REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

The Importance of Software Documentation in the Development and Maintenance Phases

Sagar Vishnubhai Sheta[1*]

[1*]Software Developer, Lathia Investments LLC

Abstract— This paper focuses on the value of documenting software the relationship between documentation and the efficiency of the development process and the usefulness of documentation in enhancing maintenance efforts. Research shows that effective documentation enhances administrative interpretable profile, shortens the time required in problem-solving, and fosters constructive cooperation, thereby boosting shorter developmental lifecycles and increasing consumer contentedness. To support elaborate documentation procedures to help achieve sustainable efficiency during software projects.

Index Terms— Software documentation, Code readability, Team collaboration, User-centered design, Documentation quality, Maintenance and troubleshooting

I. INTRODUCTION

Software documentation is the process of collecting information on software products and processes, crafted by users, developers, and maintenance teams through the complexities of management, software usage, and development. The main act of this software documentation is a roadmap that understands software functionality, operational procedures, and architecture (Ijiemr, 2024). This can be used for various purposes such as instructing end-users on navigating processes, developers use the software features by getting inside the codebase, development practices, and design decisions. Software documentation is important to act as a repository of collective knowledge and is vital for new team members in reducing their learning curve and understanding of software functionality (Ijirset, 2024). It further improves user experiences by providing necessary guidance in navigating the software. Moreover, it provides open-source projects across different locations that help to keep the work asynchronous.

1.1 Aim and Objectives

Aim

The main aim of this research is to examine the importance of comprehensive software documentation in the maintenance and development phases of software considering Python projects and focus on the impact of team collaboration, code quality, and long-term maintainability.

Objectives

- To evaluate the role of software documentation in improving software projects and code credibility that helps in team collaboration of work
- To analyze effective documentation practices that improve troubleshooting problems and maintain codebases
- To investigate the effectiveness of specific Python documentation tools in creating effective maintenance and accessible documentation for both users and developers.

II. LITERATURE REVIEW

2.1 Effectiveness of software Documentation in enhancing collaboration and code readability

Documentation can be used on software for several reasons, including increasing its readability and improving communication between the various stakeholders involved in the development process (Ijasem, 2024). It is an essential document that sets out the working and layout of the code to assist the developers in beginning implementation once they find out that the code is intended without having to analyze the code itself for this purpose. Observed code is particularly useful in environments where developers participate on a team basis and where there is usually a constant rotation of developers joining and leaving a team midway through the development of a particular project (Al-Saqqa et al., 2020). For both new and existing developers, documentation reduces time spent on learning by having the overall guideline to follow rather than clearly explaining what to do and where to find it in the codebase, decisions such as the use of some API, or execution of a complicated algorithm.

REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

Figure 2.1: Importance of code documentation
(Source: Qian et al., 2023)

Previous studies have indicated that complete documentation is helpful because it relieves the working capacity of the developers' brains helps them understand complex systems quicker, and reduces mistakes about changes (Hutchinson et al., 2021). It is the documentation that comprises commentaries within the code, descriptions of the functions, and the explanation of the overall system design presented in a project so that the developers spend much of their time and energy in an endeavor to understand how the whole program was developed rather than attempting to decipher the relative structures (Islam, and Ferworn, 2020). Furthermore, when documentation is maintained up-to-date it averts the problem of knowledge decay which is the loss of important information concerning the project. This can be especially difficult in projects that take a long time to complete because documentation may not be seen as important during some of the early phases only to become very important later on for such things as duplicating functionality, identifying problems, and so on (Akhtar et al., 2022).

2.2 Impact of documentation on troubleshooting and maintenance

Maintenance and fixing also require documentation since it creates an easy-to-understand source especially when a developer has to deal with a large code base for a long time (Carcary, 2020). Documentation is a knowledge base containing information about system architecture, and decisions taken in designing and defining the functions of the particular system, which may be extremely useful when troubles occur or when new teammates appear on the scene while the authors of the code are absent (Hou et al., 2023). It is discovered that quality documentation that may be poor or out-of-date means more time spent in maintenance because the developers cannot make heads or tails of code behavior or its interdependences.

Figure 2.2: Techniques for troubleshooting maintenance
(Source: Li et al., 2024)

REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

Code inspection could be the only other thing that is available to a developer – and, again, this is generally a slow and error-prone process that affects comprehension and even introduces new problems (Zhao, 2020). Reducing these issues is a matter of adequately documenting your work thus the purpose and rationale of segments of code can be quickly understood and fixed. Additionally, documentation is more important when transitioning from one system to another, or when there is a need to change the existing one because the documentation is helpful when guiding the implementation of change while seeking to have the least impact on the functionality of the system.
Further, documentation is also beneficial for improving troubleshooting because it enhances a systematic approach that is followed when troubleshooting (Qian et al., 2024). In that way, for instance, by pointing to detailed records of preceding incidents and their resolutions, developers can prevent work redundancies, optimize the identification of underlying problems, and apply remedies in a more effective manner (Patacas et al., 2020). This leads, on the one hand, to creating more dependable software while, on the other hand, providing for more efficient and cost-effective maintenance over time, all of which contributes to a stable, high-quality product life cycle.

