International Journal on Advanced Science Engineering Information Technology

Vol.10 (2020) No. 4
ISSN: 2088-5334

Evaluation of Software Requirement Specification Based on IEEE 830 Quality Properties

E. Stephen[a,1], E. Mit[a,2]

[a]Faculty of Computer Science and Information Technology,Universiti Malaysia Sarawak (UNIMAS),94300, Sarawak, Malaysia.
E-mail: [1]ellystephen@yahoo.com; [2]edwin@unimas.my

***Abstract*— Software requirement specification (SRS) documented an essential requirement of software and its external interface. Many studies found the quality of SRS, but lack of the informality organizing of document and representation of functional requirement. This paper aims to evaluate the quality properties of the software requirement specification (SRS). There are four quality properties to be assessed, which are completeness, correctness, preciseness, and consistency. Completeness quality is used to evaluate the structure of the SRS document; meanwhile, the other three qualities used to evaluate the functional requirement. The measurement for each quality properties has been proposed in the previous study. The evaluation process involves a few stages. In short, the prototype would extract text through the provided document, do a calculation, and came out with the result in the form of a similarity percentage. The prototype designs in such ways it minimizes the user interference. Those resulted in reducing human error. Corpus contains libraries of term and topic are expected to increase the reliability of detection. The corpus includes topics extracted from IEEE 830 standard, vague word, terms represent Create, Read, Update, and Delete (CRUD) operation, and terms denote possible datatype. The extracted functional requirement would be refined based on the Requirement Boilerplate (RB) template. RB adopted in the study to ensure the consistency of functional refinement requirements. The percentage of similarity is determined based on comparison with IEEE 830 standard. The rate of the result of each quality properties reflects the quality of the software requirement specification.**

***Keywords*— quality properties; quantitative approach; requirement engineering; software requirement specification.**

I. INTRODUCTION

Software Requirement Specification (SRS) document has become crucial for development. It acts as a guideline and medium of agreement between the client and the developer. It also used to validate the requirement with the stakeholder [1]. Due to heterogeneous software domains, this study focuses on the general properties that must be fulfilled by the domain. The general properties would reflect the quality of the SRS document itself. It is important to consider that software developer meet the quality properties as the document tend to be shared among different level of the organization [2]. Two criteria need to be considered in writing the SRS [3]. First, the SRS must be readable. In brief, the structure of the document must be organized, which lead the reviewer to view the context at ease. Second, the SRS must list a processable requirement with a degree of quality for each functional requirement are stated clearly. The SRS should be correct, complete, consistent, unambiguous, verifiable, modifiable, and traceable [4].

Nowadays, requirement engineering (RE) value increasingly important due to the complexity of capturing the requirement proposed by the client. Requirement Boilerplate has been introduced in the SRS to overcome the complexity. Requirement Boilerplate provides the uniformity for the functional requirement. By the Requirement Boilerplate, the capturing or understanding of the functional requirement could increase. The Requirement Boilerplate is also introduced due to the ambiguity in the natural language [5]. This paper discusses the output of the system, which is built to cater to the heterogeneous SRS domain. Four types of quality properties were measured. These include completeness, consistency, correctness, and preciseness [6]. The assessment target is used to measure the structural component of the SRS and its functional requirements. As stated above, the measurement would focus on four quality properties. These include completeness, consistency, correctness, and preciseness. Completeness quality properties was targeted on the structural of the SRS; meanwhile, the other three proposed quality properties focus on the functional requirement. The framework of the proposed measurement is shown in the study before [6].

A. Structural

The completeness quality properties are used to measure the percentage of similarity between the standard structure stated in IEEE 830 with the provided SRS document [7]. Synonym topic library for the similarity between the standard topic in the IEEE 830 table of content are build. The translation matrix is used to solve the structural issue in SRS document [2].

B. *Functional Requirement*

1) Consistency: The consistency quality properties were measured against any of the non-similarity of each defined functional requirement to assist by the presence of stakeholders. Uniformity of the functional requirement sentence structure can achieve consistency quality [8], [9]. The requirement Boilerplate (RB) template is adopted as an effective way to minimize ambiguity due to the use of natural language [5]. Aside from that, RB also provides uniformity as well as consistency in deriving the functional requirement. Each of the functional requirements must be unique from one another. RB can be divided into two sections, which are stakeholder and capability segment. Syntactic similarity measurement can be used to measure extracted text. Human interference would be needed if the percentage of the measured similarity exceeds a certain expectation [10].

