International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-2S4, July 2019

A Tool for Analyzing Software Requirements Document Quality

C.Arun, S.Karthick, G Sivan Muthu Krishnan and Soumitro Dutta

Abstract*: The quality of a Software Requirements Specification (SRS) is measured in terms of quality properties such as completeness, conciseness, consistency and understandability. In general, evaluation of the SRS quality is done manually during review sessions. The evaluation process, however, is hugely dependent on the expertise of human experts i.e. the reviewers. In fact, the judgment of the human experts could also be inconsistent due to various factors including experience, knowledge and domain. The objectives of this study are to (1) identify feasible rules to measure SRS quality; and (2) help requirements engineer to improve their SRS quality. In this study, we analyzed SRS quality properties from the literature and identified quality factors that are feasible to be automated. From here, we identified two types of properties that are (1) requirements sentence quality (RSQ) and (2) requirements document quality (RDQ). For each of the type, its relevant quality indicators were identified. From here, rules on how to identify the quality indicators were further investigated and documented. As a case study, we implemented SRS Quality-Checker tool concept for demonstrating how the rules were implemented to measure the SRS quality

Index Terms*: Measuring SRS Quality, Requirements Document Quality, Requirements Review, Requirements Sentence Quality, Software Requirements Specification.

I. INTRODUCTION

Requirements are features of the system-to-be-built that are discovered and identified before building any products or system. It's the milestones or outcome capability which could be satisfy by the components to adhere the contract, values, specification etc.,[12]. Requirement Engineering (RE) is a systematic process which entails the gathering of requirements from various sources and the software development life cycle is executed accordingly [1]. RE is generally performed during the early stages of software development lifecycle. RE aims to create a set of requirements for a system through activities that will help to find, examine, document, validate and maintain them [2].

The validation activity involves evaluation and verification of requirements such as review, testing, and inspection. The final outcome of the RE process is the Software Requirement Specification (SRS), which contains the needs of the stakeholders and its constraints.

A clear set of requirement statements is one of the critical success factors to ensure project success [10]. Incomplete Requirements, on surveying, were found to be the primary reason why project development is hindered and subsequently dropped [10]. As can be observed, requirements give a significant impact to success of software projects. Hence, it is important to ensure that quality of SRS is accomplished to support the achievement of any software project.

A SRS is a report that portrays all the remotely discernible practices and qualities expected of a product framework [9]. It is important to the developers as it enables them to spare time on correspondence, limit advancement endeavours, gives the client criticism, disposes of errand duplication, encourages the exchange to new clients or to new machines, and breaks issues into parts. Moreover, it likewise fills in as the fundamental record to check approval and testing forms. There is no standard way to write an SRS document. Nonetheless, a good SRS should contain all the information as suggested in the IEEE Recommended Practice for SRS [15]. A poor requirement cannot lead to excellent software because the quality of any product depends on the quality of SRS itself [10]. Moreover, not all software developers are being trained to properly document and verify the quality of requirements in the specification document.

Quality Properties of specification is one that contributes to successful, lucrative design of software that solves need of the end user [9] and some of the qualities are:

i. Conciseness

A Requirement Specification is a concise, if the intended needs aligned to only a single purpose which cannot be interpreted by different ways by the end users [15]. This criterion possibly is the most complicated feature to realize using natural language. However, the outsider checker can decide the presence of compactness by noting the predefined shut inquiries.

ii. Understandability

When the stakeholders use the SRS to develop the software, each requirement must be completely comprehended by them.

iii. Completeness

A Software Requirement Specification is complete if it includes the following elements [15]:

a) All noteworthy specifications, whether involving functional, non-functional quality, design constraints, attributes, or external interfaces.

b) Complete names and references to all figures, tables, and graphs in the SRS and meaning of all terms and units of measure.

During requirement validation activity, requirement review is

Revised Manuscript Received on July 05, 2019.

