REPORT 8

The Future of Software Documentation Maintenance Processes

[ANONYMIZED REPORT – NO IDENTIFYING INFORMATION]

Abstract

This article describes software documentation maintenance processes based on comments on FAQ websites, online forums, such as, for example, Stack Overflow, and Issue Tracking Systems, such as, for example, GitHub, and seeks to understand their future relevance and possible future solutions to this problem. Tools such as the Problem Understanding Card and the Future Scenario Maker, from the Board of Innovations, were used, which help to better understand current problems and to speculate about possible futures, in this case, for software documentation. Among the proposed future scenarios, "DocuAI" stands out, a documentation generated by AI personalized for each developer.

1. Introduction

Software documentations are artifacts widely available and used nowadays, besides helping in the learning of new technologies, they can potentially be used as a learning object when addressing the concepts used in these technologies. Despite that, there are difficulties regarding the use of these documentations for the learning of frameworks or libraries and the concepts behind them, one of these difficulties is the lack of standardization in the creation of documentations and the maintenance of these, which makes developers turn to online forums or even end up abandoning the use of certain tools due to a deficient documentation.

This lack of standardization and maintenance indicates a precariousness in the processes involved in the creation and updating of software documentations, and, with this, the doubt arises: what is being said about this nowadays and where can we go in order to improve these processes?

This article has the following structure:

1. Introduction;

2. Foundation: Section in which the main concepts used in this article are presented;
3. Board of Innovation: Section in which the speculative design methodology used is made explicit and explained;
4. State of the Art: Section in which other articles from the state of the art within the theme are explored;
5. Possible Futures: Section in which possible futures are speculated, "where are we going?";
6. Desirable Future: Section in which the projection of the desirable future is discussed, "where do we want to go?";
7. Conclusion.

2. Foundation

Over the years, with the expansion of the software industry, countless APIs (Application Programming Interface) were developed that are used by developers in their day-to-day. These APIs allow developers to create more sophisticated software in a faster way and are considered as the most important factor for choosing a programming language [Thayer et al. 2021]. However, according to the authors, when developers decide to learn and use an API it is common for them to find a gap between what they want to do and what resources are available. For example: the official documentation often is incomplete, inadequate or difficult to navigate, relevant examples are difficult to find and adapt, among others.

[Cummaudo et al. 2019] state that good documentations facilitate the development process, improving its productivity and quality, however, they also point out that there are few studies regarding the necessary elements for the creation of good developer documentation. With this, the authors identified the need to perform a systematic mapping of the literature and, based on it, developed a taxonomy for the elaboration of developer documentation with quality. This taxonomy indicates 5 main aspects for a quality documentation, these being (i) the description of API usage, (ii) the design logic (in which contexts the use of the API is or is not appropriate) (iii) the domain concepts (what is necessary to know in order to be able to use the API) (iv) support artifacts for the use of the API, and; (v) the presentation of the documentation.

In addition, [Thayer et al. 2021] define as three main components to understand and use an API (i) the domain concepts, that is, the computing concepts behind the API, (ii) the API usage patterns, that is, how to coordinate the use of the API and its features within a project and (iii) the API execution facts, that is, how to predict results and side effects when executing the API.

3. Board of Innovations

To explore the problems and the future of the creation of software documentation, it was chosen to use two tools found on the Board of Innovations website [BOI 2024]. They are the Future Scenario Maker (Creator of Future Scenarios) and the Problem Understanding Card.

3.1. Problem Understanding Card

The Problem Understanding Card is a tool that uses artificial intelligence from the Board of Innovations [BOI 2024] that, upon receiving information from the user, generates relevant information for understanding the problem brought by the user himself.

To use the Problem Understanding Card, the user must fill in two fields: Target Audience, indicating who is suffering with the problem in question, and Problem, indicating in fact the problem that is sought to understand.

For this work, the fields were filled in according to Figure 1. As Target Audience it was filled in "Software Developers" and as Problem it was filled in "Poor Documentation".

Figure 1. Filling in the fields of the Problem Understanding Card

The results of this tool are made explicit in Figure 2. The obtained result details the problem better in point 1, where it speaks about lack of clear instructions that lead to confusion and inefficiency, incomplete or outdated documentation that delays productivity and the difficulty in understanding errors and problems without the correct documentation.

In addition, these points are reinforced with data in point 3, where it is stated that, according to the Stack Overflow Developer Survey of 2020, 90% of developers say they need documentation to write code and that Gartner Research states that poor documentation can increase development time by up to 50%. Point 2 is a quote created to help better understand the positioning of the Target Audience within the problem. Point 4 brings some Design principles to take into account when creating and maintaining software documentation, such as Clarity and Consistency, and point 5 brings initial solution directions, these being the implementation of a documentation review process with regular developer feedback, the introduction of interactive tutorials to supplement traditional documentation and the use of AI tools to auto-generate documentation based on changes in code.

