Turkish Journal of Computer and Mathematics Education *Vol.12 No.3(2021), 3338-3344*

Research Article

Tailoring effective requirement's specification for ingenuity in Software Development Life Cycle.

Naveen N Kulkarni [a], Prof. (Dr.) K. P. Yadav[b]

[a]Research Scholar, Department of Computer Science, Sangam University, BhilwaraBhilwara, Rajasthan, India
[b]Research Supervisor Department of Computer Science, Sangam University, BhilwaraBhilwara, Rajasthan, India

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: Software Requirements Engineering (SRE) process define software manuscripts with sustaining Software Requirement Specification (SRS) and its activities. SRE comprises many tasks requirement analysis, elicitation, documentation, conciliation and validation. Natural language is most popular and commonly used to form the SRS document. However, natural language has its own limitations wrt quality approach for SRS. The constraints include incomplete, incorrect, ambiguous, and inconsistency. In software engineering, most applications are object-oriented. So requirements are unlike problem domain need to be developed. So software documentation is completed in such a way that, all authorized users like clients, analysts, managers, and developers can understand it. These are the basis for success of any planned project. Most of the work is still dependent on intensive human (domain expert) work. consequences of the project success still depend on timeliness with tending errors. The fundamental quality intended for each activity is specified during the software development process. This paper concludes critically with best practices in writing SRS. This approach helps to mitigate SRS limitation up to some extent. An initial review highlights capable results for the proposed practices.

Keywords: Requirement Engineering, Software Requirement Specification, SRS best practice

1. Introduction

A Software document is a representation of information designed for developing a software product by the analyst. This document explains all the requirements related to the business process, user experience, product development at different phases...etc… Such documents are persistent, and suitable in the system development life cycle for archival uses, or it may be ephemeral, lasting only for viewings.

Requirements Specification (SRS) is a detailed description of a proposed software system. This is widely acknowledged and used by most of the software developers, irrespective whether it's a service, product or system, even any distinct requirements also integrated. This document consists of all essential requirements required within the Software Development Process (SDP) or Software Development Life Cycle (SDLC). To develop a proposed system one should have clear understandings of the system. This can be achieved by gathering all necessary requirement with the stakeholders recurrently. The SRS document is flourished between stakeholder (users/clients) and company contractors based on the mutual contract. This document is prepared by System Analysts and delineates how software is developed,. What are the main business objectives and functionality of the proposed product, and how core functions are performed.

1.1 Examples of documentation in SDP

1. Business Needs Specification (BNS)
2. Business Requirements Document (BRD)
3. Functional Requirements Document (FRD)
4. Market Requirements Document (MRD)
5. Product Requirements Document (PRD)
6. User Interface Requirements Document (UIRD)
7. Technical Requirements Document (TRD)
8. Quality Requirements Document (QRD)
9. Software Requirements Specification (SRS)

10. Client Requirements Document (CRD)

Structured software documentation helps to create metrics in which capabilities are identified with their results for improvement. Uniform software development processes are necessary for training, management, review, and tools supporting.

2 Literature Review

A software development process usually consists of multiple phases as a given System development life cycle. In which, from requirement analysis to implementation most of the applications face similar problem domains with common similarities. In software engineering, most applications are object-oriented. So requirements are unlike problem domain need to be developed. To check similarities in requirement specification both SRS and MIS are chosen w.r.t. scaling small and medium companies.

Eko Handoyo, R Rizal Isnantoa, Mikhail Anachiva Sonda [1] explains "how to generate common requirements from the active business process. So it can minimize errors and helps users to understand the overall system". Valentina Lenarduzzi [2] research finds there is a lot of knowledge while developing Artificial Intelligence (AI) applications. Which are often poorly tested with low quality. This is due to inappropriate documentation and adaptation with indecent practice in development process. They also highlight "the most common quality issues that developers face during the development and why the training of developers in AI lacks".

Vineet Dubey, Vandana Pandey, and Dhirendra Pandey [3] explains how "effectiveness of expressing requirements specifications can be greatly improved with simple and readily existing methods. Project managers ensure standard requirements and design the outset the project. They also suggested the technical environment and intended apply of the SRS are emphasized to support a straightforward simple sentence structures to specify all requirement. It is necessary for SRS to communicate effectively with the user. H. A. Soares and R. S. Moura [4] describes requirements engineering is the process to define, and maintain needs that intend to support SRS in formation and maintenances. Documents must be user-friendly, easy to understand by users. This is the main fundamental quality to make success for most of the projects. Work eyes on Natural Language Processing techniques with ERS-EDITOR tool to automate process steps for promising results.

