Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 121 (2017) 423–430

Procedia Computer Science

www.elsevier.com/locate/procedia

CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS / ProjMAN / HCist 2017, 8-10 November 2017, Barcelona, Spain

Analysis of the Documentation of ERP Software Projects

Lerina Aversano*, Daniela Guardabascio, Maria Tortorella

Department of Engineering, University of Sannio
Via Traiano, 82100, Benevento

Abstract

Software documentation is a basic component of the software development process and it is very important in all the phases of a software system life cycle. It is plays a very important role from the point of view of both the software engineer and user. Software documentation usually includes textual documentation required by the Software engineering standards, API documentation, Wiki pages and source code comments. Surveys and studies indicate that the documentation is not always available and, if available, only partially addresses the developers' needs, as it is often wrong, incomplete, out-of-date and ambiguous. In the context of ERP – Enterprise Resource Planning, the relevance of the software documentation is even more important due to the complexity of such a kind of software systems and the strategic role they have within operative organizations. This paper focuses on the quality assessment of the documentation of ERP open source systems with the aim of understanding if they include high quality documentation for adequately support anyone want to adopt them and/or executing maintenance activities. Specifically, a quality model is defined and its application to three Open source software system is performed.

© 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies.

Keywords: Software documentation, ERP Open source software, Documentation maintenance, Documentation Quality, Software metrics.

* Corresponding author. Tel.: +39 0824 305551; fax: +39 0824 325246.
E-mail address: aversano@unisannio.it

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of the scientific committee of the CENTERIS - International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies.
10.1016/j.procs.2017.11.057

1. Introduction

Software documentation plays a very important role in all the phases of a software system life cycle. It is not only relevant from the software engineering point of view, in all the software development process from the requirement definition to the maintenance activities [3, 7, 20], but also from the user point of view. Actually the user needs to know, with reference how to install and use to a software system, to import and manage data, to elaborate information and so on.

The software is usually documented by the textual documentation required by the software engineering standards IEEE Std 830-1998, IEEE Std 1028-2008, IEEE Std 1063-2001, ISO/IEC 9126:2001, ISO/IEC 25010:2005, ISO/IEC 26514:2008 [11, 12, 13, 14, 15, 16], and so on. This documentation mostly consists of documents aiming to explain the functionalities the software performs, its architecture, how it is structured and implemented, and can be used. It generally includes the following documents: software requirements specifications, software design documents, code, quality and testing documents. Additional documents can exist, formalized as API documentation, Wiki pages. For describing different aspects of the software system. Such a kind of documents can be easily accessed by both developers and users, and can be considered documentation as well. API documentation specifies how software components can be used and interact with each other. Wiki pages allow for web-based visualization and knowledge management. It offers semantic-enhanced search facilities such as filtering, faceting, and graph-like exploration of knowledge. The hyper textual features make this type of documentation easy to find and consult. In addition, source code comments can represent a useful source of information helping to understand the software product [4], and it can be used as documentation as well, once extracted and adequately formatted.

The proposed study uses the term "software documentation" for referring the different types of documents indicated above [2]

For being really useful, software system documentation should include reliable, complete and unambiguous information regarding the software system. In addition, it should have a structure that makes the different documentation section easily recognizable.

The relevance of the documentation is also confirmed in surveys and papers including the results of interviews with software engineers and developers working in organizations. Unfortunately, the results also indicate that the documentation is not always available and it only partially addresses the developers and users' needs, as it is often wrong, incomplete, out-of-date and ambiguous. In addition, while in the case of closed source software, the documentation exists as it is foreseen by the internal quality procedures and is clear and complete [19], it can happen that it is not available for open source systems, where these documents may not exist, or not represent any official documentation.

In the context of ERP – Enterprise Resource Planning, the relevance of the software documentation is even more important due to the complexity of such a kind of software systems and the strategic role they have within operative organizations.

This paper focuses on the evaluation of the quality of the various types of documents that may be useful for understanding a software artefact. It mainly considers the user point of view and analyses both structural and content aspects that a good software system document should include for being effectively used. The content aspects are defined by considering the ERP system context and all the information the documentation of such a kind of system should include for successfully using it. Moreover, the paper applies the defined framework to the documentation of some ERP Open Source Software system, with the aim of understanding if it can be a valuable support to anyone who wants to adopt such a kind of system.

This paper is organized as follows: Section 2 discusses the principal research work. Section 3 describes the proposed framework for evaluating the quality of the documentation of ERP Open Source Software system. Section 4 presents the application of the measurement framework to ERP software system, and subsequent section includes some final considerations.

