1 Documentation

Writing documentation is a part of project development that every developer dislikes. No matter how large or difficult the project might be, documentation always ends up being forgotten. And even though many tools have been introduced to make writing documentation much easier, a lot of progress is still to be made.[7][8]

For the creation of a new pulsar detection library, the importance of documentation should not be underestimated. To illustrate this issue, most of the existing pulsar detection libraries contain barely any documentation, which makes it hard to understand the purpose of the library. Especially for an advanced topic like detecting pulsars using radio signals, each feature of the library should have a description in exact detail what it is about. And on top of that, the development of the library is open source, and thus requires the help of programmers who might be unfamiliar with pulsar detection, signal processing or astronomy. That is why the existence of documentation for a new pulsar detection library is a necessity.

Previous research has indicated that the most common problems for documentation are inadequate documentation, poor writing style and resistance to document.[2]

1.1 Methods

To make sure that the documentation does conform with high standards, the relevant attributes for writing and publishing documentation should be defined. With the help of several case studies and interviews, these attributes can be described as follows.

1.1.1 Completeness, consistency and credibility

The first attributes for measuring the quality of the documentation are the completeness, the consistency and the credibility. These three attributes are important because they each describe different aspects of the documentation, but have are closely related.

For example, the completeness measures the amount of modules and functions that are included in the documentation. Even though, the completeness of the documentation is very important, it becomes useless whenever the documentation is either not consistent or credible. That is why not only the completeness, but also the consistency and the credibility of the documentation are relevant.[1]

The consistency of the documentation describes whether the documentation for each module is written in the same format, and whether there are any conflicts in the information that is given. Writing the documentation in the same format, will also improve the findability and readability of the documentation. On top of that, removing conflicting information will avoid confusing readers, who are unfamiliar with the use of the library.[1]

And last, the credibility of the documentation. While the completeness and consistency are much easier to measure, the credibility should not be underestimated. The credibility gives an indiciation about how accurate the information in the documentation is. When the credibility of your documentation is low, chances are that people will not use your library. Thus the credibility of your documentation being just as, if not more than, important as the completeness and consistency of your documentation.[7]

1.1.2 Availability, findability and traceability

In addition to the completeness, consistency and credibility of the documentation, the availability, findability and traceability also play an important role. While the previous attributes were used to describe the actual information in the documentation, the new attributes could be used to measure the findability of the documentation.

For instance, the availability of the document is an indicator for which percentage of the time the document is available to view. If a document is never available to its users, the availability of the document will be low. The availability is highly necessary whenever developers rely on the availability of the documentation.[5]

Furthermore, the findability is measured by whether users are able to find the documentation, or parts of the documentation, when searching online. This could for example be when users try to look for the documentation when searching on Google (or another search engine). If users are unable to find your documentation within a short period of time, chances are their willingness to keep using your library will decrease or even decay.[1]

The traceability plays an important role in whether users are able to trace the documentation back to its original code. The importance of the traceability increases with the size of your application, because users should be able to know which part of the application is described in the documentation. This is mostly done by creating a format for the documentation that is similair to the actual application. For example, layers of the application are used as folder names for the documentation. As a result, users are able to trace back the documentation to the code using the structure of the application.[4][2]

1.1.3 Reusability and modifiability

The reusability is used to describe whether parts of the documentation can be easily adapted for creating new sections. This is important when having time constraints for writing the documentation. For example, reusing parts of the documentation can save the writer/maintainer a lot of time. And if used correctly it can also increase the modifiability of the documentation.[1]

The modifiability of the documentation is in indicator for whether changes to the structure or content of the documentation can be made without a lot of effort. And just like the reusability, the modifiability plays an important role in the time constraints of the maintainer of the documentation. That is why, if the maintainer wants to put the least amount of effort into maintaining the documentation, the reusability and modifiability of documentation should be high.[1]

1.1.4 Clarity, readability and understandability

And finally there is the clarity, readability and understandability of the documentation. These three attributes are, just like the completeness, consistency and credibility, related to the actual content of the documentation. However, where the previous three attributes played a more important role in the validity of the documentation, these three attributes are related to the ease of use.