Paiva A.C.R., Maciel D., da Silva A.R. [5] research work describe "model base test approach accepts test-cases extracted from the requirements (RSL) specifications. Automatically it is adapted to test automation framework. Which is to be executed against web application under some test". Their approach encourage practising specifying requirements as well as tests during planning stages and keep specifications aligned with one another. Sandeep Kulkarni and Vipan Kumari [6] assume that "human intelligence can be represented in symbolic structure with functions. Debate is still open to discuss, but AI developers and researchers merely wait to create imaginary computer that might model all of human intelligence".

A. Rashwan, O. Ormandjieva and R. Witte [7], study limns "Non-Functional Requirements have major impact on overall time and cost of the process in system development. To improve system development an automated test for SRS is conducted with various Non-Functional Requirements. Some cost cutting concern also highlighted in their research work". Results obtained from ontology and Vector Machine (SVM) classifier demonstrate the uses. M. Riaz, J. King, J. Slankas, L. Williams, and N. Carolina [8], [9], [10] and [11] study reveals "There are many techniques where non-functional requirements (NFR) are captured from all requirement arti_facts. These techniques used by many researchers and authors. To capture NFR multiple dataset and machine learning techniques are used, such as PROMISE-Datasets, JohnSlanks-Dataset and AlessoFerrari-Dataset which are available publicly. Brooks, Frederick P. Jr. [12],[13], "Developments in technology or management is multidimensional. Every decade, there are improvements in the order of magnitude that promised increase in productivity with simplicity and reliability". Department of defense, USA [14] documents describes "conventionality in standards that were followed during SRS writing process. Designing SRS with good fonts, style, color with highlighting make document significantly special". Alberto Rodrigues da Silva. [15], work result proposes "a generic approach to validate automatically for requirement specification by describing toolset (i.e., SpecQuA tool) with utility and practicability. Their approach help to mitigate some limitation particularly w.r.t ambiguous, inconsistency, and incompleteness".

Software engineering comprises many tasks requirement analysis, elicitation, documentation, conciliation, and validation. Natural language is most popular and commonly used to form the SRS document. However, natural language has its own limitations w.r.t quality approach for requirement specification. The constraints include incomplete, incorrect, ambiguous, and inconsistency.

3 Methods and Tools

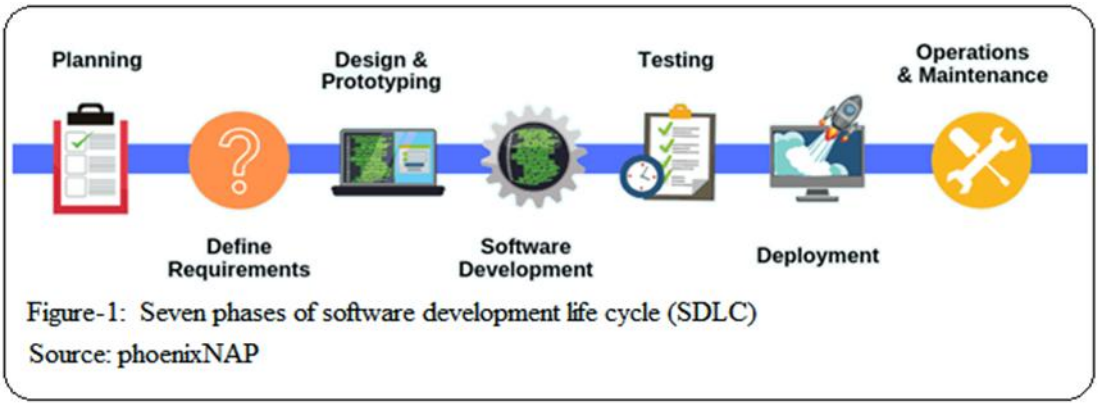

Figure-1: Seven phases of software development life cycle (SDLC)

Source: phoenixNAP

Requirements Engineering is the practice to define software manuscripts with sustaining requirement specification and its activities. SRS documentation is prepared in such a way that, all users like clients, analysts, managers, and developers can understand it. It is the base to success for most projects. Software Development Life Cycle includes 7 phases they are shown in above "Fig-1"

The fundamental quality is intended to each activity specified during the software development process. IEEE sets some standards and provides standard documentation methodology for software development process [16]. As an example, we consider AltexSoft Inc company for how they do documentation process. AltexSoft Inc, is a Technology and Solution Consulting company that offers variety of IT services. Stages in project documentation with their purpose are shown below. Below "Fig-3" is the documentation process used in AltexSoft software company

4 Analysis and results

Forbes Technology Council released insights reports [17] for tech & business on March 2020. Insights report revealed why software projects are derailed. Common factors that cause project failure includes not understanding business needs, requirement priority, lack of clarity, unclear requirements, failing in coordination and planning, undefined roles, over customization…etc…

According to insight published by Standish Group's Annual CHAOS report [18], 66% IT projects ends with failure. Out of three only one project is succeeded. Other two are prone to risk of failure. This is given in below table.

