Software Systems Documentation: A Systematic Review

Abdullah A H Alzahrani
Department of Computers-Engineering and Computing College at Alqunfuda,
Umm Al Qura University, Makkah, Saudi Arabia

***Abstract*—In the domain of software engineering, software documentation encompasses the methodical creation and management of artifacts describing software systems. Traditionally linked to software maintenance, its significance extends throughout the entire software development lifecycle. While often regarded as a quintessential indicator of software quality, the perception of documentation as a time-consuming and arduous task frequently leads to its neglect or obsolescence. This research presents a systematic review of the past decade's literature on software documentation to identify trends and challenges. Employing a rigorous systematic methodology, the study yielded 29 primary studies and a collection of related works. Analysis of these studies revealed two primary themes: issues and best practices, and models and tools. Findings indicate a notable research gap in the area of software documentation. Furthermore, the study underscores several critical challenges, including a dearth of automated tools, immature documentation models, and an insufficient emphasis on forward-looking documentation.**

Keywords*—Software engineering; software systems documentation; software maintenance; software quality; software development

I. Introduction

Software documentation can be defined as the journey of producing different types of documentation. These types vary in their purposes from describing the development processes to describing the final product to the intended user. It is believed that software engineers should have the responsibility of software documentation, however, professionals in technical writing are sometimes needed [1], [2], [3], [4].

In the past 10 years, software documentation has been an interest in the industry of software engineering. However, majority of documenters are whether technicians or peoples trained in humanity. Therefore, the need for more professionals in the software documentation had emerged [5].

Many benefits can be accrued from good software documentation such as decreased costs of maintenance [6], [7], [8], [9], [10]. However, achieving a well-formed software documentation might be challenging. One challenge in software documentation is that developers abhor being involved in software documentation. In addition, some believe that poor software documentation is worse than no documentation. Furthermore, lack of professionals in the software documentation is considered to be another challenge [5], [11].

Software documentations principles are to be considered during software documentation. These principles are the level of details, document purpose and intended readers, use of graphical aids, clarity and precision, language of document, and documents versions. Therefore, in order to acquire a well-written software documentation, it is important to pay attention to, first, the purpose of the documents and the intended audience. This will lead to choosing the appropriate language, level of details, and graphical aids. However, in order to keep the documentation alive, updating documents with use of versions management are essential [6], [11], [12], [13], [14].

Several types of software documents are generated in the process of software documentation. These types diverse based of their purposes and intended readers. However, these types fall in one of the following categories. The first category is the documentation of the process of software development. This includes documentation of requirements, planning, implementation and other documents during the development journey. The intended readers for this category of documents are the developer, software, decision makers, and the maintainers. The second category is the documentation of the product after delivery. This category includes the documents that describe the product for intended users. Examples of this category are User manuals and system main structure documents [1], [2], [3], [5], [6], [8], [11], [14].

Many techniques and tools are employed for software documentation. In general Waterfall technique and Iterative technique are the most dominant techniques for software documentations. However, for the tools that are used for the documentation, many have stated a variety of tools such as MS Word XML and other Text Editors, Doxygen, Visio, FrameMaker, Author-IT, Doc++, Rational Rose, JUnit and other tools. Some of these tools aid in automation of documentation to some extents [15], [16], [17], [18], [19].

The importance of this study derived from the need to achieve solutions that overcome the unsolved problem of poor or absent software documentation as this issue remains unsolved. Software documentation improves the quality of software systems and consequently improves maintainability and cost efficiency. Furthermore, drawing more attention to the topic of software documentation would enhance documentations models as well as templates employed, which relatively enhance automation of software documentation.

This paper has been structured as follows. Section I introduced the topic software systems documentation. In addition, it highlights the importance of systematic review on the considered topic. Section II illustrates the methodology which has been employed in this paper and formulated the research question. In addition, the main findings statistically shown and discussed. Section III has been divided into two subsections. The first subsection concludes the discussion on the findings and demonstrates the trends in the software documentation publications. The second subsection reviews and discusses the primary studies found in this research and categorizes them into two categories. Finally, Section IV draws conclusions on this research.

II. Methodology

Budgen et al. and Kitchenham [20], [21] have described a systematic review methodology which this study applies. The methodology enables conducting the reviews objectively and structurally. Furthermore, the methodology allows demonstration of broader picture of the topic of software documentation by categorizing the results into primary and related to the topic under consideration.

1) Research question: What are the unveiled trends and issues in relation to software documentation in the last 10 years?

The above research question can be responded to by first exploring the software documentation topic, then, investigating the existing techniques and tools which are employed. Consequentially, issues and difficulties will be highlighted and identified. Therefore, the keywords leading the search in the well-known databases have been enumerated. These keywords are software system documentation - software documentation - system documentation - automated documentation - software knowledge documentation - computer software documentation - software engineering documentation.

2) Sources selection: Bearing in mind the aforementioned keywords, a search query has been formulated as shown in Table I. An OR logical operator has been used in order to combine results that are related to the search. In addition, to narrow the range of the results, double quotation marks (“”) have been applied to surround the keywords.