2.3 Effectiveness of documentation in maintaining a gap between end-users and developers

Documentation should therefore be seen as a means of allowing developers to engage directly with end users and make software more usable and comprehensible between the two categories of these users (Nahar et al., 2022). From the developers' perspective, documentation is a guide that shows complex structures and work of the software, its functions, and principles of construction. For end-users, it is a resource that teaches them how to navigate the software, outlining how a specific tool works, how to fix problems that may happen, and how to complete activities on their own (Zhao et al., 2022). Current research shows that detailed and easy-to-follow documentation decreases end-user reliance on helpdesk service assistance since the users have enough information on the management of minor problems.

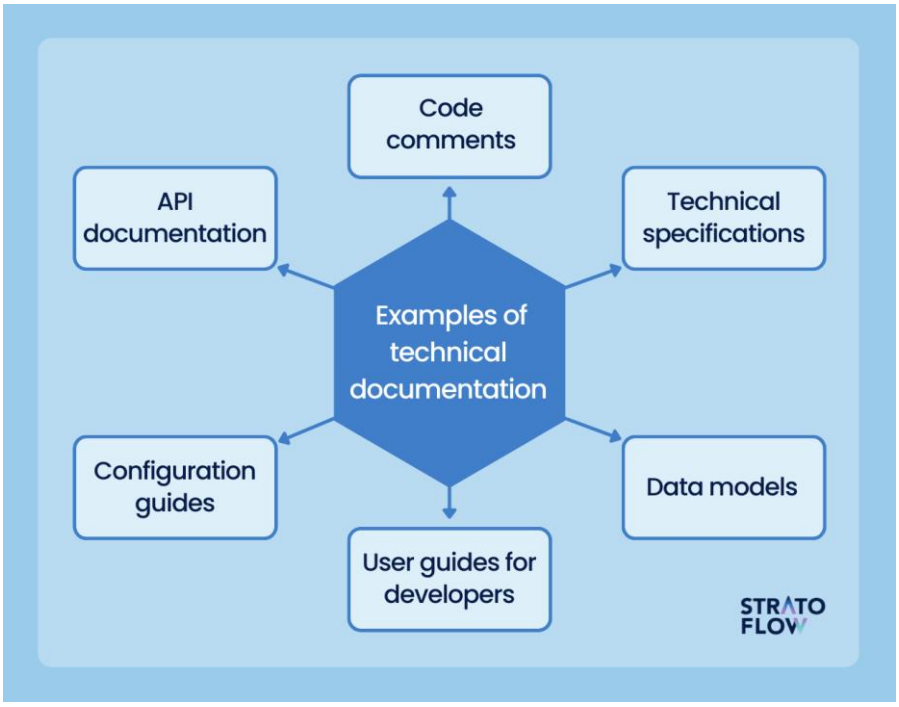

Figure 2.3: Technical documentation process
(Source: Thota et al., 2020)

System documentation is well formatted and categorized to include tutorials, FAQs, and troubleshooting guides which enable the end-users to gain confidence using the software thus enhancing satisfaction (Serban et al., 2020). Moreover, content segregated by levels of experience like for novices, intermediate users, and advanced users also increases the usability of complex software by offering information for a variety of users (Xiang and Chin, 2021). Studies prove that a process of such documentation focused on inclusion can increase the total user experience and, therefore, the rating of the software.
For the developers, the advantages of creating documents with end users in mind are secondary but quite profound (Barrane et al., 2021). Regular consumers can search out solid advice they need on their own they do not bother the developers by contacting them with common mistakes, and those developers can then spend their time fixing and upgrading their programs instead (Dewi et al., 2021). This balance helps to maintain a substantial working rate of developers while keeping the general user-to-developer relations in a positive aspect that would meet the requirements of both parties.

REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

2.4 Literature Gap

Although significant attention is paid to documentation as a tool to increase software readability, collaboration, and maintainability, the opportunities for improving documentation (Spring et al., 2022). It must have to more attention is needed with regard to techniques to document solutions that are appropriate for a given project based on the users' experience level and the nature of the project.

III. METHODOLOGY

3.1 Research Approach

The proposal for the study design of this study will draw on the mixed method research design as it offers both comprehensive and equally valid qualitative and quantitative research findings on the role of software documentation in the development and maintenance stages (Iwanaga et al., 2022). First, a bibliographic survey of the state of the art in theories will be carried out to assess the common tendencies and gaps found in the software documentation literature. This will help in putting down root understandings that will guide the collection of primary data in the study of the wider context (Marion and Fixson, 2021). After that, since participants from OSS projects would provide primary data, a quantitative analysis employing Python programming language shall be implemented. Using the Python packages like Pydoc, Sphinx, and text mining tools the information on the code documentation, code readability, and the frequency of maintenance updates will be gathered (Tan et al., 2021). The proposed choice of both quantitative and qualitative methods will further allow for the assessment of documentary practices and comparison with stakeholders' expectations that will thus embed the proposed analysis of software documents into both technical and user-oriented perspectives.

3.2 Data collection process

In the data collection process, both external secondary data and primary data which will be collected using Python shall be used to gather detailed information regarding the role and efficiency of software documentation (Majumdar et al., 2022). Secondary data will be collected through reviewing literature mainly from journals and technical publications to set the baseline of best practices, documentation, and current innovations.