Table – 1 Software Project Failure Rate

Project type	Succeeded	Challenged	Failed
Large	8%	26%	41%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%

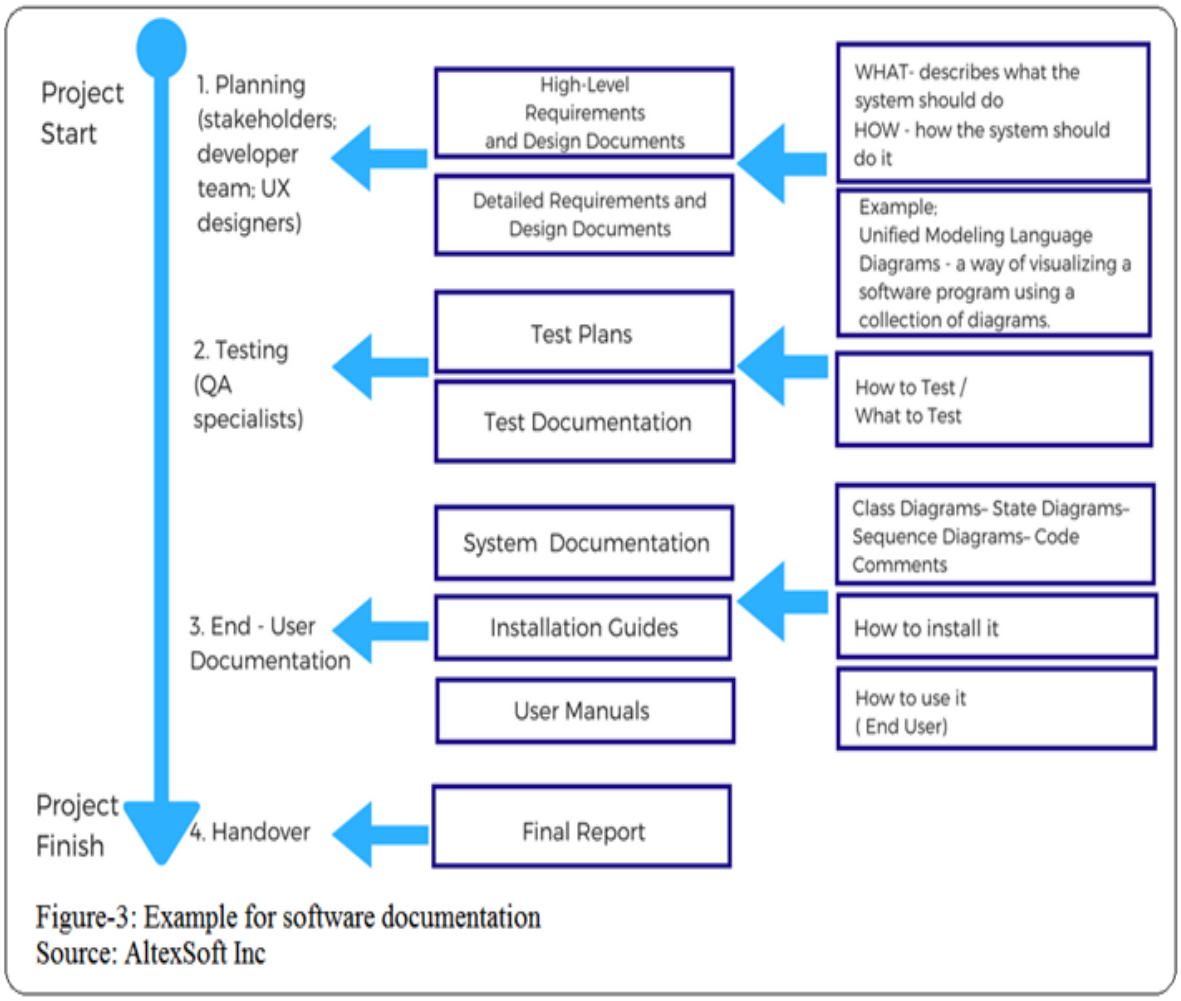

Figure-3: Example for software documentation
Source: AltexSoft Inc

When we analyze this table large projects prone to more challenge or failure. Whereas one in every tenth projects fails. One of the main reason for such project failure is lack of quality requirement specification. UK govt projects like e-borders system (200 million pound), BBC video archive project (100 million pounds), Back office project for Ministry of Justice (56 million pounds) are some best examples for project failure. millions of employs from UK govt unable to launched the said bespoke software. Now we must think about the chances? we have with smaller business.