TABLE I. SEARCH QUERY

Search query	“software system documentation” OR “software documentation” OR “system documentation” OR “automated documentation” OR “software knowledge documentation” OR “computer software documentation” OR “software engineering documentation"

After formulating the Search query, it has been entered to The Saudi Digital Library (SDL) [22] search engine. SDL is an online digital library resource that allows searching in many well-known digital libraries such as Springer, IEEE Digital Library, ACM Digital Library, SAGE, ScienceDirect, and other publishers. Different types of research items can be found; however, the results were grouped into four main categories. The first category is Conference Paper. Second category is Journal papers which include journal Article, Case Study papers, and Review Papers. Third category is Books which include books, handbooks, Thesis, and technical reports. Fourth category is Others which include else results.

This paper’s goal is to investigate the topic of software documentation in the past 10 years, so, an exclusion criterion of year of publication has been applied and configured to include work which has been done between the years 2014 and 2024. In addition, another exclusion criterion was the language of publication as non-English publications have been eliminated during the search process by configuring the search engine to exclude them before displaying the results from the digital libraries. Finally, inclusion criterion relies on the analysis of several aspects in found results. These aspects are title, keywords, abstract, and conclusion. So, the analysis is the process of manually reading deciding for publications to be considered relevant or not.

Table II illustrates the results of the searching process. It is clear in the table that the first results after applying the query string was total of 7482 publications found in different source. However, this number includes the repeated items and the flawed entries of the items. Therefore, it was necessary to accurate the results by eliminating those items. Consequently, the total number of publications declined to be 3453 items.

Inclusion criterion was then applied manually to determine the relevance of remaining items by scanning the keywords, abstracts, and conclusions. The process resulted in eliminating more items. A total of 1654 items are the relevant items to be investigated and studied in order determine the results of primary studies to the topic of software documentation.

TABLE II. NUMBER OF PUBLICATIONS FOUND FROM DIFFERENT SOURCES

Sources	**Publication**					
	Search date	**Found**	**Not repeated**	**Relevant**	**Primary**	**%**
Springer Link	3 Jan 2024	612	411	378	0	0%
IEEE Digital Library	3 Jan 2024	125	82	65	5	17%
ACM Digital Library	3 Jan 2024	41	34	12	6	21%
SAGE	3 Jan 2024	248	188	90	0	0%
ScienceDirect	3 Jan 2024	41	39	21	0	0%
Other Publishers	3 Jan 2024	6415	2699	1088	18	62%
Total		7,482	3453	1654	29	100 %

It can be seen in Fig. 1 that relevant found publications to the topic of software documentation are divided into six categories. The categories are based on the reputation of the publisher. Therefore, five categories are designated to leading and well-known publishers which are Springer, IEEE, ACM, SAGE, and ScienceDirect. All other publishers have been considered in one category as “other publishers”.

Fig. 1 illustrates 66% of the relevant found publications to the topic of software documentation are from other publishers which have less reputation than the leading publishers. However, 23% of relevant found publications are from Springer. Having this in mind, 378 relevant publications to the topic of software documentation for the last 10 years in a well-known publisher such as Springer are not a considerable number of publications. This might raise a question on the reasons behind the low number of publications in the topic of software documentation in such leading and well-known publishers.

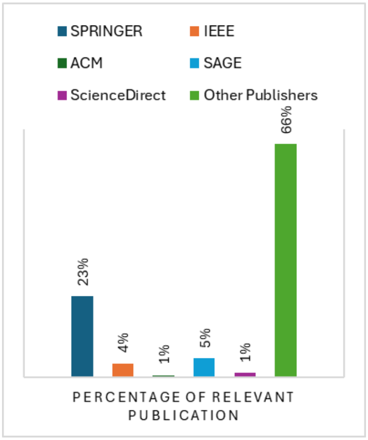

Fig. 1. Relevant studies from different publishers.

With regards to the primary studies shown in Table II, they are the studies that are major in the topic of software documentation. A complete list is included in Appendix A. These studies are identified to be primary after a manual excessive analysis and in depth reading of the relevant studies. The main criterion to identify the primary studies is the goal of the research. In particular, if the relevant study is offering a new model, approach, solution, explanation, case study, comparison, or/and review, then that relevant study is considered to be a primary study to the topic of software documentation.

III. Results and Discussion

A. Trends in Software Documentation Publications

This section is to discuss the findings on the collected data and the analysis of results of the conducted reviews. The main finding is that the topic of software documentation has not been considered sufficiently for research in the past 10 years, especially by well-known and leading publishers. This can be seen clear form the results shown in the previous section.

Fig. 2 illustrates the publications of relevant studies to the topic of software documentation over the last 10 years. From Fig. 2, a growing interest can be noticed from the year of 2020 in the publication is a well-known publisher which is Springer. However, this interest in the topic of software documentation has remained unnoticed in publications of other well-known publishers.

Table III demonstrates the types of relevant publications found. Three main categories of publications which are conference papers, journal papers, and book. Regarding journal papers, this includes regular article papers, Case Study papers, Review Paper. On the other hand, Books category includes books, handbook, Technical Report, and Thesis. Finally, all other publication types were classified into "Others" category.