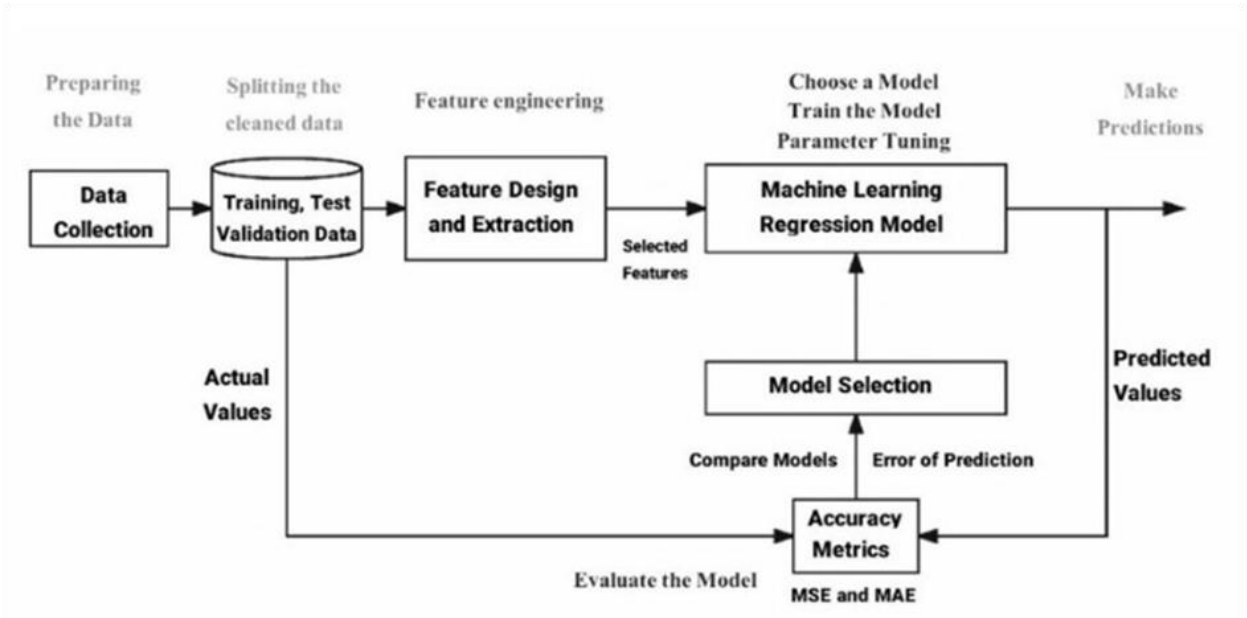

Figure 3.1: Process flow of ML algorithms
(Source: Sharma et al., 2021)

Some of the primary data will be gathered using Python to analyze docs and differences in usual documentation patterns in actual code bases. With the help of Python libraries like Pydoc and Sphinx, it became possible to assess the comprehensiveness, consistency, and compliance of various aspects of selected open-source projects' documentation (García et al., 2020). Furthermore, it is possible to obtain results indicating the average of such characteristics as code readability, comment density, and the usage of docstrings. Additional analysis can be performed using a text analysis toolkit in Python to mine information that relates to the quality of the documentation from the developers' forum or perhaps GitHub.

REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

3.3 Data Analysis

The ‘’Data Analysis Technique’ will use a statistical and machine learning approach to measure the effect of documentation on development and maintenance (Qian et al., 2023). First, the basic quantitative descriptive measures and graphical techniques like histograms and kernel density plots will be used to describe and analyze the quality and distribution of documentation for various projects (Chiozzi et al., 2024). To understand the existent links between the variables, correlation analysis is going to be used to analyze documentation quality against such factors as maintenance time, bug fix time, and development time (Abdulkareem and Abboud, 2021). Box plots, for example, will draw out and compare the maintenance time across levels of documentation quality. Furthermore, the feature correlation matrix presented as the heatmap will allow the determination of the dependencies between variables that help identify certain patterns and relationships that are important to documentation (Iwanaga et al., 2022). Regression kinds of options, for instance, might be used for predictiveness analysis, which will extend the understanding of how quality in the documentation might predict efficiency in development (Zhang et al., 2023). Collectively, these techniques offer an integrated perspective to the identified question regarding documentation about software results.

3.4 Evaluation Metrics

The consideration of evaluation metrics in this study has created an effective impact in improving the quality of documentation on the software development process and also increased maintenance efficiency (Liu et al., 2024). The evaluation metrics are also responsible for the performance of developed models in providing an appropriate assessment of this documentation process.

Documentation quality score: It helps to measure the clarity and thoroughness of the documentation on a specific scale (1-5) to get access to overall effectiveness.

Maintenance time: It can record the average time spent in maintaining the tasks and improve troubleshooting efficiency for documentation.

Bug Fix time: It also helps to measure the required time to resolve issues and solve problems based on documentation clarity.

Team collaboration index: It assesses the team cohesiveness by comparing the work of teams and their code integration analyzing how well the documentation contributed to the productivity of both (Bean et al., 2022). Combined, these measurements have been an effective scorecard for reviewing the part documentation plays when the goal has been to make productivity improvements, increase accuracy, promote development and maintenance, and decrease errors.

