. microsoft/vscode. Mar. 3, 2021. URL: https://github.com/microsoft/vscode/issues/118056 (visited on Mar. 5, 2021).

[39] Michael Plainer. *Fix setting title of renamed terminal windows*. microsoft/vscode-wiki. Dec. 11, 2020. URL: https://github.com/microsoft/vscode/pull/112317 (visited on Mar. 5, 2021).

[40] @kzhui125. *git sync issue on an empty repo*. microsoft/vscode. June 22, 2020. URL: https://github.com/microsoft/vscode/issues/100725 (visited on Mar. 5, 2021).

[41] GitHub. *Best Practices for Maintainers*. Open Source Guides. Jan. 13, 2020. URL: https://opensource.guide/best-practices/ (visited on Mar. 8, 2021).

[42] Michael Plainer. *Fix 100725: Allow empty repositories to be synced*. microsoft/vscode-wiki. Jan. 11, 2021. URL: https://github.com/microsoft/vscode/pull/114163 (visited on Mar. 10, 2021).

[43] Eric Amodio. *Finding branch upstream via rev-parse or config?* git-for-windows/git. Jan. 14, 2021. URL: https://github.com/git-for-windows/git/discussions/2981 (visited on Mar. 8, 2021).

[44] Nestor Todorov. *Do not update disabled extensions*. microsoft/vscode. July 8, 2019. URL: https://github.com/microsoft/vscode/issues/76879 (visited on Mar. 5, 2021).

[45] Michael Plainer. *Add support to deactivate auto-updating disabled extensions*. microsoft/vscode-wiki. Dec. 19, 2020. URL: https://github.com/microsoft/vscode/pull/113155 (visited on Mar. 5, 2021).

[46] Carolyn Heinze and Cameron McKenzie. *Why GitHub renamed its master branch to main*. The Server Side. Nov. 24, 2020. URL: https://www.theserverside.com/feature/Why-GitHub-renamed-its-master-branch-to-main (visited on Mar. 8, 2021).

[47] GitHub. *The default branch for newly-created repositories is now main - GitHub Changelog*. The GitHub Blog. Oct. 1, 2020. URL: https://github.blog/changelog/2020-10-01-the-default-branch-for-newly-created-repositories-is-now-main/ (visited on Mar. 8, 2021).

[48] João Moreno. *Scheduled rename 'master' branch to 'main'*. microsoft/vscode. Feb. 10, 2021. URL: https://github.com/microsoft/vscode/issues/116341 (visited on Mar. 8, 2021).

[49] Michael Plainer. *Rename master to main in Coding-Guidelines and How-to-Contribute*. microsoft/vscode-wiki. Feb. 15, 2021. URL: https://github.com/microsoft/vscode-wiki/pull/167 (visited on Mar. 5, 2021).

[50] @nihiluis. *Don't delete files when replacing dir*. microsoft/vscode. Aug. 30, 2020. URL: https://github.com/microsoft/vscode/issues/105691 (visited on Mar. 5, 2021).

[51] Michael Plainer. *Recursively expand folders when moving*. microsoft/vscode-wiki. Jan. 26, 2021. URL: https://github.com/microsoft/vscode/pull/115054 (visited on Mar. 5, 2021).

[52] International Labour Organization. *An employers' guide on working from home in response to the outbreak of COVID-19*. Geneva: ILO, 2020. ISBN: 978-92-2-032253-6. URL: https://www.ilo.org/wcmsp5/groups/public/---ed_dialogue/---act_emp/documents/publication/wcms_745024.pdf (visited on Mar. 10, 2021).

[53] Brett Dunst. *Open Source Software: A Model For Transparent Organizational Communication*. Nov. 22, 2019. URL: https://www.forbes.com/sites/forbescommunicationscouncil/2019/11/22/open-source-software-a-model-for-transparent-organizational-communication/ (visited on Mar. 19, 2021).

[54] The Open Organization Ambassadors at Opensource.com. *The Open Organization Definition*. Aug. 28, 2020. URL: https://opensource.com/open-organization/resources/open-org-definition (visited on Mar. 19, 2021).

[55] Microsoft. *Feedback Channels*. VSCode Wiki. Oct. 23, 2020. URL: https://github.com/microsoft/vscode/wiki/Feedback-Channels (visited on Mar. 11, 2021).

[56] Louis Reingold. *Be Brave and Remove Cut/Copy/Paste From The Right-Click Menu*. microsoft/vscode. Mar. 5, 2021. URL: https://github.com/microsoft/vscode/issues/118172 (visited on Mar. 10, 2021).

[57] Patrick Kelly. *Accessibility concerns; I can't reasonably use VS Code anymore*. microsoft/vscode. Jan. 20, 2020. URL: https://github.com/microsoft/vscode/issues/88987 (visited on Mar. 10, 2021).

[58] Patrick Kelly. *Better inclusivity*. microsoft/vscode. Dec. 22, 2019. URL: https://github.com/microsoft/vscode/issues/87570 (visited on Mar. 10, 2021).