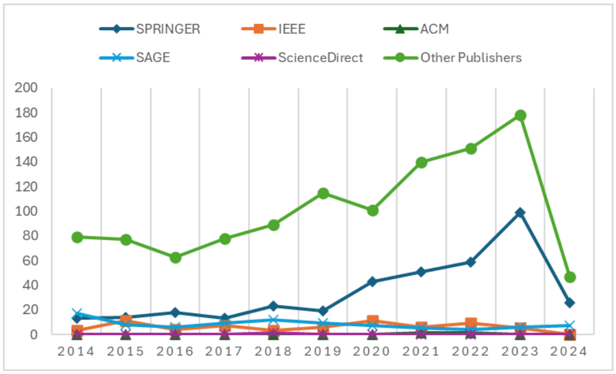

Fig. 2. Relevant studies from different publishers. Publications in the past 10 years on software documentations.

TABLE III. Types of Publications of Relevant Studies

Sources	Conference Paper	Journal Paper	Books	others	Total
Springer Link	3	348	1	26	378
IEEE Digital Library	25	10	0	30	65
ACM Digital Library	0	8	0	4	12
SAGE	0	38	51	1	90
ScienceDirect	0	12	0	9	21
Other Publishers	14	633	122	319	1088

Table III shows that the majority of relevant publications on software documentation are journals papers. Fig. 3 illustrates that around 63% of all relevant publications on the topic of software documentation over the past 10 years are journal papers. It is worth noting that from the leading and well-known publishers, SAGE has an outstanding interest on relevant books on the topic of software documentation.

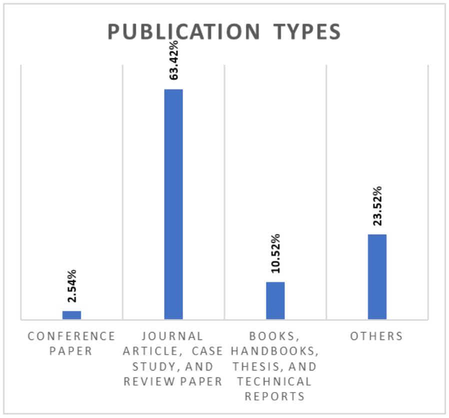

Fig. 3. Types of relevant studies.

B. State of Arts Studies

This section discusses the primary studies found in the systematic review on software documentations. The studies have been classified into three main divisions shown in the following subsections which are: 1) software documentations reviews, issues, and best practices; 2) software documentation models and tools.

1) Software documentations reviews, issues, and best practices: Many have introduced best practices and required attributes of software documentation [15], [23], [24], [25], [26]. Other researchers [17], [18], [27], [28] have considered software documentation issues which might be addressed earlier, throughout, and/or afterwards the process of the software documentations.

Uddin et al. [15] have conducted two surveys based on their previous work [23] on Application Programming Interface (API) documentation. First, the authors designed a three-questions survey circulated among IBM Canada labs. The number of participants was 69 with the following job titles: developer, architect, tester, and consultant. Second, the authors designed a seven-questions survey circulated among developers and architects at IBM Canada and UK. This time, the number of participants was 254. The authors concluded that the API documentation suffers contents issues which are related to the fact that the engagement of experts is needed in the documentation of API.

Rai et al. [29] have conducted a review on the topic of source code documentation over last 10 years. The authors formed six research questions that were to be answered. These questions focused on the methodologies, tools, and evaluation means for source code documentation. In addition, the authors found that source code documentation has grown interest from researchers and automatic generation of source code documentation begins to use Deep Learning approaches instead of Information Retrieval approaches.

Lethbridge et al. [6] have published a study in order to gather best practices in software documentation. The authors conducted a study that includes surveying, interviewing, and observing software engineers. in addition, the authors investigate the tools used by these software engineers. The findings of the study can be summarized in the followings: 1) focus on requirements documentation and high-level documentation of the systems rather than complete and UpToDate documentation; 2) focus on simple and customized documentation rather than forcing the use of particular documentations methods or tools. This is to avoid time overhead and complexity.

Forward et al. [18] have investigated the tools and technologies used in software documentations. The authors divided software documentation into six types of documents namely requirements, specifications, detailed design, and low level design, architecture, and QA documents. The main findings can be summarized as follows: 1) software documents are important and useful even if they are obsolete; 2) tools for software documentations are needed to automate the process and should be chosen based on the nature of the software project.

Santos et al. [7] have conducted a review on software documentation in order to check the essential quality attributes in software documents. In addition, the authors reviewed the best practices for software documentation from 14 different publications in order to establish relations between the quality attributes and the best practices offered in those publications.

De Souza et al. [16] have studied the impact of Agile on software documentations. The authors highlighted the software development using Agile relies on informal communication which might lead to lack of software documentation or obsolescence. The authors addressed the issue that might face documentation teams as they immensely require documentation to accomplish their tasks.

Aghajani et al [9], [19] have conducted an empirical study to investigate the issues in software documentation. The outcome of the study has shown 162 types of software documentation issues linked to tools, process, and presentations of information. In addition, the authors believe that the attitude and experience of people involved in the software documentation process are important factors in the success or failure of software documentation.

Meng et al. [30] have conducted a study to investigate the learning strategy that developers use when they encounter API documentation. The study was carried out using interviews and questionnaire methodologies on 17 developers who have been asked 45 questions in the interview and to answer online survey of 39 questions. Main findings showed that documentation of API lack of clarity and completeness. Moreover, un-updated documentation is another recurring issue in API documentation.