IV. RESULT AND DISCUSSION

4.1 Result

```
print(data.head())
```

```
   DocumentationQuality  MaintenanceTime  DevelopmentTimeReduction  \
0                     4               26                        34
1                     5               20                        22
2                     3               23                        27
3                     5               18                        42
4                     5               21                        25

   BugFixTime  TeamCollaborationRating  CodeQuality DocumentationFrequency  \
0    2.091886                        5            2                Weekly
1    2.083385                        4            3                Rarely
2    1.798983                        4            4                 Daily
3    2.139515                        3            3                Weekly
4    2.309014                        5            5                 Daily

   HighQualityDocumentation
0                         0
1                         0
2                         0
3                         0
4                         1
```

Fig. 4.1: Load dataset

This figure shows how to import the dataset into the analysis environment of jupyter notebook. The data was gathered with the help of background research largely based on the effects that software documentation can have on development and maintenance. These are the quality of documentation, the time taken in maintenance, time taken in development, time taken in bug fixing, the team collaboration index, the quality of codes, and frequencies of documentation updates (Tjanaka et al., 2023). The data was collected to assess the level of complementarities between these aspects of documentation and

REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

the levels of efficiency and effectiveness in software development and maintenance. This is an important step in the data preparation process for machine learning, preprocessing, feature engineering, and model construction. A basic understanding of the structure of data is critical before decoding it and making means so that other critical analyses are relevant (Liu et al., 2020). The data is then prepared for preprocessing operations such as dealing with the missing values and the conversion of the categorical features which is so important in aspects of machine learning.

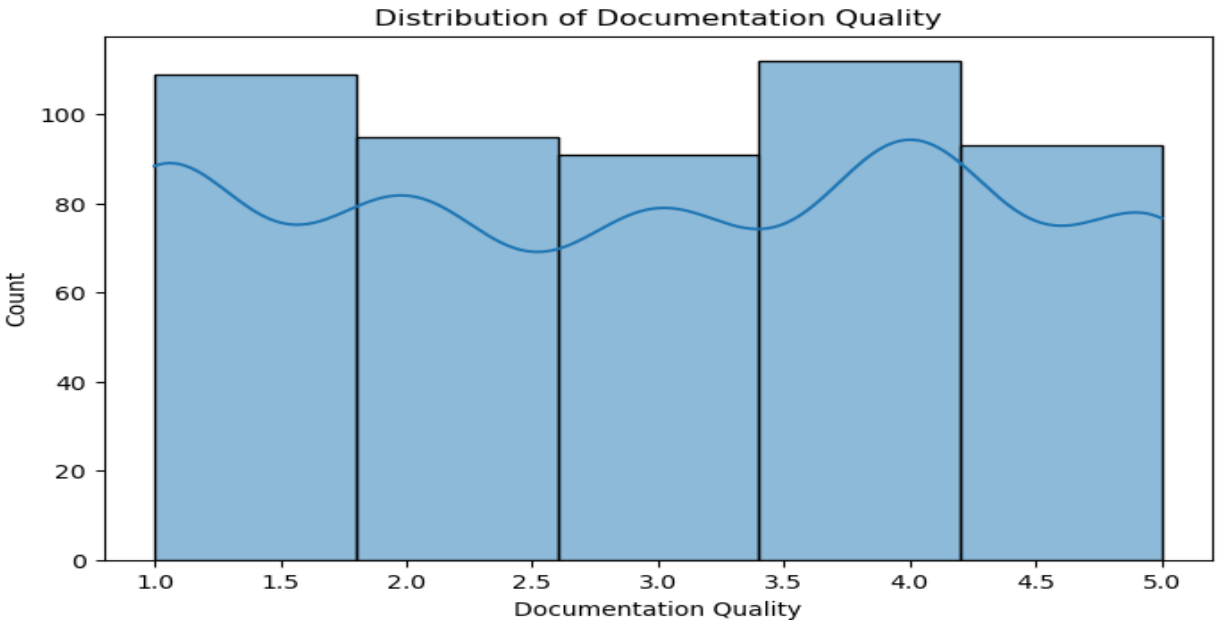

Fig 4.2: Distribution of Documentation Quality

The above image shows the actual distribution of the DocumentationQuality as one of the values indicating the perceived quality of documentation as found within the dataset. The data collected is secondary data, The documentation quality is scored out of 5 based on background research done. The distribution is shown in the form of the histogram that presents number of times a certain quality rating was given. It should also add a kernel density estimate overlay to help visualize the smoothness and pattern of the data representation (Dehaerne et al., 2022). This visualization is important as it will give an understanding of the approximate condition of the documentation within the dataset. It also emphasizes where possible weakness might be, for example, a dense area of lower ratings which may indicate poor documentation procedures that in turn affect software development and maintenance (Hou et al., 2023). This distribution is a critical knowledge area when performing analysis on various other variables related to documentation quality.