4.1 Suggested criteria for developing high- quality SRS

1. Ensure reliability i.e., requirement must satisfy output with either pass/ fail.
2. Avoid Ambiguity. Where each and every specification is clearly defined with single meaning even at multiple level. different meaning is strictly avoided.
3. All types of requirements should be complete. It should not possess any type of errors. But errors are unavoidable. In case of an error a respective message in displayed.
4. There should be no conflicts between requirements event at distinct places. Maintain consistent makes testing and debugging easier.
5. Requirements ID helps uniquely identify particular module. It is the base for test case.
6. Ranking criteria helps to address problem with its importance and stability
7. Verification of requirements with required results for intended system
8. SRS must be dynamic with index and cross-reference and should not impact on dependents.
9. Document with unique identification is traceable with bidirectional reference.
10. Pre-conditions must be verified clearly.

11. Design independence is nothing but providing alternate choice without implementation details.
12. Abstraction is rightly explained with less details
13. Shunning notations and symbols makes user friendly system even for non computer experts.
14. Freeze document means, it remain intact until further action/review.
15. Scrapping irrelevant requirements not only reduces workload, it stops confusion.
16. Ensure software security. Any kind of attack and misuse must be prevented
17. Assessing performance w.r.t key elements like quality, speed, cost, flexibility...etc
18. Feasibility test checks whether new system is practically adoptable and fulfil specific condition within the said time frame.
19. As per set condition, document is tested with User Acceptance Testing (UAT). New system is implemented only UAT and client approval.
20. In validation. the whole product with code is examined by testers to ensure legality of new products.

4.2 Proposed solution for ranking requirements

There are number of choices (n) out of which only one (r) should be choose to assign ranking for requirement. This can be represented mathematically using combination formula when order doesn't matter.

Combination Formula :

C (n, r) = nCr = nCr =C (n¦r) = n!/r!(n-r)!

Where n □ is a number of choices which has 9 possibilities and only one rank assigned at a time i.e., r=1. Each requirement is assigned to one rank at a time. Analyst must ensure that each requirement should be complete, non- ambiguous and maintain consistency while assigning rank. The structure format for ranking looks like below.

The structure of matrix will be

[■(Rank_1&Rank_2&Rank_3@Rank_4&Rank_5&Rank_6@Rank_7&Ranl_8&Rank_9)]

According to the output we needed from the specific requirement, the intensity of the ranking is decided and assigned by the analyst or developer. Assigning ranking depends on knowledge of the analyst, developer skill and nature of the project.

Table-2 :Requirement classification with ranking specification

Rank Metric	**More Important**	**Less Important**	**Un important**
More Critical	Rank_1: More Critical and More Important	Rank_2: More Critical and Less Important	Rank_3: More Critical but Unimportant
Less Critical	Rank_4: Less Critical and More Important	Rank_5: Less Critical and Less Important	Rank_6: Less Critical but Unimportant
Non_Critical	Rank_7: Non_Critical but More Important	Rank_8: Non_Critical but Less Important	Rank_9: Non_Critical and Unimportant

This ranking approach reduces complexity in the software development process. There is some consequence when requirements address complex problems, then it becomes a difficult task to a design document that ease rather than inhibit understandings.

5 Conclusion and Future scope

It is hard to assume, but true that the writing SRS process is rigid and impossible for most clients to understand what are the exact business requirements?. It is unfeasible, even for software engineers who are working in the company to specify a complete, precise, and correct requirement specification for the proposed system. Since IT projects are complex and uncertain inherently. This is okay, modern tools help to mitigate risks that are prone to failure. The best practical approaches are discussed above. Keeping time constraint requirements are planned in advance and executed efficiently. Our effective analyses claim that the ranking approach for requirement specification and User Acceptance Testing (UAT) Since, both factors play an important role in project success.

As Data Science is rising rapidly. Artificial Intelligence is still in development state. Both technology are capable to reinvent the whole SRS writing process. Intelligent system may reduce skilled human intensive work. In this transformation stage, our upcoming study will be based on these technologies. How best practices are achievable through AI and data science.