C.Arun, Dept of Software Engg, SRMIST, Chennai
S.Karthick, Dept of Software Engg, SRMIST, Chennai
G. Sivan Muthu Krishnan, Dept of Software Engg, SRMIST, Chennai
Soumittro Dutta, Dept of Software Engg, SRMIST, Chennai
Email: soumitrodutta96@gmail.com, srivats0811@gmail.com, arunc@srmist.edu.in

conducted to measure the SRS quality. Requirement review involves several independent inspectors that individually analyze the SRS to search for defects, and then meet to discuss the findings and recommend appropriate actions for fixing the agreed defects [12]. The purpose of conducting the requirement review is to find errors and point out other matters of concern in the requirement specification.

One of the challenges of requirement review is when there is a lack of ability to recognize the defect in the requirement. It will be tedious for the reviewers to search the errors in the requirements due to factors including time constraint. Despite that, the real problem arises is inconsistent judgment among the human experts [4]. The reason for that is due to the different background of expertise and knowledge.

Hence, we suggested a solution to reduce the challenges and issues in such problems by analyzing and proposing the Quality Properties that are feasible to be implemented in our tool. The purpose of this paper is to identify the Quality Properties rules to measure the SRS quality. Hence, this can help the requirement engineers to improve their SRS quality. This paper is divided into five sections. The Introduction section lays out the background details, Section II provides the literature review that briefly summarizes the aspects of SRS quality, while Section III presents the Research Methodology that we applied in this study. Next, Section IV provides the discussion regarding our study and finally, Section V concludes this paper

II. EXISTING METHODOLOGIES

In this paper, we analyzed two research papers that discussed the Quality Attributes to determine the quality of SRS. The papers are:

i. Quality Evaluation of Software Requirements Specification [5];

ii. Writing Effective Requirements Specification [6].

After we analyzed both the papers, we scoped this research paper into two aspects of the SRS quality [4]:

i. Requirement Sentence Quality (RSQ): the linguistic nature of single sentences thought about independently;

ii. Requirement Document Quality (RDQ)): the nature of the sentences considered with regards to the entire necessities reports.

From the papers, we finalized the quality attributes that are feasible to be automated by using the rules to determine the RSQ and RDQ that will further be discussed in the subsequent subsection.

2.1. Quality Attributes

The definition of each of the Quality Properties has been discussed in the Introduction, while this section will briefly explain the definition of each of the Quality Attributes. The Quality Attributes are divided into attributes related to RSQ and RDQ. The following are Quality Attributes with its brief definition of each attribute.

2.1.1 RSQ related attributes:

a) Implicit Sentences

A sentence is a certain subject sentence if:

- Its subject contains an expressive descriptive word;
- Is communicated by methods for pronouns;
- Is determined by relational words and is indicates by a descriptor [5].

b) Optional Sentences:

A sentence is discretionary in the event that it contains an alternative [5].

c) Fuzzy Sentences

A sentence is considered to be vague or fuzzy, if it lacks distinctive meaning [5].

d) Weak Sentences

Category of clauses that will cause uncertainty and might give different meanings according to different perspectives [6].

e) Multiple Sentences

It an expression which may adjusted towards numerous significance or diverse translation for various clients which ordinarily has in excess of one subject or in excess of one fundamental verb as a piece of the expression [5].

f) Directive Words

Class of words and phrases that cite information that elucidate the meaning within the specification document. The data and information pointed to by directive words will enhance the SRS [6].

2.1.2. RDQ related attributes

a) Readability Index

The Readability Index is a measure that indicates the level of ease of understanding when going through the requirements document [6].Each of the above attributes had been associated with the Quality Properties under the Goal Properties as listed in Table 1.

Table 1. List of Quality Attributes

Aspects	Quality Properties	Quality Attributes
RSQ	Conciseness	Implicit Sentences
		Optional Sentences
		Vague Sentences
		Weak Sentences
	Understandability	Multiple Sentences
	Completeness	Directives Sentences
RDQ	Understandability	Readability Index

2.2. Existing Measurement Techniques

The current estimation tools in the business are Automated Requirements Measurement (ARM) device [6] and Quality Analyzer of Requirements Specifications (QuARS) Fig .1, device [5]. The ARM examines the assigned record by the client as the content document that contains the

prerequisites particular. The other looks through every content line of prerequisites detail for particular words and expressions [6]. The accompanying words and expressions demonstrate the report's quality as a particular of prerequisites. In the end of the tool's process, it can generate a report contained imperative report and detailed weak phrase report.