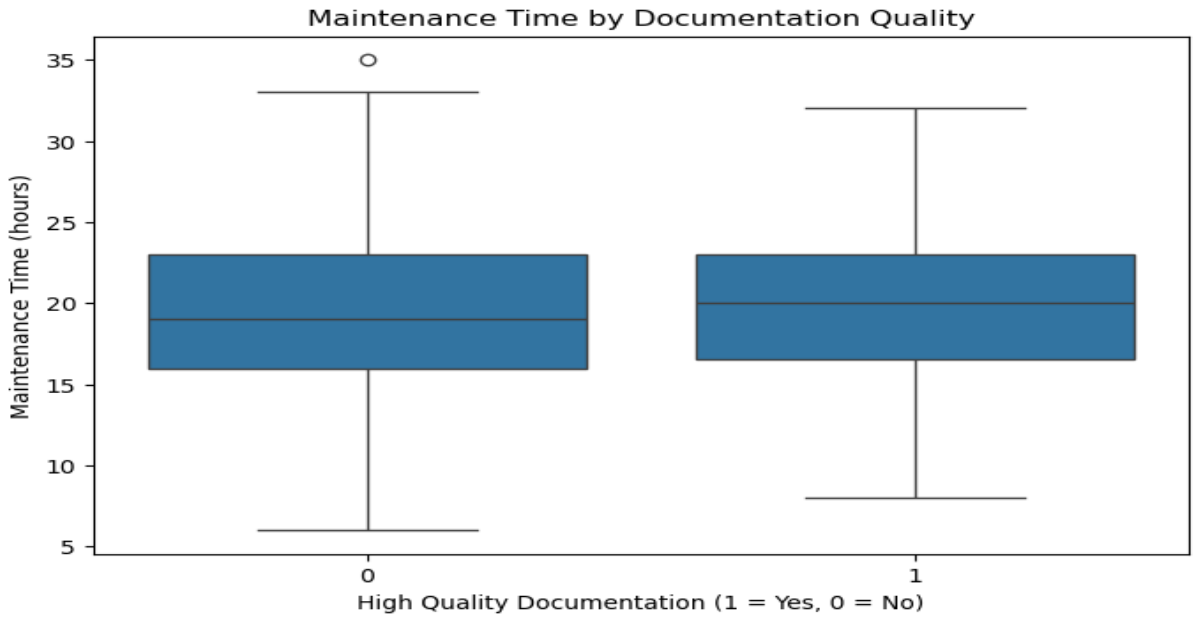

Fig. 4.3: Maintenance Time by Documentation Quality

REDVET - Revista electrónica de Veterinaria - ISSN 1695-7504
Vol 24, No. 3 (2023)
http://www.veterinaria.org
Article Received: 03/06/2023 Revised: 04/06/2023 Accepted: 04/07/2023

The effect of quality of documentation on software maintenance time is depicted in Figure 4.3 above. Having gathered background data from online research, this paper expounds on how the quality of documentation influences the performance of maintenance work. This relationship is best represented with the help of a box plot wherein the maintenance time is compared for the high-quality documentation, that is, if it was rated 4 or above, and low-quality documentation. It is however easy to deduce the differences in central tendencies (median values) and dispersion of the maintenance times of the two groups from the help of the box plot. It also gives an understanding of variation in maintenance time, and how the well-documented systems decrease the maintenance activities time and probable time delay (Golendukhina et al., 2022). This is exemplified in the figure where the relative time spent on maintenance is low for projects within the second category of documented projects The implication is that good documentation helps to fast-track the troubleshooting processes and minimize or eliminate errors while handling updates or bug fixes (Golendukhina et al., 2022). This diagram also proves that massive documentation greatly minimizes the time one spends maintaining the software systems.