2. Related work

The literature reports several studies discussing the use that software practitioners make of different kinds of documentations. In different papers, the focus is on the usefulness of the documentation of a software product.

In particular, some of these studies are surveys. Forward and Lethbridge made a survey involving different developers, and presented several documentation attributes, such as document writing style, grammar, level of upgrade, type, format, visibility, etc. [9]. They observed that the documentation content is an important support for communication and should always be useful and serve a purpose. It can even be relevant if it is not updated or inconsistent. The same authors highlighted the general attitudes of software engineering documentation [17]. Some results indicated that various types of abstract documentation are a valid guidance for the maintenance work and that inline comments of the source code are often a good support to assist detailed maintenance work. The study also discussed negative results, such as the fact that multiple types of documents are often out of date or that the documentation is poorly written.

de Souza et al. established in their surveys the importance of each documentation artefacts for fully understanding the software product and executing maintenance activities [4]. The results of this study have also shown that the documentation is often incomplete or out-of-date, and the developers have to use the source code and related comments for fully understanding the software product.

Another aspect coming from the contribution of Kipyegen and Korir regards the little usage of the documentation, and, therefore, the decrease of its efficiency during the software development task [10].

Several other research works focused on the documentation quality.

Arthur and Stevens identified four Document Quality Indicators (DQI) that are attributed to an appropriate documentation [1]: Accuracy, Completeness, Usability and Expandability. As quality is an intangible characteristic, without direct measure, the authors of the cited paper associated each quality characteristic to some factors, and each factor to some quantifiers, whose combination gave rise to quality indicators (DQI) that are quantifiable.

In [18, 22], the following additional documentation quality attributes are proposed: Accuracy, Clarity, Consistency, Readability, Structuring, and Understandability. Indeed, from a conducted survey, it emerges that the typical problems related to the documentation quality deals with unreliable, incomplete or non-existent documentation, not documented changes in the software system and lack of integrity and coherence [22]. In [17], a tool is presented for analysing the quality of software systems, documentation included.

Problems related to the documentation quality have also been encountered by Uddin and Robillard with reference to the API documentation quality [21], and by Diaz-Pace et al. with reference to the Wiki pages [6].

This paper considers all the aspects arising from the previous studies and proposes a quality model for evaluating various types of documentation associated specifically to ERP Open Source Software system.

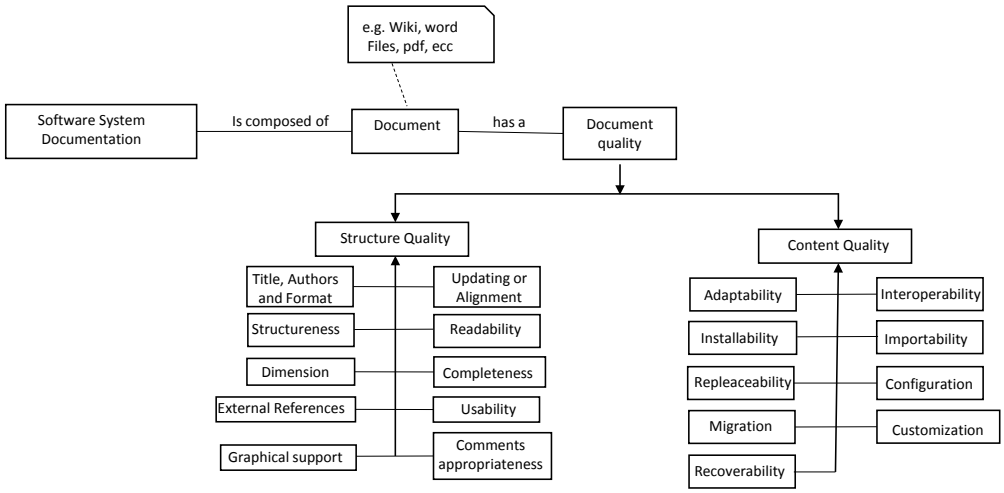

Fig. 1. Documentation evaluation framework

3. A framework for evaluating the quality of ERP software projects

This section describes the proposed approach for assessing and evaluating the quality of ERP software documentation. The approach is metric-based and considers different quality indicators. Figure 1 shows the organization of the proposed framework. It highlights that the documentation is composed of documents that can be of the different kinds, API, Wiki, text and code comments. In addition, the quality of each document is considered and analysed from two different points of view: Structure Quality and Content Quality.