In 2015, Zhi et al. [31] reviewed 69 research articles published between the years 1971-2011 on software documentation in order to study the impact of documentation on cost and quality. The authors concluded their review with several findings which can be summarized as: 1) main quality attributes that should be in the documentation are completeness, accessibility, and consistency; 2) most of the evaluation on the documentation models are on a single case study or academic prototypes.

2) Software documentation models and tools: Falessi et al. [13] have conducted an empirical study on 50 postgraduate students in order to evaluate the effect effects using different techniques of Design Decision Rationale Documentation (DDRD). The focus on the experiment is to check how different groups react in requirements change.

Kajko-Mattsson [17] has introduced a model for software documentations that serves corrective maintenance. The model includes 19 requirements with each has a set of goals. The model has been examined by surveying 18 different Swedish organisations with the use of interview mean. The author reported that the results show that collaboration with maintenance teams is to the minimum and the maintenance teams are absent from the documentation process. This has led to inadequate support for decision making for any change as well as quality assurance costs time and effort.

Bachmann et al. [32] have introduced their model for documenting software architecture. The model aims to document layered view of the architecture of the software system. The main purpose is to provide documentation that helps in sharing understanding of the system, tracing the changes, and discussing trade-offs. In addition, the model clearly identifies different views of the software architecture based on the audience of the documentation.

Falessi et al. [12] have introduced value-based (VD) method for software documentation in particular documentation of design decision. The proposed method aims to enhance the use of DDRD approach and is called VD DDRD. The authors conducted an experiment in order to validate their approach and the results show that VD DDRD can moderate inhibitions which might be shown using DDRD.

Aguiar et al. [8] have introduced a methodology for software documentation in particular documenting object-oriented frameworks. The authors have addressed several issues that need to be considered when documenting frameworks. These issues are related to quality, processes, and tools. The approach is tailored to help naïve software engineers in documenting software frameworks. The approach addresses three roles namely writers, developers, and documentation managers. In addition, it emphasizes on the collaborations and involvement throughout the development process.

Véras et al. [33] have introduced an approach which helps assessing the software requirement specifications. The approach aims to provide a benchmark for the assessment process. Three checklists are offered by the approach which is based on the standardizations of Packet Utilization Standard (PUS) by European Cooperation for Space Standardization (ECSS) standards.

Farwick et al. [34] have proposed a semi-automatic approach that document Enterprise Architecture (EA). The approach is composed of 4 models each of which needs manual interventions. The authors aim to overcome academic approaches offered by Hauder et al. [35]. Although the approach is promising, evolution and more case studies are needed to mature it.

Mathrani et al. [36] have proposed a new approach for software documentation that relies on the use of a quality management standards model (ISO 9001). The approach has been case studied on healthcare software with teams applying Scrum methodology for software development. The authors reported that issues such as incompleteness and ambiguity might rise due to the constraints in ISO 9001.

Aversano et al. [37] have proposed a quality model that evaluates the documentation of Enterprise Resource Planning (ERP) software. The model aims to investigate different quality attributes of the documentations. These attributes are linked either to content or to structures. Main purpose is to ensure readability and completeness of the documentation. The model has been experimented with in the open-source ERP systems, however, more case studies are required in order to generalize the results.

Carvalho et al. [38] proposed a tool named Documentation Mining Open-Source Software (DMOSS) for evaluating the quality of software documentation of non-source code information. The tool has been tested on 4 open-source codes. The tool aims to help maintainers in understanding the software.

Theunissen et al. [39] have introduced a model composed of three approaches for software documentation. The model focuses on categorizing software knowledge into 3 types namely acquiring, building, and transferring knowledge. These types highlight the information to be documented based on the stage of the software life cycle. However, the model has not been evaluated.

Rong et al. [40] introduced a new approach named DevDocOps which can be integrated to DevOps in order to automate the process of software documentation. The approach has been implemented and evaluated in telecommunication enterprise and the results were promising.

Krunic [41] have studied the benefits and difficulties of Documentation as Code (DaC) in vehicle software. The author conducted a case study with 150 participants as software engineers. The author concluded the research by providing a model as a guideline for applying DaC and assessing the quality of documentation.

Kazman et al. [42] have introduced a method to architecturally document open-source software. The authors designed a case study to experiment their method on Hadoop Distributed File System (HDFS). However, the results showed that the proosed method had an effect on the project of HDFS.

Righolt et al. [43] have introduced a tool named Code Diary for automatically documenting decisions from SAS source code. Unlike similar tools, the authors claimed that the proposed tool Code Diary aims to produce code documentation for researchers and other audiences. However, no graphical user interface is available for the tool.

AlOmar et al. [44], [45] have proposed a model that acts as a data set for the documentation of refactoring process of the software. The authors conducted an experiment of 5 stages that has the documentation as the last stage. The experiment carried out over 800 open-source java code found in GitHub.

Geist et al. [46] have introduced their approach which employs Machine Learning, in specific, Deep learning in order to re-document legacy software system from their source code the exploitation of the comments found in the source code. The authors developed the tool based on the approach in one of the well-known automotive companies. However, the generalization aspect of the approach remains in maturing process.

Bhatia et al. [47] proposed an automated tool for code documentation that is based on the ontology-driven development. The authors examine the tool with comparison to a manual tool named WCopyfind. The authors reported that the tool can generate documentation in two types which are targeting human and machine audiences.