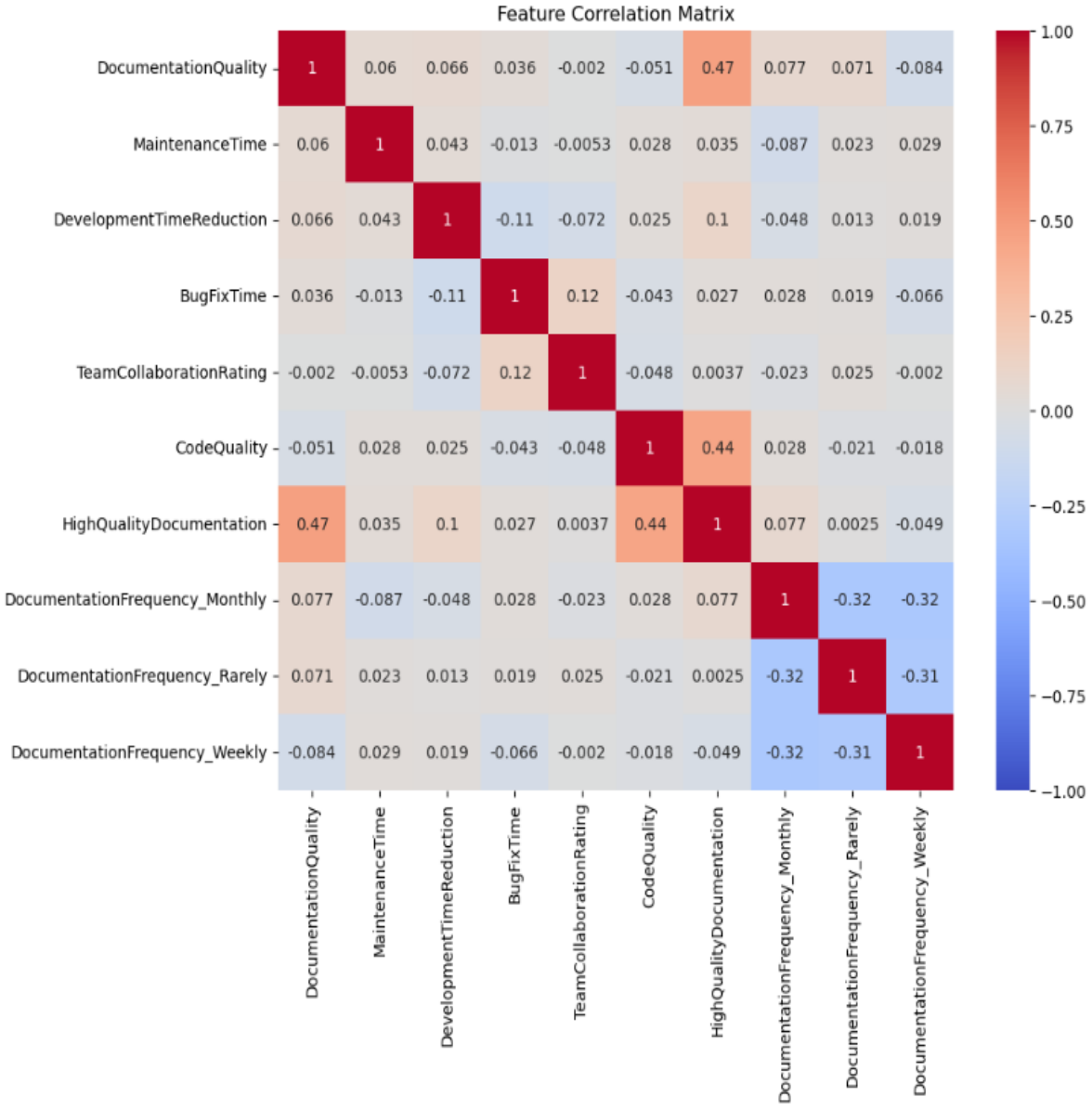

Fig. 4.4: Feature Correlation Matrix

Figure 4.4 shows the correlation matrix of different variables present in the dataset which has highlighted the association of documentation-related variables with other characteristics of the project. The matrix created takes the form of a heatmap where colored blocks represent the level of correlation between the given pair of features. It is very easy to detect strong positive or negative relationships when using CausalPie charts to understand how features like DocumentationQuality, DevelopmentTimeReduction, MaintenanceTime, and BugFixTime are related (Pavao et al., 2023). High covariance between Documentation quality and Development time reduction could be interpreted that a high level of documentation

implies shorter development cycles. The matrix also shows limited cohesiveness between specific variables and found that they are likely independent of each other in those fields. This visualization is useful for understanding which issues should be focused on in software documentation processes. From these relationships, the patterns obtained in the data analysis can inform better decisions concerning documentation to heighten the robustness of development and maintenance.

4.2 Discussion

The results of the collected data on software documentation reflect many aspects of the effect of the documentation on software development and maintenance. It becomes apparent that documenting work efforts is important feedback while working on a project to ensure that the time and effort to maintain a project are reduced as well as the time taken for software development (Mastropaolo et al., 2021). Maintenance costs of well-documented systems are usually low since good documentation enhances a great understanding of a system by the teams working on the system. This enables faster identifying the issues, less time being spent on fixage, and further quickens updates (Muñoz et al., 2024). Also, there is an association between the number of times a documentation is updated and improved teamwork. This means that having updated documentation always keeps all the team members including the new joiners aware of the system as it is, hence minimizing the confusion that is brought about by outdated information on the system.

The results also show that where documentation quality is high, development time is normally brought down drastically. Documentation helps a lot because it gives the developer a clear look at what is already in the code and where to look as well as how to address known problems in a specific area (Lin et al., 2021). On the other hand, poor documentation results in a lengthened development and maintenance cycle, since the developers spend more time interpreting the system, and additional time is required in implementing or maintaining software (Wei et al., 2022). These observations suggest that comprehensive and contemporary software documentation contributes a critical mass to enhance efficiency and avoid higher operating costs in the long run.

V. CONCLUSION

Most development activities require software documentation that affects the improvement of development and maintenance. Documentation benefits project time reduces cost, writes well-structured and comprehensible code, and uses less time for bug fixing. Research indicates and evidence proves that well-commented code results in less development time, and fewer errors and most importantly, acknowledges the roles of multi-coding as a positive protocol. However, properly documented software ensures closer working proximity of the developers with the end-users, and that is a very important aspect of user-centricity.

VI. Acknowledgment

I am pleased to present my report titled "***The Importance of Software Documentation in the Development and Maintenance Phases***". I wish to extend my heartfelt gratitude to those who have supported me in completing this research. I am deeply thankful to those who assisted in gathering the necessary data throughout this study. My sincere appreciation goes to my professors for their invaluable guidance and insights.

I also want to express my gratitude to my friends whose support and encouragement played a crucial role in achieving our shared objectives.

I acknowledge the unwavering support of my batch mates, supervisors, and professors throughout this endeavor. Any shortcomings in this research are entirely my responsibility.

REFERENCES

[1] Ijiemr, 2024. EXPLORE HOW AI CAN BE USED TO CREATE DYNAMIC AND ADAPTIVE FRAUD RULES THAT IMPROVE THE DETECTION AND PREVENTION OF FRAUDULENT ACTIVITIES IN DIGITAL BANKING. Available at: https://www.ijiemr.org/downloads/paper/Volume-13/explore-how-ai-can-be-used-to-create-dynamic-and-adaptive-fraud-rules-that-improve-the-detection-and-prevention-of-fraudulent-activities-in-digital-banking [Accessed on: 11th November, 2024]

[2] Ijirset, 2024. Artificial Intelligence Ethics: Investigating Ethical Frameworks, Bias Mitigation, and Transparency in AI Systems to Ensure Responsible Deployment and Use of AI Technologies. Available at: https://www.ijirset.com/upload/2024/september/2_Artificial.pdf [Accessed on: 11th November, 2024]

[3] Ijasem, 2024. DESIGN AND DEVELOPMENT OF ARDUINO-BASED COLD CONTAINER. Available at: https://www.ijasem.org/previousissue.php?year=2023&issue=3 [Accessed on: 11th November, 2024]

[4] Al-Saqqa, S., Sawalha, S. and AbdelNabi, H., 2020. Agile software development: Methodologies and trends. International Journal of Interactive Mobile Technologies, 14(11).