[59] Per Kragh Andersen et al. "The Mathematical Background". In: *Statistical Models Based on Counting Processes*. Series Title: Springer Series in Statistics. New York, NY: Springer US, 1993, pp. 45–120. ISBN: 978-0-387-94519-4 978-1-4612-4348-9. DOI: 10.1007/978-1-4612-4348-9_2. URL: http://link.springer.com/10.1007/978-1-4612-4348-9_2 (visited on Mar. 10, 2021).

[60] Manish Kumar Goel, Pardeep Khanna, and Jugal Kishore. "Understanding survival analysis: Kaplan-Meier estimate". In: *International journal of Ayurveda research* 1.4 (Oct. 2010), pp. 274–278. ISSN: 0974-7788. DOI: 10.4103/0974-7788.76794. URL: https://europepmc.org/articles/PMC3059453.

[61] R Peto et al. "Design and analysis of randomized clinical trials requiring prolonged observation of each patient. II. Analysis and examples". In: *British Journal of Cancer* 35.1 (Jan. 1977), pp. 1–39. ISSN: 0007-0920, 1532-1827. DOI: 10.1038/bjc.1977.1. URL: http://www.nature.com/articles/bjc19771 (visited on Mar. 10, 2021).

[62] Frank Wilcoxon. "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* 1.6 (Dec. 1945), p. 80. ISSN: 00994987. DOI: 10.2307/3001968. URL: https://www.jstor.org/stable/10.2307/3001968?origin=crossref (visited on Mar. 10, 2021).

[63] Kristin Yeager. *SPSS Tutorials: Pearson Correlation*. Kent State University. Mar. 9, 2021. URL: https://libguides.library.kent.edu/SPSS/PearsonCorr (visited on Mar. 11, 2021).

[64] Opensource.org. *The Open Source Definition (Annotated)*. Open Source Initiative. Mar. 22, 2007. URL: https://opensource.org/osd-annotated (visited on Mar. 11, 2021).

[65] OpenStackFoundation. *What Does Openness Mean?* OpenStack. URL: https://wiki.openstack.org/wiki/Open (visited on Mar. 11, 2021).

[66] Microsoft. *Milestone: Backlog Candidates*. GitHub. Mar. 11, 2021. URL: https://github.com/microsoft/vscode/milestone/107 (visited on Mar. 11, 2021).

[67] Microsoft. *README.,d*. GitHub. Feb. 22, 2021. URL: https://github.com/microsoft/vscode/blob/1c9e7eb296eacfb5939f3e9268ff2836e7710e62/README.md (visited on Mar. 12, 2021).

[68] K.S. Vishnu. *How to build a profitable business around your open-source project*. personal communication. Feb. 2, 2021. URL: https://vishnuks.com/ (visited on Mar. 11, 2021).

[69] Microsoft. *Issue Grooming*. VSCode Wiki. Oct. 14, 2019. URL: https://github.com/microsoft/vscode/wiki/Issue-Grooming (visited on Mar. 13, 2021).

[70] Simon Brown. *The C4 model for visualising software architecture*. C4 Model. URL: https://c4model.com/ (visited on Mar. 13, 2021).

[71] René Moddejongen. *Microsoft loves open source*. URL: https://pulse.microsoft.com/nl-nl/transform-nl-nl/na/fa1-microsoft-loves-open-source/ (visited on Mar. 14, 2021).

Appendices

A. Communication Records

In the following table all communications initiated by me to which a response was expected are listed. Messages where no answer was required (e.g. stating what I was currently doing) are omitted from this table. The 12.03.2021, 12:00 was chosen as cut-off date for this study, meaning that everything past this date is not listed.

Column	Meaning
Type	Whether this communication was inside an issue or a PR. PRW means a PR in the wiki.
ID	The unique identifier for each issue or PR.
Message Date	The date and time the message was posed (in UTC, rounded to 15 minutes).
Response Date	When status is 1, date and time when answer was received (in UTC). Otherwise, a cut-off date was entered which is either the end of study (12.03.2021, 12:00) or when a response would have no meaning anymore.
Time	The time difference in hours between Message Date and Response Date.
Status	1 means received answer, 0 means no answer received (i.e. censored).
Words	The number of words in the request.
Words Response	The number of words in the response. x if censored.
Satisfactory	Whether the response was satisfactory 1 (e.g. partner seems to have read the question) or 0 otherwise. x if censored.
Pinged	How many times the message was pinged additionally. 0 means no ping was required.