Bastos et al. [48] have proposed an approach that aims to help orgnisations in documenting the software project development. The approach employs the ontology methodology. The authors evaluated the approach using a questionnaire circulated to 8 postgraduate students as participants. In addition, the authors reported that the results cannot be generalized due to the low number of participants.

IV. Conclusion

In this paper, a systematic review of the topic of software documentation has been conducted. Budgen et al. and Kitchenham [20], [21] methodology of carrying out systematic review was employed in this research. The main purpose of this research was to investigate the trends and issues related to software documentation in the last 10 years. An important remark is that Software documentation plays a significant role in quality assurance of software [49]. Therefore, it was essential to investigate the research trends in the topic and the related issues. The main conclusions can be summarized in the following points:

1) Software documentation has not been sufficiently considered as a research interest in the last 10 years.

2) There is a collective recognition that documentation is a difficult process and has many problems. In addition, maintenance teams are the effected party with poor documentation.

3) Three types of documentation can be deemed. First is the initial documentation, for example SRS documents. Second are the ongoing documentations, for example TODO lists. Third is the final documentation which includes user manuals.

4) Despite the model being followed in the documentation process, the majority of found primary studies considered the quality of software documentations.

5) Most tools for software documentations are focused on reverse documentations from the source code. This might lead to the loss of the lessons learned and decisions made as they were not previously documented.

6) Automated tools for documentation are in high demand.

7) Despite the importance of software documentation in the quality of software, developers tend not to pay attention to it.

Future research endeavors will focus on developing a comprehensive, standardized model for software documentation that exhibits broad applicability across diverse software systems. Furthermore, this investigation will delve into the identification of ambiguities within existing software documentation typologies and establish interconnections between these categories to facilitate seamless transitions based on the specific developmental phase.

ACKNOWLEDGMENT

The author wishes to express profound appreciation to his family and friends for their unwavering support during the course of this project. Their encouragement and steadfast faith in the author's abilities served as a wellspring of motivation.

The author is also indebted to Umm Al Qura University for providing indispensable resources and cultivating an intellectually stimulating academic environment. These factors significantly contributed to the successful culmination of this research endeavor.

REFERENCES

[1] I. Sommerville, “Software documentation,” Softw. Eng., vol. 2, pp. 143–154, 2001.

[2] I. Sommerville, Software Engineering, 6th edition. Harlow, England ; New York: Addison Wesley, 2000.

[3] I. Sommerville, “Systems engineering for software engineers,” Ann. Softw. Eng., vol. 6, no. 1/4, pp. 111–129, 1998, doi: 10.1023/A:1018901230131.

[4] R. Ries, “IEEE standard for software user documentation,” in International conference on professional communication, communication across the sea: North American and European practices, IEEE, 1990, pp. 66–68. Accessed: Apr. 25, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/111154/

[5] T. T. Barker, Perspectives on Software Documentation: Inquiries and Innovations. Routledge, 2020.

[6] T. C. Lethbridge, J. Singer, and A. Forward, “How software engineers use documentation: The state of the practice,” IEEE Softw., vol. 20, no. 6, pp. 35–39, 2003.

[7] J. Santos and F. F. Correia, “A Review of Pattern Languages for Software Documentation,” in Proceedings of the European Conference on Pattern Languages of Programs 2020, Virtual Event Germany: ACM, Jul. 2020, pp. 1–14. doi: 10.1145/3424771.3424786.

[8] A. Aguiar and G. David, “Patterns for Effectively Documenting Frameworks,” Trans. Pattern Lang. Program. II, vol. 6510, pp. 79–124, 2011, doi: 10.1007/978-3-642-19432-0_5.

[9] E. Aghajani et al., “Software documentation issues unveiled,” in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, 2019, pp. 1199–1210. Accessed: Apr. 24, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8811931/

[10] P. Naur and B. Randell, Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO. 1969. Accessed: Apr. 24, 2024. [Online]. Available: https://dl.acm.org/doi/abs/10.5555/1102020

[11] A. Rüping, Agile Documentation: A Pattern Guide to Producing Lightweight Documents for Software Projects. John Wiley & Sons, 2005.

[12] D. Falessi, G. Cantone, and P. Kruchten, “Value-based design decision rationale documentation: Principles and empirical feasibility study,” in Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), IEEE, 2008, pp. 189–198. Accessed: Apr. 25, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4459157/

[13] D. Falessi, G. Cantone, and M. Becker, “Documenting design decision rationale to improve individual and team design decision making: an experimental evaluation,” in Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, Rio de Janeiro Brazil: ACM, Sep. 2006, pp. 134–143. doi: 10.1145/1159733.1159755.

[14] I. Sommerville, Software Engineering, 10th edition. Boston: Pearson, 2015.

[15] G. Uddin and M. P. Robillard, “How API documentation fails,” Ieee Softw., vol. 32, no. 4, pp. 68–75, 2015.

[16] S. C. B. De Souza, N. Anquetil, and K. M. De Oliveira, “A study of the documentation essential to software maintenance,” in Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information, Coventry United Kingdom: ACM, Sep. 2005, pp. 68–75. doi: 10.1145/1085313.1085331.