[5] Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z. and Sun, M., 2023. Communicative agents for software development. arXiv preprint arXiv:2307.07924, 6.

[6] Hutchinson, B., Smart, A., Hanna, A., Denton, E., Greer, C., Kjartansson, O., Barnes, P. and Mitchell, M., 2021, March. Towards accountability for machine learning datasets: Practices from software engineering and

infrastructure. In Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (pp. 560-575).

[7] Islam, A.K.M.Z. and Ferworn, A., 2020. A Comparison between Agile and traditional software development methodologies. Global Journal of Computer Science and Technology, 20(2), pp.7-42.

[8] Akhtar, A., Bakhtawar, B. and Akhtar, S., 2022. Extreme programming vs scrum: A comparison of agile models. International Journal of Technology, Innovation and Management (IJTIM), 2(2), pp.80-96.

[9] Carcary, M., 2020. The research audit trail: Methodological guidance for application in practice. Electronic Journal of Business Research Methods, 18(2), pp.pp166-177.

[10] Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J. and Wang, H., 2023. Large language models for software engineering: A systematic literature review. ACM Transactions on Software Engineering and Methodology.

[11] Li, K., Zhu, A., Zhao, P., Song, J. and Liu, J., 2024. Utilizing deep learning to optimize software development processes. arXiv preprint arXiv:2404.13630.

[12] Zhao, J., 2020. Quantum software engineering: Landscapes and horizons. arXiv preprint arXiv:2007.07047.

[13] Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X. and Xu, J., 2024, August. Chatdev: Communicative agents for software development. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 15174-15186).

[14] Patacas, J., Dawood, N. and Kassem, M., 2020. BIM for facilities management: A framework and a common data environment using open standards. Automation in Construction, 120, p.103366.

[15] Nahar, N., Zhou, S., Lewis, G. and Kästner, C., 2022, May. Collaboration challenges in building ML-enabled systems: Communication, documentation, engineering, and process. In Proceedings of the 44th International Conference on software engineering (pp. 413-425).

[16] Zhao, J., Feng, H., Chen, Q. and de Soto, B.G., 2022. Developing a conceptual framework for the application of digital twin technologies to revamp building operation and maintenance processes. Journal of Building Engineering, 49, p.104028.

[17] Thota, M.K., Shajin, F.H. and Rajesh, P., 2020. Survey on software defect prediction techniques. International Journal of Applied Science and Engineering, 17(4), pp.331-344.

[18] Serban, A., Van der Blom, K., Hoos, H. and Visser, J., 2020, October. Adoption and effects of software engineering best practices in machine learning. In Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (pp. 1-12).

[19] Xiang, Z.T. and Chin, J.F., 2021. Implementing total productive maintenance in a manufacturing small or medium-sized enterprise. Journal of Industrial Engineering and Management (JIEM), 14(2), pp.152-175.

[20] Dewi, L.J.E., Wijaya, I.N.S.W. and Seputra, K.A., 2021, March. Web-based Buleleng Regency agriculture product information system development. In Journal of Physics: Conference Series (Vol. 1810, No. 1, p. 012029). IOP Publishing.

[21] Barrane, F.Z., Ndubisi, N.O., Kamble, S., Karuranga, G.E. and Poulin, D., 2021. Building trust in multi-stakeholder collaborations for new product development in the digital transformation era. Benchmarking: An International Journal, 28(1), pp.205-228.

[22] Spring, M., Faulconbridge, J. and Sarwar, A., 2022. How information technology automates and augments processes: Insights from Artificial-Intelligence-based systems in professional service operations. Journal of Operations Management, 68(6-7), pp.592-618.

[23] Iwanaga, T., Usher, W. and Herman, J., 2022. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. Socio-Environmental Systems Modelling, 4, pp.18155-18155.

[24] Marion, T.J. and Fixson, S.K., 2021. The transformation of the innovation process: How digital tools are changing work, collaboration, and organizations in new product development. Journal of Product Innovation Management, 38(1), pp.192-215.

[25] Tan, J., Feitosa, D., Avgeriou, P. and Lungu, M., 2021. Evolution of technical debt remediation in Python: A case study on the Apache Software Ecosystem. Journal of Software: Evolution and Process, 33(4), p.e2319.

[26] Majumdar, S., Bansal, A., Das, P.P., Clough, P.D., Datta, K. and Ghosh, S.K., 2022. Automated evaluation of comments to aid software maintenance. Journal of Software: Evolution and Process, 34(7), p.e2463.

[27] Sharma, P.N., Savarimuthu, B.T.R. and Stanger, N., 2021, May. Extracting rationale for open source software development decisions—a study of python email archives. In 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE) (pp. 1008-1019). IEEE.