Figure 2. Result of the Problem Understanding Card

3.2. Future Scenario Maker

The Future Scenario Maker is a tool that uses artificial intelligence from the Board of Innovations [BOI 2024] that, upon receiving information from the user, generates future scenarios based on a product or service aimed at a specific industry in a determined time.

To use the Future Scenario Maker, the user must fill in three fields: Product/Service, indicating which product or service is offered, Industry, indicating in which industry the product or service will be applied and Year, indicating in which year it is desired to project the scenarios.

For this work, the fields were filled in according to Figure 3. As Product/Service it was filled in “Software Documentation created using comments on Issue Tracing Systems”, as Industry it was filled in “IT” and as Year it was filled in “2040”.

Figure 3. Filling in the fields of the Problem Understanding Card

The results of this tool are made explicit in Figure 4. The first scenario brings the “DocuVerse”, an integration of software documentation with virtual reality. When putting on a virtual reality headset, the user would enter a world where the documentations and the comments about the code transform into an interactive 3D space where one can stroll, interacting with diagrams, tutorials, among others, leaving behind the traditional text-based documentation.

The second scenario brings the “DocuAI”, documentation generated by AI. It is an AI assistant that automatically generates a personalized documentation for each developer according to his needs made explicit through his comments in ITSs, besides promoting suggestions in real time, predicting possible bugs and offering solutions even before the developer finds it.

The third scenario brings the “DocuChain”, a decentralized documentation network. It brings an idea of using blockchain in ITS comments, guaranteeing integrity and transparency to documentations, making them resistant to data loss and unwanted modifications.

Figure 4. Result of the Problem Understanding Card

4. State of the Art

Three articles were identified that deal with the state of the art in software documentation from FAQ websites, online forums and comments in Issue Tracking Systems (ITS).

4.1. FAQs

[Ellmann and Timmann 2019] studied more than 2000 questions in more than 40 FAQ (Frequently Asked Questions) websites. The authors analyzed these questions regarding their accessibility metrics (e.g. step by step of the main documentation, markings), as well as their structures and readability. In addition, the analyzed questions were compared with more than 69000 Stack Overflow posts that covered the same topics and were posted more than once.

The results of [Ellmann and Timmann 2019] reveal that different software producers give different importance to their FAQs, either investing more or less effort in their structuring and presentation. The authors also point out that the answers in

FAQs usually include more references and are more difficult to read than their corresponding Stack Overflow posts, besides normally covering more additional topics when compared with their corresponding posts.

4.2. Online Forums

[Uddin et al. 2021a] analyzed that, currently, several researches study techniques for improving software documentation using information taken from online forums, specifically, in this case, from Stack Overflow. In addition, the authors identified that software developers consider the combination of code examples and their evaluations, found on Stack Overflow, more useful than the official documentation, since it can be incomplete, ambiguous, incorrect or outdated. However, the authors did not identify in the literature ways to generate documentation from Stack Overflow posts considering both the code examples and their evaluations.

From this, [Uddin et al. 2021a] present two algorithms for automatic production of software documentation from information taken from Stack Overflow, combining code examples and their evaluations. The first algorithm, named Statistical Documentation, shows the distribution of negativity and positivity in relation to the code examples of a certain software. The second algorithm, called Concept-based Documentation, joins similar and conceptually relevant usage scenarios. The usage scenarios include code examples, a textual description of the task addressed by the code examples and the evaluations made by other developers about these code examples.

4.3. Issue Tracking Systems

[Arya et al. 2019] present techniques for identifying types of information present in discussions present in ITSs. These information are present in several comments that accumulate in discussion threads about the projects present on the platforms. The authors performed qualitative analyses in the contents present in 15 complex threads distributed in three different projects on GitHub, from which 16 types of information were discovered.

5. Speculation of the Possible Futures

The first scenario discussed in Section 3.2 brings the "DocuVerse", an integration of software documentation with virtual reality. When putting on a virtual reality headset, the user would enter a world where the documentations and the comments about the code transform into an interactive 3D space where one can stroll, interacting with diagrams, tutorials, among others, leaving behind the traditional text-based documentation.

This scenario is not ideal, since, although there is indeed space for updating and maintenance of developer documentation, these updates and the artifacts present in documentations are not easily translatable to an interactive 3D space. The same idea could easily be applied in a virtual environment that is not in virtual reality, where diagrams, tutorials, among others, would be made available, as supplements to documentations.