[17] M. Kajko-Mattsson, “A Survey of Documentation Practice within Corrective Maintenance,” Empir. Softw. Eng., vol. 10, no. 1, pp. 31–55, Jan. 2005, doi: 10.1023/B:LIDA.0000048322.42751.ca.

[18] A. Forward and T. C. Lethbridge, “The relevance of software documentation, tools and technologies: a survey,” in Proceedings of the 2002 ACM symposium on Document engineering, McLean Virginia USA: ACM, Nov. 2002, pp. 26–33. doi: 10.1145/585058.585065.

[19] E. Aghajani et al., “Software documentation: the practitioners’ perspective,” in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, Seoul South Korea: ACM, Jun. 2020, pp. 590–601. doi: 10.1145/3377811.3380405.

[20] D. Budgen and P. Brereton, “Performing systematic literature reviews in software engineering,” in Proceedings of the 28th international conference on Software engineering, 2006, pp. 1051–1052.

[21] B. Kitchenham, "Procedure for undertaking systematic reviews," Comput. Sci. Depart-Ment Keele Univ. TRISE-0401 Natl. ICT Aust. Ltd 0400011T 1 Jt. Tech. Rep., 2004.

[22] "Saudi Digital Library (SDL)." Accessed: Apr. 26, 2024. [Online]. Available: https://sdl.edu.sa/SDLPortal/Publishers.aspx

[23] M. P. Robillard and R. DeLine, "A field study of API learning obstacles," Empir. Softw. Eng., vol. 16, no. 6, pp. 703–732, Dec. 2011, doi: 10.1007/s10664-010-9150-8.

[24] G. Garousi, V. Garousi-Yusifoğlu, G. Ruhe, J. Zhi, M. Moussavi, and B. Smith, "Usage and usefulness of technical software documentation: An industrial case study," Inf. Softw. Technol., vol. 57, pp. 664–682, 2015.

[25] J. D. Arthur and K. T. Stevens, "Document quality indicators: A framework for assessing documentation adequacy," J. Softw. Maint. Res. Pract., vol. 4, no. 3, pp. 129–142, Sep. 1992, doi: 10.1002/smr.4360040303.

[26] A. Dautovic, "Automatic assessment of software documentation quality," in 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), IEEE, 2011, pp. 665–669. Accessed: May 10, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6100151/

[27] J.-C. Chen and S.-J. Huang, "An empirical analysis of the impact of software development problem factors on software maintainability," J. Syst. Softw., vol. 82, no. 6, pp. 981–992, 2009.

[28] B. Dagenais and M. P. Robillard, "Creating and evolving developer documentation: understanding the decisions of open source contributors," in Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering, Santa Fe New Mexico USA: ACM, Nov. 2010, pp. 127–136. doi: 10.1145/1882291.1882312.

[29] S. Rai, R. C. Belwal, and A. Gupta, "A Review on Source Code Documentation," ACM Trans. Intell. Syst. Technol., vol. 13, no. 5, pp. 1–44, Oct. 2022, doi: 10.1145/3519312.

[30] M. Meng, S. Steinhardt, and A. Schubert, "Application Programming Interface Documentation: What Do Software Developers Want?," J. Tech. Writ. Commun., vol. 48, no. 3, pp. 295–330, Jul. 2018, doi: 10.1177/0047281617721853.

[31] J. Zhi, V. Garousi-Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, benefits and quality of software development documentation: A systematic mapping," J. Syst. Softw., vol. 99, pp. 175–198, 2015.

[32] F. Bachmann et al., "Software architecture documentation in practice: Documenting architectural layers," 2000, Accessed: Apr. 24, 2024. [Online]. Available: https://www.getforms.org/forms/forms-pdf/5022.pdf

[33] P. C. Véras, E. Villani, A. M. Ambrosio, M. Vieira, and H. Madeira, "A benchmarking process to assess software requirements documentation for space applications," J. Syst. Softw., vol. 100, pp. 103–116, Feb. 2015, doi: 10.1016/j.jss.2014.10.054.

[34] M. Farwick, C. M. Schweda, R. Breu, and I. Hanschke, "A situational method for semi-automated Enterprise Architecture Documentation," Softw. Syst. Model., vol. 15, no. 2, pp. 397–426, May 2016, doi: 10.1007/s10270-014-0407-3.

[35] M. Hauder, F. Matthes, and S. Roth, "Challenges for Automated Enterprise Architecture Documentation," in Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation, vol. 131, S. Aier, M. Ekstedt, F. Matthes, E. Proper, and J. L. Sanz, Eds., in Lecture Notes in Business Information Processing, vol. 131. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 21–39. doi: 10.1007/978-3-642-34163-2_2.

[36] A. Mathrani, S. Wickramasinghe, and N. P. Jayamaha, "An evaluation of documentation requirements for ISO 9001 compliance in scrum projects," TQM J., vol. 34, no. 5, pp. 901–921, 2022.

[37] L. Aversano, D. Guardabascio, and M. Tortorella, "Analysis of the documentation of ERP software projects," Procedia Comput. Sci., vol. 121, pp. 423–430, 2017.

[38] N. R. Carvalho, A. Simoes, and J. J. Almeida, "DMOSS: Open source software documentation assessment," Comput. Sci. Inf. Syst., vol. 11, no. 4, pp. 1197–1207, 2014.