2) Correctness: The correctness quality properties is measured against the presence of the validation process for each of the function. Evaluation of the correctness quality can be performed on the requirement phase [11]. This idea also supported by a few other researchers [12], [13]. A Test-driven approach, which is a use case, has been chosen to validate the functional requirement [14]. Test engineer uses their knowledge and experience to create a test case [15]. The effectiveness of it has been proven in many cases. The effectiveness of the generated test case depends on its correctness and completeness [15]. Test cases usually prepared during the requirement phase. By developing a test case for the requirements, the checking or testing of the requirements is part of the IEEE 830 best practice or management [16]. The requirement analyst should be able to come out with the test case based on the defined functional requirement in the first place. If the requirement analyst is unable to formulate the test case, then perhaps the functional requirement is ambiguous or lack of necessary information needed to develop testing components or test case [17].

3) Preciseness: The preciseness quality properties are described to measure the presence of the vague word and term represent possible datatype for each functional requirement. If the requirements do not reflect the needs or wishes of the client precisely or if the requirements are described in an imprecise way; thus, allow for several interpretations on the system, then, the result is often a system that does not meet the expectations of the client or the users [18]. The presence of the vague word would cause several different interpretations [19]–[22]. A list of the vague word is used to detect the presence of the ambiguous word in the form of assessed functional requirements [19]. The library contains a numerous number of possible vague word and datatype are built in order to increase the reliability of detection. The presence of the datatype increased the readability of the functional requirement, which eventually lead to the traceability of it.

II. Materials and Method

The prototype is expected to come out with the quantitative measurement of the quality properties. As stated in Section II, the evaluation of the SRS would be based on the general quality properties. The subcategorization of general quality properties between the functional requirement and structure is shown in Fig. 1. Further discussion on each quality properties assessment is in the following sub-section.

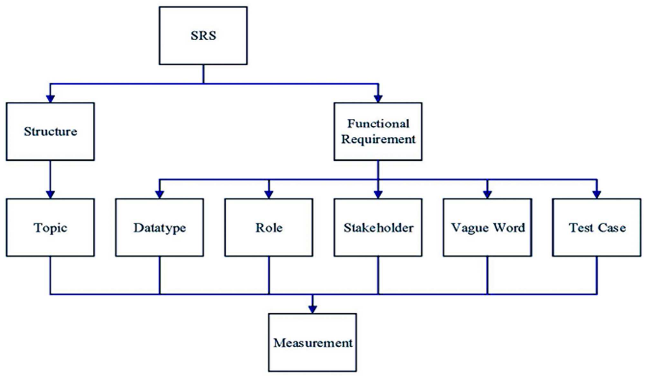


Fig. 1 Quality properties assessment

A. *Prototype Background*

Web-based programming has been used to build the prototype. There would be two-level of the user, which are the developer and admin. The knowledge-based would be updated regularly by the admin. Currently, the knowledge-based is localized. The developer would play an important role in controlling all of the activities aside from maintaining the knowledge-based. The developer would need to come out with a complete SRS document for the assessment. The assessment itself would involve the structure and functional requirements of SRS. The SRS to be assessed must be in the form of .docx format as the prototype design would only accept those formats. The prototype would automatically do the measurement. The updated knowledge-based would be input during the pattern-matching process.

B. *Prototype Assessment*

The assessment would be divided into two, namely the structural and the functional requirement. The assessment would be based on the quality properties to be measured.

1) Structural: First, the developer must create their own individual login account. It is to ensure the security of each evaluated SRS document as each of the developers may involve in a different type of project. Each project would have its own SRS document. By using the prototype, the developer can assess the quality of the document structure. As mentioned before, the assessed SRS document must be in the .docx format. The prototype would extract the text out form the provided document. Then the text would be cleansed. The cleanse text would undergo the pattern-matching process. The cleanse text would be matched against the topic in IEEE 830 standard. Twenty-three topics in standard IEEE 830 table of content would become a constant used to validate each of the provided SRS. The result of the similarity is presented in the form of Table and Graph. The reliability of the result increased by the implementation of the corpus contain a synonym topic. The measurement of overall quality properties for completeness would be done automatically by the prototype itself. To increase the percentage of completeness, the developer may need to replace or update the current SRS document, as suggested by the detection result produce by the prototype. The measurement is based on the similarity topic found in the SRS document. It is recommended that the developer use the topic retrieved from the standard IEEE 830 table of content due to readability.

2) Functional Requirement: The developer needs to have a separate document which only contains functional requirements. Those documents must be in .docx format. The prototype would extract the functional requirement from the provided document. The extracted functional requirement would undergo a cleansing process which involves the singularization process. Then the cleanse functional requirement would be refined into RB template.

Each of the functional requirements would be assessed individually. The assessment would include the possible presence of the test case, vague word, term represent possible datatype, stakeholder, and similarity check of functional requirement. The assessment itself is presented by the properties of the qualities that are assessed. Before the functional requirement can be inputted, the stakeholder element must be defined first. So, the identification of the stakeholder element in each sentence can be made. This can reduce the ambiguity of the functional requirement [9].