Figure 1: Research Methodology

Moreover, QuARS is a model tool that consequently plays out the quality assessment of SRS by recognizing the markers and calling attention to the sentences containing potential errors and ambiguities [5]. The document is used during review sessions where the output supports the reviewers in terms of detection of inaccuracies, ambiguities and linguistic inconsistencies in the document

III. RESEARCH METHODOLOGY

3.1 Literature Review

The existing works on the related areas to measuring SRS quality had been investigated. The outcome of this is presented in Section II.

3.2 Analyze Rule

Table 2. Rules of Quality Attributes

Aspects	Quality Attributes	Rules
RSQ	Rule 1: Implicit Sentences Rule 2: Optional Sentences Rule 3: Vague Sentences Rule 4: Weak Sentences Rule 5: Multiple Sentences Rule 6: Directives Words	Refer to the corresponding terms in Table 3
RDQ	Rule 7: Readability Index	Flesch Reading Ease Readability. Calculate the total words, total sentences, and total syllables. Then include all the calculation using the following formula:- $206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$

In order to determine the feasible rules, the Quality Attributes mentioned in both research papers were analyzed, compared and the similar rules were mapped to the same Quality Attributes as listed in Table 2. The table lists the rules to identify each of the Quality Attributes according to RSQ and RDQ [5] [6].

For each of the rules, the respective terms which are applicable to the respective rules are listed in Table 3.

Table 3: Terms to Identify Error

Type	Words
Implicit Words	this, these, that, those, it, they, above, below, previous, next, following, last, and first.
Optional Words	can, eventually, if appropriate, if needed, may, optionally, possibly.
Vague Words	adequate, back, bad, clear, close, easy, efficient, far, fast, future, good, in front, low, near, new, old, past, recent, significant, slow, strong, today's, useful, weak, and well.
Weak Sentences	as a minimum, as applicable, as appropriate, be able to, be capable, not limited to, the capability of, the capability to, easy, effective, if practical, normal, provide for, to be determined, and timely.
Directive Words	figures, for example, table, note

Then, these rules (i.e. Rule 1-7) and terms (see Table 3) are designed to be implemented in the system.
Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar.

3.3 Implementation

We developed a tool called SRS Quality Checker as a proof-of-concept of the rules, which measures the quality of SRS document based on RSQ and RDQ. Our system is developed by following the spiral model, Fig 2. It is worth to point out that the spiral model used here is the software development methodology that we applied to guide us for the tool development only.

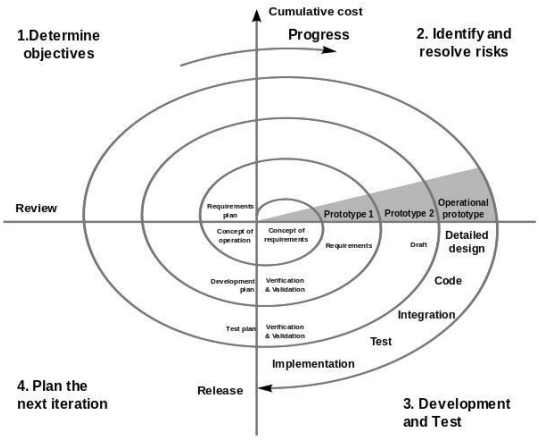

Figure 2: Spiral model

The spiral model consists of phases wherein each phase initiates the implementation of a design goal and terminates with a client's (who may be internal) assessment [7]. The process happens in clockwise directions beginning at the origin of the spiral, in iterations, resulting in a deliverable at the end of rotation.
In the first iteration, requirement specification is gathered for our system. This

phase is needed as requirement specification is crucial for any development of system or product because it describes how a product will work.