With reference to the Structure Quality the following indicators have been defined for establishing to what extent the documentation is understandable and well structured:

- *Title*, *Authors*, and *Format*, have been consider as general information regarding the document.
- *Structureness*: regarding the textual documentation and aiming at evaluating the quality of the actual structure in terms of number of chapters, sections, sub-sections nesting, document length, and density of tables and figures. A good structure and organization of the document helps to consult it.
- *Dimension:* to analyse the length of the document sentences: sentences that are too long express text that is too difficult to be understood, and sentences too short may not exhaustively express a concept. Cutts asserted that over the whole document, average sentence length should be 15-20 word [5].
- *External references:* to understand if references to documents external to the documentation or to source code exist to better support the comprehension of software engineers and users.
- *Graphical Support:* to verify the availability of visual aids (figures and tables) facilitating the understandability of the text. The citations of the visual aids within the text and existence of clear captions are verified. It may occur that figures or tables not included in the document are indexed, in this case the score is greater than 1.
- *Updating or alignment:* to verify whether the documentation is updated with reference to the project release it refers to.
- *Readability:* to examine if the sentences express clear and understandable concepts. It is evaluated trough the Flesch readability test (Flesch Reading Ease), designed for indicating how difficult is to be understood an English reading. The index score ranges from 0 to 100 and a value equal to 50 is considered adequate as it is associated with the 10th to 12th grade school level; while higher scores refers to material that is easier to read and lower marks indicate documents that are more difficult to read [8].
- *Completeness*: to evaluate the completeness level of the documentation with reference to the source code; in particular, these quality indicator checks whether the documentation describes all of the source code items (packages, classes, methods).
- *Usability*: concerning API and Wiki and assessing the organization from a usability point of view. Aspects that will be investigated are the fragmentation of the concepts within web pages, and the size of the Java Doc and Wiki.
- *Comments appropriateness*: to estimate the density of comments in the code. A high density of comments helps the developer in the comprehension of source code.

Most of the metrics above can be evaluated with reference to the API, Wiki, code comments and text documentation, while others such as the Graphical Support are evaluated just with reference to Documentation; Usability is evaluated with reference to Wiki and JavaDoc; finally, the Appropriateness of the comments is evaluated with reference to the code comments.

The evaluation of these metrics requires the application of NLP, Information Extraction and Information Retrieval techniques in order to make an objective analysis and not a subjective one. Most of the formula that have been applied for their evaluation is included in [2].

The section of the content quality model considers the set of quality indicators identified to evaluate to what extent the documentation reflects the current state of the software system. This section of framework has been defined by considering the main characteristics of ERP systems, as it will be applied to such a kind of systems. The indicators are:

- *Adaptability:* to evaluate if in the documentation adequately indicates the supported operating system and database management systems.
- *Installability:* to verify whether the documentation adequately describes the installation process.

- *Replaceability:* to evaluate is a description exists in the documentation with reference to the possibility of replacing the ERP system with a different one.
- *Migration:* to verify whether the documentation adequately describes the migration process referring the generated reports.
- *Recoverability:* to evaluate if the functionality for the transactions management are adequately described in the documentation.
- *Interoperability:* to evaluate how the documentation discusses the web-services support.
- *Importability:* to evaluate if the documentation adequately describes the formats for importing data.
- *Configuration:* to analyze if all the ERP system configuration functionality, such as Wizard, charts of accounts, supported languages, tax category management, Multicurrency coverage, setup, are well described.
- *Customisation:* to verify if the customization functionality regarding the full system, user interface, workflows and reports is adequately described.

The first three columns of Tables 1 contain the list of the attributes, questions and related metrics that the quality model considers. Each question aims at evaluation if a set of aspects are described in the documentation for verifying if the related attribute is possessed by the software system. In particular, it can be answered with the following values: *Def*, indicating that the information required by the question is clearly and completely defined; *NDef*, pointing out that the analyzed documentation does not consider the specific aspect; *Part*, indicating that the aspect indicated in the question is only partial addressed; *NCl*, indicating that the documentation does not clearly describe the required information.