[39] T. Theunissen, S. Hoppenbrouwers, and S. Overbeek, "Approaches for documentation in continuous software development," Complex Syst. Inform. Model. Q., no. 32, pp. 1–27, 2022.

[40] G. Rong, Z. Jin, H. Zhang, Y. Zhang, W. Ye, and D. Shao, "DevDocOps: Enabling continuous documentation in alignment with DevOps," Softw. Pract. Exp., vol. 50, no. 3, pp. 210–226, 2020, doi: 10.1002/spe.2770.

[41] M. V. Krunic, "Documentation as code in automotive system/software engineering," Elektron. Ir Elektrotechnika, vol. 29, no. 4, pp. 61–75, 2023.

[42] R. Kazman, D. Goldenson, I. Monarch, W. Nichols, and G. Valetto, "Evaluating the effects of architectural documentation: A case study of a large scale open source project," IEEE Trans. Softw. Eng., vol. 42, no. 3, pp. 220–260, 2015.

[43] C. H. Righolt, B. A. Monchka, and S. M. Mahmud, "From source code to publication: Code Diary, an automatic documentation parser for SAS," SoftwareX, vol. 7, pp. 222–225, Jan. 2018, doi: 10.1016/j.softx.2018.07.002.

[44] E. Abdullah AlOmar, A. Peruma, M. Wiem Mkaouer, C. Newman, A. Ouni, and M. Kessentini, "How We Refactor and How We Document it? On the Use of Supervised Machine Learning Algorithms to Classify Refactoring Documentation," ArXiv E-Prints, p. arXiv-2010, 2020.

[45] E. Abdullah AlOmar et al., "On the Documentation of Refactoring Types," ArXiv E-Prints, p. arXiv-2112, 2021.

[46] V. Geist, M. Moser, J. Pichler, S. Beyer, and M. Pinzger, "Leveraging machine learning for software redocumentation," in 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2020, pp. 622–626. Accessed: May 12, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9054838/

[47] M. P. S. Bhatia, A. Kumar, and R. Beniwal, "Ontology Driven Software Development for Automated Documentation.," Webology, vol. 15, no. 2, 2018, Accessed: May 12, 2024. [Online]. Available: https://www.researchgate.net/profile/Rohit-Beniwal-5/publication/331489088_Ontology_Driven_Software_Development_for_Automated_Documentation/links/5c7d1c90458515831f81987c/Ontology-Driven-Software-Development-for-Automated-Documentation.pdf

[48] E. C. Bastos, M. P. Barcellos, and R. De Almeida Falbo, "Using Semantic Documentation to Support Software Project Management," J. Data Semant., vol. 7, no. 2, pp. 107–132, Jun. 2018, doi: 10.1007/s13740-018-0089-z.

[49] J. E. Tyler, "Asset management the track towards quality documentation," Rec. Manag. J., vol. 27, no. 3, pp. 302–317, Jan. 2017, doi: 10.1108/RMJ-11-2015-0039.