Figure 3. Algorithm for measuring multiple sentences

After specifying the requirements, the prototype is developed in the second iteration. Our prototype is developed according to its requirements. The algorithms for both the

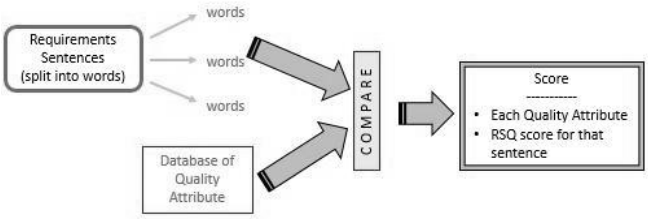

Figure 4. Algorithm for measuring the quality attribute except for multiple sentences

RDQ and RSQ properties are shown in Figure 3-6. Fig. 3 and Fig. 4 demonstrate how to measure RSQ, while Fig. 5 explains how to measure RDQ. In order to measure RSQ, each requirement sentences in SRS is evaluated for each quality attributes that had been mentioned above (see Table 2).

Fig. 3 explains the algorithm that is for multiple sentences. See Rules 5. This rule implements the Natural Language Processing (NLP), which is Part-Of-Speech Tagger or called POS Tagger [7]. POS Tagger is a software that can tag each word in a text given in a particular language with a part of speech like noun, adverb, verb, adjective, etc. depending upon the context of its usage. [8]

Pseudo code for identifying multiple sentences:
Data: *Each requirement sentence of the document, R*

Result: *Number of error found in the document with respect to quality of requirements.*

```
while not at end of the input document do
   read POS tagger;
   if word tagged equals VERB then
      countVerb = countVerb - 1;
   end;
   if countVerb less than -1 then
      totalVerb = totalVerb + countVerb;
   end;
end
totalError = totalError + totalVerb;
return totalError;
```

For this algorithm, firstly, each requirement sentence is read by the POS Tagger method then assigns parts of speech to each word. Then, the algorithm read one by one character of the requirement sentences to find the multiple verbs. If the requirement sentences have more than one verbs, it stated as having an error of multiple sentences. Due to space constraint, we only included the algorithm for this rule's implementation. Whilst, the algorithm in Fig.5 applies for all quality attributes except for multiple sentences. It compares every word in each requirement sentences with the database that contain the rules of quality attributes. This algorithm works, first of all, by splitting each requirement sentences into word.

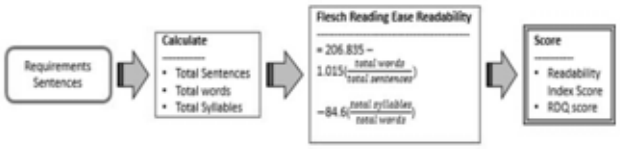

Figure 5. Measuring readability index

Then, it compares the split word with each rule in the database of quality attributes. If they match, the error word is displayed. For every error found, it is scored with a negative one and this is applied in both process in Fig. 4 and Fig. 5. Each requirement sentence is free from error based on this work or rules.

To measure RDQ, Flesch Reading Ease is used as a rule to identify whether SRS document is readable by the user or not [6]. For this algorithm, total sentences, total words and total syllables of SRS document are calculated first. Then, they are included in the formula as in Figure 6. The score is calculated from the formula and used as an indicator to assess the ease of readability of a document.

For design, based on the algorithms, we used UML class diagram to visualize all the elements in the implementation. See Appendix for the class diagram of the SRS Quality Checker tool.

In the third iteration, the prototype is tested by using the real SRS document. It is worth to note here that a filtering process is required to ensure the input (i.e. SRS document) is in the correct format.