Table 1. Content quality attribute evaluation for Openbravo, Compiere, Adempiere

Content Quality attribute	Question	Abbr	OpenBravo	Compiere	Adempiere
Adaptability	Are the supported operating systems indicated in the documentation?	SupOS	Def	Def	Def
	Are the supported DBMS systems indicated in the documentation?	SupDB	Def	Def	Def
Installability	Is available the installation manual?	InstMan	Def	Def	Def
	Is there the manual multi-language installing?	LanMan	Def	NDef	NDef
Replaceability	Is the functionality for data backup adequately described?	CrBkp	Def	Def	Def
	Is there functionality for restoring the backup data adequately described?	ReBkp	Def	Def	Def
	Is the functionality for backup services adequately described?	BkpServ	Def	NCl	NDef
	Are the migration processes between versions adequately described?	MigDoc	Def	Part	Def
	Are the automatic migration tools adequately described?	AutoMig	Part	Part	Part
	Is the database migration adequately described?	DBDoc	Def	Part	Def
Migration	Are the formats for the reporting adequately described?	StdExp	Part	Def	Def
Recoverability	The tools for the transactions management are adequately described?	TnsMgm	NCl	Def	NDef
Interoperability	Is the web-services support adequately described in the documentation?	WSS	Def	Def	Part
Importability	Are the formats for importing data adequately described?	StdImp	Def	Def	Def
Configuration	Are the wizard for configuring the system adequately described?	ConfWiz	Def	Def	Def
	Are the charts of accounts adequately described?	ChtAcc	Def	Def	Def
	Are the supported language adequately indicated in the documentation?	LanPck	Part	Part	Part
	Is the tax category management adequately described?	TaxConf	Def	Def	Def
	Is the Multicurrency coverage index adequately indicated?	CurrInd	Part	NCl	Def
	Is the starting setup adequately described in the documentation?	SetDoc	Def	Def	Def
	Is the language configuration adequately described in the documentation?	LanDoc	Def	Def	NDef
	Is the tax category configuration adequately described?	TaxDoc	Def	Def	NDef
	Is the configuration of the charts of accounts adequately described?	AccDoc	Part	Def	Def
Customisation	Is the functionality for installing an extension of the user interface adequately described?	ExtInst	Def	Def	NDef
	Is the functionality for creating a new module of the user interface adequately described?	NewMod	Def	Def	Def
	Is the functionality for customizing the user interface adequately described?	UICust	Def	Def	Def
	Is the functionality for creating a new workflow adequately described?	WFCust	Def	Def	Def
	Is the functionality for creating a new reports adequately described?	RepCust	Def	Def	Def
	Is the customization process adequately described in the documentation?	CustDoc	Def	Part	Part

4. Case Study

This section presents the results obtained from applying the described quality model to three ERP FlOSS systems: Openbravo (http://www.openbravo.com/), Compiere (http://www.compiere.com/) and ADempiere (www.adempiere.org/). The first analysis that was performed regarded the documentation composition. Table 2 lists all the documents composing the documentation of the systems and assigns to each of them an identification code. These codes are then used in Table 3 for introducing the values achieved from the evaluation of the quality structure

Table 2. Documentation composition of Openbravo, Compiere, Adempiere

Openbravo		**Compiere**		**ADempiere**	
Document Title	Document Code	Document Title	Document Code	Document Title	Document Code
General Information	D1	Release Notes and Documentation	D7	Preface	D12
System Administration Guide	D2	Admin Guide	D8	User Guide	D13
Quick Guide	D3	Users Guide	D9	Business Functionality Guide	D14
User Guide	D4	Technical Guide	D10	System Administrator's Guide	D15
Developers Guide	D5	Database installation guides	D11	Developer's Guide	D16
Localization guide	D6				

Table 3. Evaluation of the documentation structure of Openbravo, Compiere, ADempiere