[28] García, S., Strüber, D., Brugali, D., Berger, T. and Pelliccione, P., 2020, November. Robotics software engineering: A perspective from the service robotics domain. In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 593-604).

[29] Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z. and Sun, M., 2023. Communicative agents for software development. arXiv preprint arXiv:2307.07924, 6.

[30] Chiozzi, G., Andolfato, L., Argomedo, J., Cano, C.D., Frahm, R., Hofer, J., Jeram, B., Kornweibel, N., Pellegrin, F., Schilling, M. and Sommer, H., 2024, July. Status of the ELT control software development. In Software and Cyberinfrastructure for Astronomy VIII (Vol. 13101, pp. 36-56). SPIE.

[31] Abdulkareem, S.A. and Abboud, A.J., 2021, February. Evaluating Python, c++, javascript, and Java programming languages based on software complexity calculator (Halstead metrics). In IOP Conference Series: Materials Science and Engineering (Vol. 1076, No. 1, p. 012046). IOP Publishing.

[32] Iwanaga, T., Usher, W. and Herman, J., 2022. Toward SALib 2.0: Advancing the accessibility and interpretability of global sensitivity analyses. Socio-Environmental Systems Modelling, 4, pp.18155-18155.

[33] Zhang, Q., Fang, C., Xie, Y., Zhang, Y., Yang, Y., Sun, W., Yu, S. and Chen, Z., 2023. A survey on large language models for software engineering. arXiv preprint arXiv:2312.15223.

[34] Liu, J., Wang, K., Chen, Y., Peng, X., Chen, Z., Zhang, L. and Lou, Y., 2024. Large language model-based agents for software engineering: A survey. arXiv preprint arXiv:2409.02977.

[35] Liu, J., Wang, K., Chen, Y., Peng, X., Chen, Z., Zhang, L. and Lou, Y., 2024. Large language model-based agents for software engineering: A survey. arXiv preprint arXiv:2409.02977.

[36] Bean, B., Bhatnagar, S., Castro, S., Meyer, J.D., Emonts, B., Garcia, E., Garwood, R., Golap, K., Villalba, J.G., Harris, P. and Hayashi, Y., 2022. CASA is the Common Astronomy Software Application for radio astronomy. Publications of the Astronomical Society of the Pacific, 134(1041), p.114501.

[37] Tjanaka, B., Fontaine, M.C., Lee, D.H., Zhang, Y., Balam, N.R., Dennler, N., Garlanka, S.S., Klapsis, N.D. and Nikolaidis, S., 2023, July. my ribs: A bare-bones Python library for quality diversity optimization. In Proceedings of the Genetic and Evolutionary Computation Conference (pp. 220-229).

[38] Liu, H., Eksmo, S., Risberg, J. and Hebig, R., 2020, June. Emerging and changing tasks in the development process for machine learning systems. In Proceedings of the International Conference on software and System Processes (pp. 125-134).

[39] Dehaerne, E., Dey, B., Halder, S., De Gendt, S. and Meert, W., 2022. Code generation using machine learning: A systematic review. Ieee Access, 10, pp.82434-82455.

[40] Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J. and Wang, H., 2023. Large language models for software engineering: A systematic literature review. ACM Transactions on Software Engineering and Methodology.

[41] Golendukhina, V., Lenarduzzi, V. and Felderer, M., 2022, May. What is software quality for AI engineers? Towards a thinning of the fog. In Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI (pp. 1-9).

[42] Golendukhina, V., Lenarduzzi, V. and Felderer, M., 2022, May. What is software quality for AI engineers? Towards a thinning of the fog. In Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI (pp. 1-9).

[43] Pavao, A., Guyon, I., Letournel, A.C., Tran, D.T., Baro, X., Escalante, H.J., Escalera, S., Thomas, T. and Xu, Z., 2023. Codalab competitions: An open-source platform to organize scientific challenges. Journal of Machine Learning Research, 24(198), pp.1-6.

[44] Mastropaolo, A., Aghajani, E., Pascarella, L. and Bavota, G., 2021, September. An empirical study on code comment completion. In 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 159-170). IEEE.

[45] Muñoz, A.D., Monje, M.R. and Velthuis, M.G.P., 2024. Towards a set of metrics for hybrid (quantum/classical) systems maintainability. Journal of Universal Computer Science, 30(1), p.25.

[46] Lin, J., Ma, X., Lin, S.C., Yang, J.H., Pradeep, R. and Nogueira, R., 2021. Pyserini: An easy-to-use Python toolkit to support replicable IR research with sparse and dense representations. arXiv preprint arXiv:2102.10073.

[47] Wei, A., Deng, Y., Yang, C. and Zhang, L., 2022, May. Free lunch for testing: Fuzzing deep-learning libraries from open source. In Proceedings of the 44th International Conference on Software Engineering (pp. 995-1007).

[48] Ostadabbas, H., Weippert, H. and Behr, F.J., 2020. Using the synergy of the field for collecting data on-site and qgis for interactive map creation by alkis® data extraction and implementation in Postgresql for urban planning processes. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 43, pp.679-683.

[49] Gao, Z., Xia, X., Lo, D., Grundy, J. and Zimmermann, T., 2021, August. Automating the removal of obsolete TODO comments. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (pp. 218-229).

[50] Liu, M., Fang, S., Dong, H. and Xu, C., 2021. Review of digital twin about concepts, technologies, and industrial applications. Journal of manufacturing systems, 58, pp.346-361.