The clarity of the documentation is an indicator for whether the structure of the documentation is simple and clear. Because even though the clarity of the documentation is not a complicated attribute, it should not be overlooked when deciding which format or platform to use when publishing the documentation.[3]

Furthermore, the readability describes how much effort it could take to read the documentation. The readability could increase when using a font type that is easy to read, using picture or figures only when necessary and highlighting keywords. Furthermore, a glossary could be provided when using terms that are only used in a specific research domain.[6][2][3]

Last of all, the understandability is used to tell whether the stakeholders of the documentation are able to understand the concepts that are being described in the documentation. If the stakeholders of your documentation are unable to understand the language used or concepts described in the documentation, changes should be made to your documentation so stakeholders are able to understand it again.[6][4]

1.2 Results

After defining all the different attributes that might play a role in the quality of the documentation, we can start testing whether our documentation does conform with these attributes, and whether any improvements should made to conform with the standards. Because even though a lot of work has already been put into improving the documentation, development is still ongoing and changes can stil be made.

1.2.1 Continuous integration and development

The completeness, consistency and credibility of our documentation is reviewed each time new functionality is added to the application. By performing pull requests whenever new code is pushed to GitHub, members of the development team can validate whether documentation is included. Furthermore, members of the development team can validate whether the included documentation is complete (every class and method is described in the documentation), consistent (the new documentation is written in the same format as previous documentation) and credible (is the information in the documentation accurate).

Further improvements to the completeness of the documentation could be made by putting an indicator of the documentation coverage on the page of the GitHub repository. This indicator could tell other developers which part of the documentation requires more attention, and thus receive the attention it needs. The consistency of the current documentation is already high. For example, all our documentation is written in Markdown and included in the repository, and new documentation is written in the same format as previous documentation. Although, the consistency of the format could improve whenever the documentation is automated. However, due to the requirements of this projects, the use of automated documentation is not allowed and thus impossible. In addition, the credibility of the documentation could improve whenever we request feedback from our supervisors. Though, due to time constraints it is unlikely that they will be able to validate all the documentation.

1.2.2 Distribution of repository

Secondly, there is the availability, findability and traceability of the documentation. The documentation is hosted together with the source code on GitHub. This means that whenever the source code goes offline, the documentation with go down as well. However, since it is very unlikely that GitHub will go offline any time soon, the documentation and source code will stay available for years to come. And even if GitHub goes offline, every developer who cloned the library from GitHub will have the documentation included. The findability of our GitHub repository has not received any attention so far. This attribute has not been given any attention so far, because the library is still in pre-alpha. Whenever, the library goes beta, the development team might put more effort into improving the findability of the repository. By structuring the documentation the same way our application is build up, we make it easier to trace the documentation back to the source code. Furthermore, our documentation includes the names of modules, classes and functions in which the source code is placed.

The availability of our library could be improved by distributing the repository on several platforms. For example, if we host our code not only on GitHub, but also on GitLab or SourceForge, we lower the chances of our repository being not available. Furthermore, we could distribute our documentation in PDF, so users can download the documentation. By adding keywords to the repository of our library, we improve the findability. So whenever users search for keywords related to pulsar detection on GitHub, they are able to find our library. On top of that, distributing our library on several platforms will also improve the findability of our repository. And last, by accompanying our documentation with links to the original source code, we are able to increase the traceability of our documentation. However, since it is impossible to accompany the documentation with links to the exact functions (only the classes), it might not be worth the effort.

1.2.3 Trade-off

Third, the reusability of our documentation is low. This is because the documentation is used to describe very specific functions. Making the documentation more generic could improve the reusability of our documentation, but will hurt the quality. However, currently we reuse certain Markdown structures whenever needed. For example, whenever we need the syntax for writing a block of code, we reuse parts of the existing documentation. Furthermore, the modifiability is currently high. It is high, because we use Markdown to write and edit our documentation. The only problem we encountered so far, is the amount of effort it requires to make small changes to the documentation on our 'Master'-branch. For instance, when we want to make a change, we need to perform the following steps. First we need to update the actual Markdown documentation. Then we push the changes to a new branch. After that, a pull request should be made to the master, to update the master. And after that, two other developers need to review the changes, and approve them. After that, the changes can finally be merged.