APPENDIX A

PRIMARY STUDIES

Item	Bibliography	Sources
1.	T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," IEEE Softw., vol. 20, no. 6, pp. 35–39, 2003.	IEEE
2.	J. Santos and F. F. Correia, "A Review of Pattern Languages for Software Documentation," in Proceedings of the European Conference on Pattern Languages of Programs 2020, Virtual Event Germany: ACM, Jul. 2020, pp. 1–14. doi: 10.1145/3424771.3424786.	ACM
3.	A. Aguiar and G. David, "Patterns for Effectively Documenting Frameworks," Trans. Pattern Lang. Program. II, vol. 6510, pp. 79–124, 2011, doi: 10.1007/978-3-642-19432-0_5.	Other Publishers
4.	E. Aghajani et al., "Software documentation issues unveiled," in 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), IEEE, 2019, pp. 1199–1210. Accessed: Apr. 24, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8811931/	IEEE
5.	D. Falessi, G. Cantone, and P. Kruchten, "Value-based design decision rationale documentation: Principles and empirical feasibility study," in Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008), IEEE, 2008, pp. 189–198. Accessed: Apr. 25, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4459157/	IEEE
6.	D. Falessi, G. Cantone, and M. Becker, "Documenting design decision rationale to improve individual and team design decision making: an experimental evaluation," in Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, Rio de Janeiro Brazil: ACM, Sep. 2006, pp. 134–143. doi: 10.1145/1159733.1159755.	ACM
7.	G. Uddin and M. P. Robillard, "How API documentation fails," Ieee Softw., vol. 32, no. 4, pp. 68–75, 2015.	Other Publishers
8.	S. C. B. De Souza, N. Anquetil, and K. M. De Oliveira, "A study of the documentation essential to software maintenance," in Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information, Coventry United Kingdom: ACM, Sep. 2005, pp. 68–75. doi: 10.1145/1085313.1085331.	ACM
9.	M. Kajko-Mattsson, "A Survey of Documentation Practice within Corrective Maintenance," Empir. Softw. Eng., vol. 10, no. 1, pp. 31–55, Jan. 2005, doi: 10.1023/B:LIDA.0000048322.42751.ca.	Other Publishers
10.	A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in Proceedings of the 2002 ACM symposium on Document engineering, McLean Virginia USA: ACM, Nov. 2002, pp. 26–33. doi: 10.1145/585058.585065.	ACM
11.	E. Aghajani et al., "Software documentation: the practitioners' perspective," in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, Seoul South Korea: ACM, Jun. 2020, pp. 590–601. doi: 10.1145/3377811.3380405.	ACM
12.	S. Rai, R. C. Belwal, and A. Gupta, "A Review on Source Code Documentation," ACM Trans. Intell. Syst. Technol., vol. 13, no. 5, pp. 1–44, Oct. 2022, doi: 10.1145/3519312.	ACM
13.	M. Meng, S. Steinhardt, and A. Schubert, "Application Programming Interface Documentation: What Do Software Developers Want?," J. Tech. Writ. Commun., vol. 48, no. 3, pp. 295–330, Jul. 2018, doi: 10.1177/0047281617721853.	Other Publishers
14.	J. Zhi, V. Garousi-Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe, "Cost, benefits and quality of software development documentation: A systematic mapping," J. Syst. Softw., vol. 99, pp. 175–198, 2015.	Other Publishers
15.	F. Bachmann et al., "Software architecture documentation in practice: Documenting architectural layers," 2000, Accessed: Apr. 24, 2024. [Online]. Available: https://www.getforms.org/forms/forms-pdf/5022.pdf	Other Publishers
16.	P. C. Véras, E. Villani, A. M. Ambrosio, M. Vieira, and H. Madeira, "A benchmarking process to assess software requirements documentation for space applications," J. Syst. Softw., vol. 100, pp. 103–116, Feb. 2015, doi: 10.1016/j.jss.2014.10.054.	Other Publishers
17.	M. Farwick, C. M. Schweda, R. Breu, and I. Hanschke, "A situational method for semi-automated Enterprise Architecture Documentation," Softw. Syst. Model., vol. 15, no. 2, pp. 397–426, May 2016, doi: 10.1007/s10270-014-0407-3.	Other Publishers
18.	A. Mathrani, S. Wickramasinghe, and N. P. Jayamaha, "An evaluation of documentation requirements for ISO 9001 compliance in scrum projects," TQM J., vol. 34, no. 5, pp. 901–921, 2022.	Other Publishers
19.	L. Aversano, D. Guardabascio, and M. Tortorella, "Analysis of the documentation of ERP software projects," Procedia Comput. Sci., vol. 121, pp. 423–430, 2017.	Other Publishers
20.	N. R. Carvalho, A. Simoes, and J. J. Almeida, "DMOSS: Open source software documentation assessment," Comput. Sci. Inf. Syst., vol. 11, no. 4, pp. 1197–1207, 2014.	Other Publishers
21.	T. Theunissen, S. Hoppenbrouwers, and S. Overbeek, "Approaches for documentation in continuous software development," Complex Syst. Inform. Model. Q., no. 32, pp. 1–27, 2022.	Other Publishers
22.	G. Rong, Z. Jin, H. Zhang, Y. Zhang, W. Ye, and D. Shao, "DevDocOps: Enabling continuous documentation in alignment with DevOps," Softw. Pract. Exp., vol. 50, no. 3, pp. 210–226, 2020, doi: 10.1002/spe.2770.	Other Publishers
23.	M. V. Krunic, "Documentation as code in automotive system/software engineering," Elektron. Ir Elektrotechnika, vol. 29, no. 4, pp. 61–75, 2023.	Other Publishers
24.	R. Kazman, D. Goldenson, I. Monarch, W. Nichols, and G. Valetto, "Evaluating the effects of architectural documentation: A case study of a large scale open source project," IEEE Trans. Softw. Eng., vol. 42, no. 3, pp. 220–260, 2015.	IEEE
25.	C. H. Righolt, B. A. Monchka, and S. M. Mahmud, "From source code to publication: Code Diary, an automatic documentation parser for SAS," SoftwareX, vol. 7, pp. 222–225, Jan. 2018, doi: 10.1016/j.softx.2018.07.002.	Other Publishers
26.	E. Abdullah AlOmar, A. Peruma, M. Wiem Mkaouer, C. Newman, A. Ouni, and M. Kessentini, "How We Refactor and How We Document it? On the Use of Supervised Machine Learning Algorithms to Classify Refactoring Documentation," ArXiv E-Prints, p. arXiv-2010, 2020.	Other Publishers
27.	E. Abdullah AlOmar et al., "On the Documentation of Refactoring Types," ArXiv E-Prints, p. arXiv-2112, 2021.	IEEE
28.	V. Geist, M. Moser, J. Pichler, S. Beyer, and M. Pinzger, "Leveraging machine learning for software redocumentation," in 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 2020, pp. 622–626. Accessed: May 12, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9054838/	ACM
29.	M. P. S. Bhatia, A. Kumar, and R. Beniwal, "Ontology Driven Software Development for Automated Documentation.," Webology, vol. 15, no. 2, 2018, Accessed: May 12, 2024. [Online]. Available: https://www.researchgate.net/profile/Rohit-Beniwal-5/publication/331489088_Ontology_Driven_Software_Development_for_Automated_Documentation/links/5c7d1c90458515831f81987c/Ontology-Driven-Software-Development-for-Automated-Documentation.pdf	Other Publishers