The second scenario brings the "DocuAI", documentation generated by AI. It is an AI assistant that automatically generates a personalized documentation for each developer according to his needs made explicit through his comments in ITSs, besides promoting suggestions in real time, predicting possible bugs and offering solutions even before the developer finds it.

Of the proposed scenarios, this is the one with the greatest possibility of real implementation. The idea generated by the tool brings an AI assistant for the developer of the application that, when accessing the comments about his project, receives the help of this assistant with bugs, features, among others, automatically generating the necessary documentation to solve the comments posted in the ITS and, therefore, to update the current documentation.

The third scenario brings the "DocuChain", a decentralized documentation network. It brings an idea of using blockchain in ITS comments, guaranteeing integrity and transparency to documentations, making them resistant to data loss and unwanted modifications.

This scenario only added security to the information brought in the comments, removing a step of screening these for the documentation creator, which does not solve the problems brought in this article about software documentation.

6. Projection of the Chosen Desirable Future

Besides what was pointed out in the previous section about the “DocuAI” scenario, it is already possible to see today solutions aimed at documentation generation using Artificial Intelligence.

In [Filgueira and Garijo 2022] a code analysis framework was created that automatically extracts main features, metadata and documentation from Python code repositories from the input of a folder with the code. This tool uses AI techniques to perform its function.

Already in [Uddin et al. 2021b] two algorithms are presented to generate API usage scenarios from Stack Overflow comments. The first algorithm shows the distribution of positive and negative comments around a code example present in a comment, while the second joins usage scenarios presented in similar and conceptually relevant comments.

Finally, in [Berhouma 2021] a generic software documentation model is presented taking into account development in Scrum. The model generates guidelines so that developers can manage to structure well the creation of software documentation.

7. Conclusion

Software documentation plays a crucial role in the efficiency and quality of software development, but it continues to face significant challenges due to the lack of standardization and adequate maintenance. This study used tools from the Board of Innovation [BOI 2024], such as the Problem Understanding Card and the Future Scenario Maker, to explore these challenges and propose innovative future scenarios. The analyzed scenarios were the “DocuVerse”, which integrates software documentation with virtual reality; the “DocuAI”, which uses artificial intelligence to generate personalized documentation; and the “DocuChain”, which applies blockchain technology to ensure the integrity and transparency of documentation.

The most promising scenario identified was the “DocuAI”, which offers a dynamic and personalized approach to documentation creation, being able to revolutionize the way developers interact with these resources. However, the implementation of these solutions requires a meticulous and collaborative approach, involving both developers and researchers to ensure that documentation evolves in a way to meet practical needs and market demands. The adoption of such emerging technologies may not only improve the quality of documentations, but also increase the productivity and satisfaction of developers.

This study reinforces the importance of investing in researches and tools that aim at standardization and continuous improvement of software documentation, promoting a more efficient and collaborative development environment.

References *(kept verbatim from the original)*

Arya, D., Wang, W., Guo, J. L. C., and Cheng, J. (2019). Analysis and detection of information types of open source software issue discussions.

Berhouma, H. (2021). A generic model for software documentation and its application in embedded systems developed with scrum. In Proceedings of the 9th International Conference on Software and Information Engineering, ICSIE ’20, page 33–36, New York, NY, USA. Association for Computing Machinery.

BOI, I. (2024). Ai powered innovation. https://ai.boardofinnovation.com [Accessed: (12/07/2024)].

Cummaudo, A., Vasa, R., and Grundy, J. (2019). What should I document? A preliminary systematic mapping study into API documentation knowledge. arXiv:1907.13260 [cs].

Ellmann, M. and Timmann, I. (2019). A comparative study of faqs for software development. In Proceedings of the 2nd ACM SIGSOFT International Workshop on Software Qualities and Their Dependencies, SQUADE 2019, page 8–11, New York, NY, USA. Association for Computing Machinery.

Filgueira, R. and Garijo, D. (2022). Inspect4py: A knowledge extraction framework for python code repositories. In 2022

IEEE/ACM 19th International Conference on Mining Software Repositories (MSR), pages 232–236.

Thayer, K., Chasins, S. E., and Ko, A. J. (2021). A theory of robust api knowledge. ACM Trans. Comput. Educ., 21(1).

Uddin, G., Khomh, F., and Roy, C. K. (2021a). Automatic api usage scenario documentation from technical q&a sites.

Uddin, G., Khomh, F., and Roy, C. K. (2021b). Automatic api usage scenario documentation from technical qa sites.