Even though, the reusability and modifiability of our documentation is not perfect, making changes to increase to these attributes will decrease other attributes. That is why no changes are proposed to improve these two attributes.

1.2.4 Scaleability

The last three attributes that require improvements are the clarity, readability and the understandability. Thanks to the use of Markdown for writing our documentation, the documentation remains clear and readable. By adding a table of content in the root of our repository, users do not have to search inside our folders to find the parts of the documentation they are looking for. Furthermore, links back to the table to content are added to each individual document, so users can navigate back to the root, whenever they finished reading. In addition, the documentation includes both descriptions and examples to improve the readability. The understandability of the documentation has been emphasized by looking the possible actors for our library, and writing in a language our actors can understand. For example, the manual for previous pulsar detection libraries made use of language that is well known by astronomers, but unknown by developers. That is why we payed extra attention to using a language that both astronomers and developers are able to understand.

When the library increases in size, the clarity of our documentation might need more attention. For example, subfolders could be created to organize folders in a logical structure. Furthermore, when the size of individual documents increases, a table of content for each individual document might be added. And, paragraphs and chapters might be reorganized to improve the overall structure. The same goes for the understandability of our documentation. Whenever, new actors might become interested in the use of our library, changes to documentation should be made to improve the overall understandability of the documentation.

1.3 Discussion

As a result of determining all the different attributes that play a role in writing documentation with high standards, and testing these standards on the current state of our library, we are able to conclude the following statements.

The consistency, modifiability, traceability and reusability of our documentation is in an excellent state. Even though changes could be made to these four attributes, it will hurt the overall quality of our documentation. That is why no changes will be made to increase the consistency, modifiability, traceability and reusability of our documentation.

However, the completeness, credibility, availability, findability could receive some improvements. The completeness could be improved by adding an indicator to the repository, that tells the developers how much functions are described in the documentation. Furthermore, the credibility of our documentation could be improved by asking expert (in our case the supervisors) to give us feedback about the credibility of our documentation. On top of that, by distributing our repository to multiple platforms, other than GitHub, the documentation and source code will increase in availability and findability.

And last, the remaining attributes, the clarity, readability and understandability, do not need any improvements at the moment, but might need more attention in the future. For instance, whenever more modules, classes and functions are added to the source, the documentation might need changes to the structure of the documentation. For example, subfolders could be added to indiciate smaller modules, more paragraphs and chapters could be added to indicate smaller functionality of larger functions and a table of content could be added to individual documents to make navigation easier.

In conclusion, the overall state of our documentation is above average. Currently some improvements could be made to improve the quality of our documentation. Last of all, the structure of the documentation might need more changes in the future, if the structure of the library is subject to larger changes.

References

[1] W Ding, P Liang, A Tang, and Hans van Vliet. Knowledge-based approaches in software documentation: A systematic literature review. 2014.

[2] G Garousi, V Garousi-Yusifoglu, G Ruhe, J Zhi, M Moussavi, and B Smith. Usage and usefulness of technical software documentation: An industrial case study. 2014.

[3] R Guillemette. Application software. 1987.

[4] A Jansen, P Avgeriou, and Jan Salvador van den Ven. Enriching software architecture documentation. 2009.

[5] L Nguyen-Hoan, S Flint, and R Sankaranarayana. A survey of scientific software development. 2010.

[6] S Voigt. A method for documenting agile software projects. 2017.

[7] S Voigt, Jorg von Garrel, Julia Muller, and Dominic Wirth. A study of documentation in agile software projects. 2016.

[8] R Watson, M Stamnes, J Jeannot-Schroeder, and J Spyridakis. Api documentation and software community values: A survey of open-source api documentation. 2013.