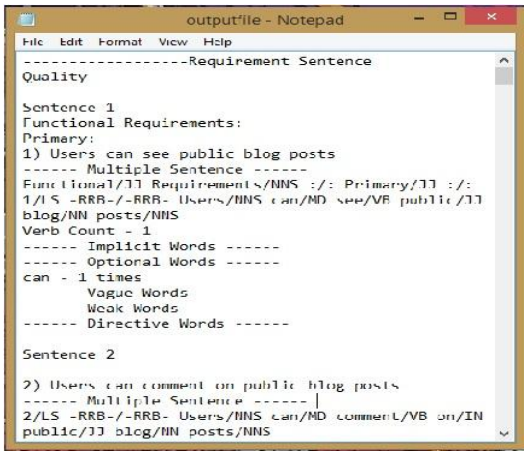

Figure 6. Snapshot of the RSQ report.

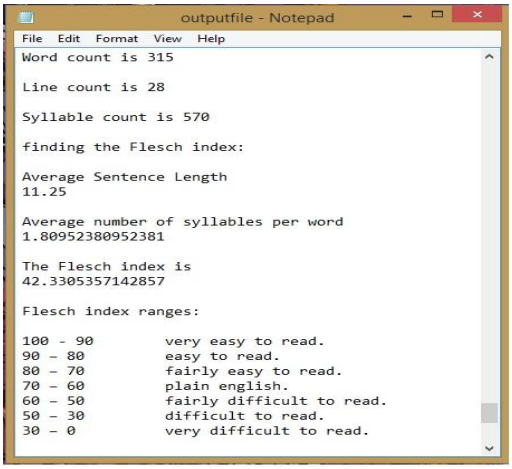

Figure 7. Snapshot of the RDQ report

Whilst, in the fourth iteration, all of these changes from one iteration to another iteration are repeated until it leads to a system, which is functional.

Finally, the tool shall generate a summary report consisting of all the errors detected from the SRS document. See Figure 7 and Figure 8. This allow the users to monitor the quality of their SRS document as well as updating the contents of the document to improve the SRS quality

3.4. Evaluation and Result

For every error found during measuring the RSQ properties, it is given the score of negative one and this is applied in both algorithms in Fig. 3 and Fig. 5. The errors are displayed in the tool according to their quality attributes. Each requirement sentence is free from error if it has a score of zero. In addition, the scores obtained after measuring RDQ can be interpreted as shown in Table 4.

Table 4: Score of Flesch Ease Readability Index [13]

Score	Level Details
100.0–90.0	Very easy to read.
90.0–80.0	Easy to read.
80.0–70.0	Fairly easy to read.
70.0–60.0	Plain English.
60.0–50.0	Fairly difficult to read.
50.0–30.0	Difficult to read.
30.0–0.0	Very difficult to read.

The Flesch Ease Readability Index [14] depends on the normal number of syllables per word and the normal number of words per sentence. Scores go from 0 to 100 with standard written work averaging 60 to 70. The higher the score, the more prominent the quantity of individuals who can promptly comprehend the archive [6].

IV. DISCUSSION

We are aware that this research is incomprehensive as there are Quality Attributes that are not feasible to be automated. In such cases, human reviewers are still required to make judgments. In addition, the feasibility of the approach is subjected to further evaluation by experts. Nonetheless, our effort in conducting this research is at least can be useful for requirements engineers to measure their SRS quality. The implementation can benefit in measuring SRS quality for the applied attributes or even to be used in pre-review sessions.

V. CONCLUSION

This paper has analyzed and proposed a methodology to measure the SRS Quality according to Quality Attributes. We applied the methodology and rules to the SRS document and evaluated the results which showed than on using this as a framework will result in improved measuring of the SRS quality. As a conclusion, by measuring the quality of SRS using rules, it could assist requirement engineers to improve their SRS quality.