References

Eko Handoyo, R Rizal Isnantoa, Mikhail Anachiva Sonda, (2013), Title: "SRS Document Proposal Analysis on the Design of Management Information Systems According to IEEE STD 830-1998", Procedia - Social and Behavioral Sciences, Vol-67, 2012, Pages 123-134, ISSN 1877-0428, doi: 10.1016/j.sbspro.2012.11.313.

Lenarduzzi V., Lomio F., Moreschini S., Taibi D., Tamburri D.A. (2021) Software Quality for AI: Where We Are Now?. In: Winkler D., Biffl S., Mendez D., Wimmer M., Bergsmann J. (eds) Software Quality: Future Perspectives on Software Engineering Quality. SWQD 2021. Lecture Notes in Business Information Processing, vol 404. Springer, Cham. doi: 10.1007/978-3-030-65854-0_4

Dubey, Vineet & Pandey, Vandana & Pandey, Dhirendra & Csvtu, & Bhilai, (2020), Title: "Making of Quality SRS for Requirement Engineering Process", DRR Journal UGC Care Group I Journal, ISSN : 2347-7180 Vol-10 Issue-07 No. 38 July 2020

H. A. Soares and R. S. Moura, (2015), "A methodology to guide writing Software Requirements Specification document," 2015 Latin American Computing Conference (CLEI), Arequipa, 2015, pp. 1-11, doi: 10.1109/CLEI.2015.7360001.

Paiva A.C.R., Maciel D., da Silva A.R. (2020) From Requirements to Automated Acceptance Tests with the RSL Language. In: Damiani E., Spanoudakis G., Maciaszek L. (eds) Evaluation of Novel Approaches to Software Engineering. ENASE 2019. Communications in Computer and Information Science, vol 1172. Springer, Cham. https://doi.org/10.1007/978-3-030-40223-5_3

Sandeep Kulkarni & Vipan Kumari, (2018) Title: "Use of Artificial Intelligence in Software Development Life Cycle Requirements and its Model", eISSN: 2395-0056, vol-05 Issue: 08/2018

A. Rashwan, O. Ormandjieva and R. Witte, "Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier," 2013 IEEE 37th Annual Computer Software and Applications Conference, Kyoto, 2013, pp. 381-386, doi: 10.1109/COMPSAC.2013.64.

M. Riaz, J. King, J. Slankas, L. Williams, and N. Carolina, (2014), "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts". in Proc. IEEE 22nd Int. Requirements Eng. Conf. (RE), Aug. 2014, pp. 183_192.

D. Ameller, C. Ayala, J. Cabot, and X. Franch, ``Non-functional requirements in architectural decision making," IEEE Softw., vol. 30, no. 2, pp. 61_67, Mar. 2013.

Apache PDFBox | A Java PDF Library. Accessed: [Jan. 14, 2021]. [Online].

URL: https://pdfbox.apache.org/

A. Ferrari, G. O. Spagnolo, and S. Gnesi (2017), Title: "Towards a dataset for natural language requirements processing," in Proc. REFSQ Workshops, 2017, pp. 52_62.

Brooks, Frederick P. Jr., (2017), Title: "No Silver Bullet: Essence and accidents of software engineering", IEEE Computer, vol-15, no-1, April 2017, pp. 10-18.

Brooks, Frederick., Title: "No Silver Bullet Essence and Accidents of Software Engineering," in Computer, ISSN: 0018-9162, vol. 20, no. 4, pp. 10-19, April 1987, doi: 10.1109/MC.1987.1663532.

Department of defence USA, Title: "Military Standard Specification Practice", OD MIL-STD-490A, June 4, 2015. URL: https://www.product-lifecycle-management.com/download/MIL-STD-490A.pdf (DOA: Jan 14-2021)

Alberto Rodrigues da Silva, (2014-15), Title: "Quality of Requirements Specifications A Framework for Automatic Validation of Requirements", In Proceedings of the 16th International Conference on Enterprise Information Systems, pages 96-107, DOI: 10.5220/0004951900960107, Copyright: c SCITEPRESS

ISO/IEEE (2018), Title: "International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering," in ISO/IEC/IEEE 29148:2018(E) , vol., no., pp.1-104, 30 Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686.

Forbes Technology Council (2020), Article Title: "Common Reasons Software Projects Fail", Retrieved from URL: https://www.forbes.com/sites/forbestechcouncil/2020/03/31, (DOA: 15/01/2021)

Insight report by CHAOS, Title: " The Standish Group Report", 2014 © The Standish Group 1995, Retrieved from URL: (DOA: 01/01/2021)