Type	ID	Message Date	Response Date	Time	Status	Words	Words Response	Satisfactory	Pinged
Issue	#100016	06.12.2020, 12:45	08.12.2020, 09:45	45.00	1	88	11	1	0
	#99072	06.12.2020, 18:00	11.12.2020, 18:45	120.75	0	81	x	x	0
	#76879	12.12.2020, 17:45	13.12.2020, 08:15	14.50	1	50	20	1	0
	#97732	13.12.2020, 15:45	14.12.2020, 10:30	18.75	1	118	24	1	0
	#76879	13.12.2020, 16:00	14.12.2020, 08:00	16.00	1	17	19	1	0
	#100725	13.12.2020, 17:15	14.12.2020, 10:30	17.25	1	163	15	0	0
	#76879	14.12.2020, 08:45	19.12.2020, 14:30	125.75	0	49	x	x	0
	#100725	14.12.2020, 11:00	17.12.2020, 11:45	72.75	1	131	12	0	0
	#105691	25.01.2021, 15:45	25.01.2021, 16:15	0.50	1	115	72	1	0
	#105691	26.01.2021, 14:30	26.01.2021, 14:30	0.00	1	18	6	1	0
	#80244	27.01.2021, 08:45	27.01.2021, 08:45	0.00	1	15	11	1	0
	#116726	15.02.2021, 20:15	23.02.2021, 19:00	190.75	1	314	186	0	0
	#116787	16.02.2021, 17:00	16.02.2021, 17:15	0.25	1	17	6	1	0
	#117161	22.02.2021, 08:00	22.02.2021, 12:30	4.50	1	21	25	1	0
	#116726	25.02.2021, 17:15	12.03.2021, 12:00	354.75	0	483	x	x	0
	#118056	03.03.2021, 17:45	12.03.2021, 12:00	210.25	0	57	x	x	0
PR	#112317	11.12.2020, 18:30	19.01.2021, 16:15	933.75	0	120	x	x	2
	#113155	19.12.2020, 14:15	27.01.2021, 16:45	938.50	1	146	23	1	3
	#112317	22.12.2020, 11:00	04.01.2021, 14:45	315.75	1	47	25	1	0
	#114163	11.01.2021, 18:15	14.01.2021, 04:45	58.50	1	434	33	1	0
	#112317	14.01.2021, 08:45	14.01.2021, 09:00	0.25	1	43	10	1	0
	#114163	14.01.2021, 12:30	14.01.2021, 18:45	6.25	1	31	10	1	0
	#114163	14.01.2021, 20:30	12.03.2021, 12:00	1359.50	0	62	x	x	4
	#112317	19.01.2021, 14:15	19.01.2021, 16:15	2.00	1	22	2	1	0
	#115054	26.01.2021, 14:30	26.01.2021, 15:15	0.75	1	248	119	1	0
	#115054	27.01.2021, 09:15	02.02.2021, 15:00	149.75	1	198	98	1	0
	#115054	02.02.2021, 18:00	03.02.2021, 12:45	18.75	1	132	27	1	0
	#115054	04.02.2021, 09:15	09.02.2021, 15:15	126.00	1	83	144	1	0
	#113155	08.02.2021, 11:30	03.03.2021, 08:45	549.25	1	42	7	1	1
	#115054	09.02.2021, 16:00	09.02.2021, 16:15	0.25	1	194	34	1	0
	#115054	09.02.2021, 19:45	12.02.2021, 17:45	70.00	1	79	203	1	0
	#115054	12.02.2021, 18:45	22.02.2021, 08:45	230.00	1	247	10	0	1
PRW	#167	15.02.2021, 07:15	15.02.2021, 16:45	9.50	1	63	1	1	0

B. Message Length Scatterplot

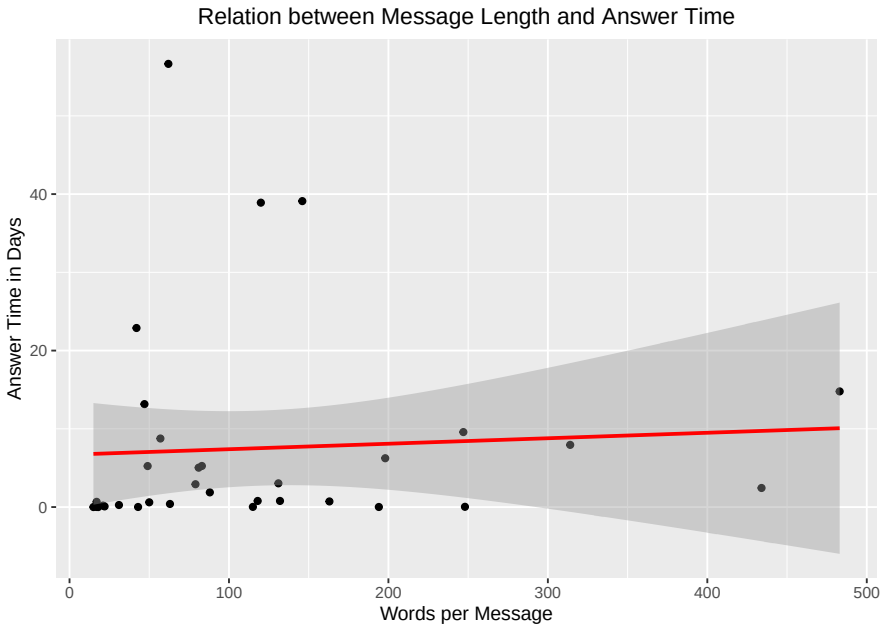

Figure 3: A scatterplot showing the relation between length of messages sent by me and the corresponding answer time in days. The red line is a fit linear regression with the accompanied confidence interval of the means. No clear relationship is visible, neither a linear nor a non-linear one.