Each of the functional requirement may have more than one possible test case. The test case is one way to validate the functional requirement. If the test case development is parallel with the requirement development, the possible number of errors can be detected easily [17]. The test case is more challenge for the user compare to the user story. However, the test case provides more organize information. The example of technicality level of the user story and test case can be seen in Table 1 below:

TABLE I
USER STORY VS. TEST CASE

User Story	As a user, I can click on the viewed picture so that I can view picture info.
Test Case	• The user views the picture. • The user clicks on the picture. • The system searches the database based on the picture id. • The system views the picture info in model-dialog.

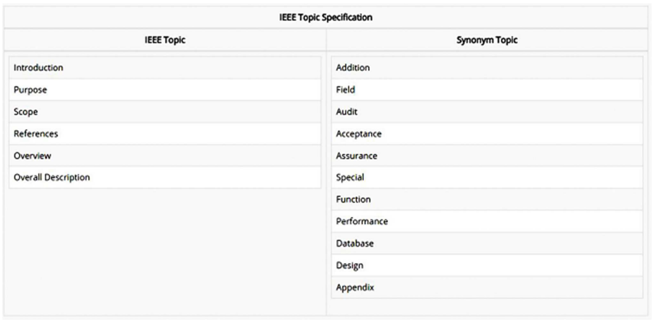
That information may contribute toward the quality of traceability as it also can be used for the next phase after the requirement phase. As each of the projects may have a different domain, the developer needs to have the deep knowledge to understand each functional requirement of the system that need to be developed. It is also a challenge toward the developer as each of the functional requirement must have its flow.

The prototype allows the developer to create more than one test case as it may create differences in the possible graphical user interface design. The prototype would run a similarity index for each of the test cases to ensure the integrity of it. The similarity index includes the possible design and the involvement of the stakeholder. The prototype is designed in such a way it would not be allowed multiple same functional requirements to be input within the same project. The developer needs to confirm each of the functional requirements before they proceed with the measurement. The unconfirmed functional requirement allows the developer to foresee any following possible functions than the proposed system may have. To overcome this issue, the developer needs to define stakeholder involvement before adding any functional requirement. This allows the prototype to run the similarity index to ensure the defined stakeholder well presents each of the functional requirements. Aside from that, it also promotes the degree of uniformity of the use of stakeholders.

The use of natural language to define the functional requirement may cause ambiguity. The functional requirement should not contain any vague detail. Requirement Boilerplate has been adopted as it helps to provide a semiformal structural template [9]. The developer tends to use informal language to write up the functional requirement; it is prone to the ambiguous and inconsistent. This is due to the natural language is a free text, and no rule applied. Corpus contains a list of vague words also adapted into the prototype to lower the ambiguity percentage for each functional requirement. Aside from that, the corpus includes a list of term represent the possible datatype also adapted which help in restricting the word used in describing the functional requirement.

III. RESULTS AND DISCUSSION

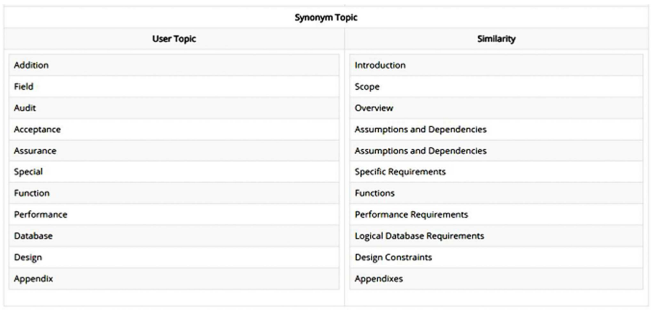
SRS from accounting domain had been selected as a case study. The SRS in .docx format was inserted as input. The prototype was scanned, extracted, and stored in the database. Extracted text was undergone cleansing process to increase the reliability of pattern-matching process. The prototype was run through the pattern-matching to gather the possible similar topic on the document. Number of similar topics compared against IEEE 830 table of content was calculated. The developer can view the possible synonym topic. This allow the developer to have the possibility to amend the SRS to comply with the standard. The prototype was used to generate the graph to show the number of data on the number of topics complied with the standard, the number synonym topic, and the possible number of topics after amendment. Based on the case study provided, the snapshot result of identified topic is shown in Fig. 2 below. The list topic identified similar to the IEEE 830 table of content is on the left of the snapshot table, while the list identified possible synonym topic is on the right side of the snapshot table.


IEEE Topic Specification

IEEE Topic	Synonym Topic
Introduction	Addition
Purpose	Field
Scope	Audit
References	Acceptance
Overview	Assurance
Overall Description	Special
	Function
	Performance
	Database
	Design
	Appendix

Fig. 2 List of identified topics

The identification of the IEEE topic is straight forward. For the list of synonym topic, the system would suggest a possible similar meaning with the IEEE topic. The snapshot list of recommended related issues to the IEEE topic shown in Fig. 3.