	Openbravo						**Compiere**					**ADempiere**				
	D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11	D12	D13	D14	D15	D16
Title and Authors																
% Doc with title	80%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
% Doc with authors	0%	0%	0%	0%	0%	0%	17%	83%	100%	100%	0%	0%	0%	0%	0%	0%
Structureness																
# SubSections of Level 1	5	5	0	9	6	0	6	6	25	13	4	8	2	3	8	6
AVG Num of SubSections of Level 2	1	3.6	0	0	0	0	0	0	0	0	0	1.13	4	2.67	2.75	4.5
# Doc	5	23	1	9	6	1	6	6	25	13	4	17	8	8	25	27
% Doc with paragraphs	60%	48%	100%	100%	100%	100%	33%	83%	96%	100%	100%	100%	100%	100%	100%	100%
Dimension																
Doc Dimension	125	1711	51	1161	224	330	13481	1548	11259	1256	18114	1160	3554	1753	3555	4194
External References																
% Doc with external links	60%	100%	0%	100%	100%	100%	33%	33%	72%	15%	100%	100%	88%	100%	100%	100%
% Doc referring source code	0%	0%	0%	0%	100%	0%	0%	0%	0%	46%	0%	0%	38%	63%	3%	24%
Graphical Support																
% Doc with figures	0%	39%	100%	100%	17%	100%	100%	50%	92%	38%	75%	18%	75%	75%	48%	44%
AVG Figures per Doc	0	1.65	8	1.56	0.17	1	7.67	2.5	6.88	3.15	28.3	0.35	10	7.63	2.08	2.48
# Figures	0	38	8	14	1	1	46	15	172	41	113	6	80	61	52	67
% of document with tables	0%	4%	0%	0%	0%	0%	17%	0%	24%	8%	0%	0%	13%	38%	8%	11%
AVG Tables per Doc	0	0.17	0	0	0	0	0.67	0	0.64	0.15	0	0	0.75	1.38	0.2	0.11
# Tables	0	4	0	0	0	0	4	0	16	2	0	0	6	11	5	3
% Doc with diagrams	0%	0%	0%	78%	17%	0%	0%	0%	12%	0%	0%	6%	0%	0%	6%	11%
Updating																
% updated Doc	60%		100%	100%	100%	100%	50%	100%	0%	0%	0%	6%	0%	0%	0%	11%
Readability																

AVG Flesch index for Level 1	12.8	45.8	52.25	45.23	43.05	28.05	56.69	41.71	59.75	44.5	48.1	52.04	57.9	50.71	48	46.9
AVG sentences length in Level 1	10.5	8.98	15	15.28	11.33	14.77	8.68	8.08	8.49	11.9	7.74	10.62	8.15	8.85	9.24	8.22

metrics. In particular, there is a column of Table 3 for each document identified through an identification code.

Observing the data collected in Table 2, it emerges that all the documents of the three systems include a title, while just Compiere reveals the authors of the documents. Even the dimension of the documents is significantly higher in Compiere, while Openbravo has the most reduced documentation. Similarly, the structureness in terms of number of subsections and paragraphs is more elevated in Compiere and ADempiere. Looking at the data referring the Graphical Support, it is possible to note that the values regarding Compiere and ADempiere are again higher than the one referring to Openbravo, with the exception of some documents in this last system. The average values of the Flesch index and sentences length highlight that Readability is similar in all the documents of all the systems. In fact, all the systems documentations include some documents that are more readable and other ones that are less. The data above indicate that the poorest documentation is the one of Openbravo, even if indicator Updating points out that it is the only one that has updated documentation. In addition, the results included in Table 4 indicate that the quality of the API is higher in Openbravo, with the higher values of updating, completeness and density, while the fragmentation in terms of classes and packages description is quite high. Table 4 also shows that the best quality of the inline comments can be found ADempiere, while source code comments have not been found in Compiere.

Table 4. Evaluation of the quality of API and inline comments in Openbravo, Compiere, ADempiere

		Openbravo	**Compiere**	**ADempiere**
API	Used Formulas			
API Updating		Yes	NO (3.1)	NO (3.5.2a)
API Completeness	(# packages + # classes in API)/(# packages + # implemented classes)	1.022161742		0.5897
JavaDocDensity	# classes / JavaDoc MB	20.49235993	14.30051813	13.43962585
Info Framentation	# classes / #WebPages	0.915434206	0.242238069	0.235983576
Source Code comments				
Comments compleness	(# classes + # commnented methods) / (# classes + # implemneted methods)	0.245716812	Absent	0.74707081
AVG sentences lenght		11.8994	Absent	6.214280982
Flesch index		52,726	Absent	50.34706473
Commnets density	Lines of Comments / LinesOfCode	0,073	Absent	0.212496169

The last three columns of Table 1 summarize the evaluation of the quality content and compare the content quality attributes of the selected projects. The results indicate that the documentation of the three systems covers the large part of the defined quality content attributes. Indeed, most of the attributes assume the *Def* value, indicating that the related topic is adequately described in the documentation. Nevertheless, Table 1 also highlights some points to be improved for each system. As an example, the manual for the multi-language nstalling and the backup services are not completely described in Compiere and Adempiere, while they are adequately defined in Openbravo. Moreover, the migration process between versions and the database migration are not completely described in Compiere, while the presence of supporting tools for this process is just partially described in the documentation of all the systems. The Recoverability is only addressed in Compiere, and it is not clear in Openbravo. The documentation of ADempiere partially describes the web-service support. Other comments and indications can be obtained from Table 1. All this observation can be addressed for improving the lacking aspects in the documentation systems.