REFERENCES

1. D. Pandey, U. Suman, and A. K. Ramani, “An Effective Requirement Engineering Process Model for Software Development and Requirements Management,” in Proceedings of the InternationalConference on Advances in Recent Technologies in Communication and Computing, 2010, pp. 287–291.
2. J. Siddiqi, “Requirement engineering: The emerging wisdom,” IEEESoftw., vol. 13, no. 2, pp. 15-19, Mar. 1996.
3. S. O. Mokhtar, R. Nordin, Z. A. Aziz and R. M. Rawi, “Issues and challenges of requirement review in the industry.” Indian Journal of Science and Technology, vol. 10, no. 3, pp. 1-5, Jan. 2017.
4. J. Krogstie, O. I. Lindland and G. Sindre, “Towards a deeper understanding of quality in requirements engineering,” in Proceedings of the International Conference on Advanced Information Systems Engineering, 1995, pp. 82–95.
5. F. Fabbrini, M. Fusani, S. Gnesi and G. Lami, “Quality evaluation of software requirement specifications,” in Proceedings of the Software and Internet Quality Week 2000 Conference, 2000, pp. 1–18.
6. Wilson, W. M., “Writing effective requirements specifications,” CrossTalk: The Journal of Defense Software Engineering, pp. 16–19,1999.
7. V. Rastogi, “Software Development Life Cycle Models-Comparison, Consequences,” International Journal of Computer Science and Information Security, vol. 6, no. 1, pp. 168–172, 2015.
8. S. Mall and U. C. Jaiswal, “Evaluation for POS tagger, chunk and resolving issues in word sense disambiguate in machine translation for Hindi to English languages,” in Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 14–18.
9. A. Davis, S. Overmyer, K. Jordan, J. Caruso, F. Dandashi, A. Dinh, G. Kincaid, G. Ledeboer, P. Reynolds, P. Sitaram, A. Ta, and M. Theofanos, “Identifying and measuring quality in a software requirements specification,” in Proceedings of the First International Software Metrics Symposium, 1993, pp. 141–152.
10. D. Rubinstein, “Standish group report: There’s less development chaos today,” Software Development Times, vol. 1, 2007.
11. P. Jalote, An Integrated Approach to Software Engineering, 3rd Edition. Narosa Publishing House, India, 2005.
12. A. van Lamsweerde, Requirements Engineering: From System Goals to UML Models to Software Specification. 1stedition. NJ: Wiley, 2009.
13. R. Flesch, “How to Write Plain English”. University of Canterbury. Available at http://www.mang.canterbury.ac.nz/writing_guide/writing/flesch.shtml. [Retrieved 5 February 2016].
14. J. P. Kincaid, J. Fishburne, R. L. Rogers and B. S. Chissom, Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel.University of Central Florida, Feb. 1975
15. “IEEE Recommended Practice for Software Requirements Specifications,” IEEE Std., 830-1998, pp. 1–40, Oct. 1998.

AUTHORS PROFILE

C.Arun received his B.Tech in Information Technology and M.E in Software Engineering and currently pursuing his Ph.D. He is currently working as an Assistant Professor in SRM Institute of Science and Technology, India having more than 10 years of experience and 1 year industry experience. His research interest includes software testing, software architecture, quality analysis, and machine learning.
Email: arunc@srmist.edu.in
Department of Software Engineering SRMIST, Chennai, Tamil Nadu.

S.Karthick received his B.E and M.E in Computer Science and Engineering and currently pursuing his Ph.D. He is working as an Assistant Professor in SRM Institute of Science and Technology, India having more than 13 years of teaching experience and 2 years of industrial experience. His research interest includes Cyclone data Analysis, Data Mining, Genetic Algorithms, Artificial Intelligence and Machine learning. Email: karthiks2@srmist.edu.in
Department of Software Engineering SRMIST, Chennai, Tamil Nadu.

G. Sivan Muthu Krishnan, he received B.Tech in Software Engineering from SRM Institute of Science and Technology and currently working as Graduate Engineering Trainee in L&T Infotech. His research interest include software Engineeing and Machine Learning.
Email: srivats0811@gmail.com,

Soumittro Dutta, he received B.Tech in Software Engineering from SRM Institute of Science and Technology and currently working as Graduate Engineering Trainee in L&T Infotech. His research interest include software Engineeing and Machine Learning.
Email: soumitrodutta96@gmail.com