5. Conclusions

This paper proposes a quality model for the evaluation of ERP Open Source Software systems Documentation and an operational framework to support the model.

The proposed quality model considers both the quality of the documentation structure and the one of the content of the considered documents.

Different attributes have been considered in relation to both these aspects, such as the Readability of the document, the Completeness, and the Updating level. Moreover, the description of the main required indicators doe ERP system

has been investigated. The applicability of the model was analysed through the study of the documentation of three open Source ERP systems, and allowed the identification of specific point to address for improving the overall system documentation.

Future work will consider a refinement of the quality model with the introduction of additional needed metrics considering grammatical correctness of documentation, ambiguity of the text, duplication of arguments, and the analysis of more case studies. Indeed, a larger base of software systems should be measured to increase the practical relevance of the achieved results.

References

1. Arthur, J.D., Stevens, K. T., 1989, Assessing the adequacy of documentation through document quality indicators. Conference on SoftwareMaintenance, IEEE comp. soc. press, pp. 40-49.
2. Aversano L., Guardabascio D., Tortorella M., 2017, Evaluating the Quality of the Documentation of Open Source Software. Conference on Evaluation of Novel Approaches to Software Engineering.
3. Chomas, V. S., Saini, J. R., 2015, Software Template for Evaluating and Scoring Software Project Documentations. Int. Journal of Computer Applications, 116(1).
4. Cozzetti de Souza, S., Anquetil, N., de Oliveira, K.M., 2005. A Study of the Documentation Essential to Software Maintenance. 23rd Annual Int. Conference on Design of communication: documenting & designing for pervasive (SIGDOC '05), ACM press, pp.68-75.
5. Cutts, M., 2013. Oxford guide to plain English. OUP Oxford.
6. Diaz-Pace, J.A., Nicoletti, M., Schettino, S., Grisando, R., 2014. A recommender system for technical software documentation in Wikis. Biennial Congress of Argentina (ARGENCON), IEEE comp. soc. press, pp. 393-398.
7. Garousi, G., Garousi, V., Moussavi, M., Ruhe G., Smith, B., 2013. Evaluating usage and quality of technical software documentation: an empirical study. 17th Int. Conference on Evaluation and Assessment in Software Engineering (EASE). ACM, pp. 24-35.
8. Flesch, R., 2016. How to Write Plain English. University of Canterbury.
9. Forward, A., Lethbridge, T. C., 2002. The relevance of software documentation, tools and technologies: a survey. Symposium on Document engineering, ACM press, pp. 26-33.
10. Kipyegen, N. J., Korir, W. P. K., 2013. Importance of Software Documentation. IJCSI Int. Journal of Computer Science, 10(5), pp.1694-0784
11. IEEE Std 830-1998. 1998. IEEE Recommended Practice for Software Requirements Specifications.
12. IEEE Std 1028-2008. 2008. IEEE Standard for Software Reviews and Audits.
13. IEEE Std 1063-2001. 2001. IEEE Standard for Software User Documentation.
14. ISO/IEC 9126:2001. 2001. Software engineering – Product quality.
15. ISO/IEC 25010:2005. 2005. Software engineering – Software product Quality Requirements and Evaluation (SquaRE).
16. ISO/IEC 26514:2008. 2008. Systems and software engineering - Requirements for designers and developers of user documentation.
17. Lethbridge, T. C., Singer, J., Forward, A., 2003. How software engineers use documentation: The state of the practice. IEEE Software, 20(6), pp. 35-39.
18. Plösch, R., Dautovic, A., Saft, M., 2014, The Value of Software Documentation Quality. 14th Int. Conference on Quality Software. IEEE comp. soc. press, pp. 333-342.
19. Satzinger, J. W., Jackson, R. B., Burd, S. D., 2000. System Analysis and Design in a Changing World, Thomson Learning.
20. Sommerville I, 2005. Software Documentation. Software Engineering, Volume 2: The Supporting Process, pp. 143-154.
21. Uddin, G., Robillard, M. P., 2015. How API documentation fails. IEEE Software, 32(4), pp. 68-75.
22. Wingkvist, A., Ericsson, M., Lucke, R., Lowe, W., 2010. A metrics-based approach to technical documentation quality. 7th Int. Conference on the Quality of Information and Communications Technology (QUATIC). IEEE comp. soc. press.