Synonym Topic	
User Topic	Similarity
Addition	Introduction
Field	Scope
Audit	Overview
Acceptance	Assumptions and Dependencies
Assurance	Assumptions and Dependencies
Special	Specific Requirements
Function	Functions
Performance	Performance Requirements
Database	Logical Database Requirements
Design	Design Constraints
Appendix	Appendixes

Fig. 3 List of synonym topic with IEEE 830 Table of content similarity

The list of similarities shows two possibilities. If the SRS topic detects as part of the topic in the IEEE 830 table of content and if the user topic detected as a synonym topic and the similar topic in the IEEE 830 table of content already exist, then it would be treated as an additional topic. Else the system suggests the possible similarity to the IEEE 830 table of content.

It is shown in Figure 2 that the topic of the “Introduction” is already detected. As in Figure 3, the topic of “Addition,” it is similar to Introduction in the IEEE table of content, then are treaded as an additional topic. The list in Figure 3 shows that there is the possibility of more than one topic define by the assessed SRS are under a similar topic. This may help the developer to consider either grouping it under the suggested topic, as in Figure 3 or ignore it. If the proposed topic has not detected in Figure 2 under column IEEE topic, then it is advisable to amend it as suggested in the similar column in Fig. 3.

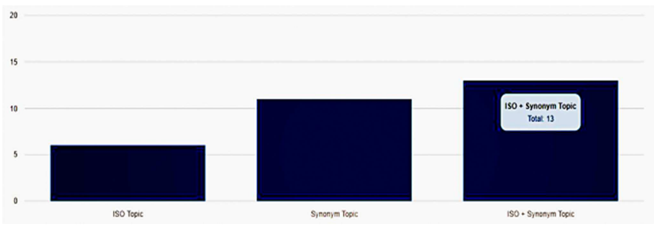


Fig. 4 Graph number of topics

The graph in Fig. 4 above shows the number of topics detected in the case study. The first column shows the number of topics detected based on the IEEE 830 table of content. The second column shows the number of synonym topic detected. The third column indicates the number of the topic after the amendment. If the developer amends the detected synonym topic based on suggested in column similarity in Fig. 3, the possible maximum number would increase. Based on the case study, the number of IEEE topics detected is 6, the number of synonym topics detected is 11, and the possible number of the topic after an amendment is 13. The measurement of the SRS structure is based on the number of the topic from the IEEE 830 table of content detected. Based on the case study without any amendment percentage of the completeness properties are 26%.

Meanwhile, assessment of the functional requirement started with the functional requirement document, which extracted manually from the assessed SRS. Those documents must be in “.docx” format. The document then inputted into the prototype. The prototype extracted the functional requirement, and the cleansing process was done. The final product would be a functional requirement that is refined into RB template. Before the text cleansing process started, the developer is required to define the possible stakeholder of the system. Proper define stakeholder ensures there is no conflict in defining its role. The developer should use a semi-formal template, as discussed in Section II, if needed to insert the functional requirement manually into the prototype. This is due to the system are designed to achieve an accurate result based on the Requirement Boilerplate template.

The initial inserted functional requirement is in the unconfirmed phase. The unconfirmed functional requirement would not affect the measurement. Once confirmed, the confirmed functional requirement was undergone by automated measurement. Each of the functional requirements should have at least one test case. The prototype allows the developer to have created a test case more than once for each functional requirement. This allows the developer to choose which suitable test case that can be implemented. The summarization of the quality properties of the functional requirement for the case study shown in Figure 5 below.

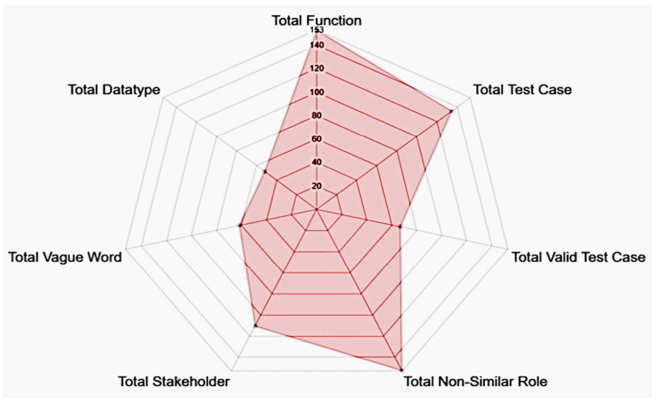
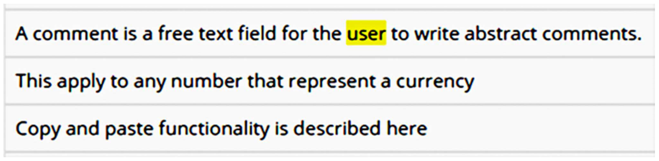


Fig. 5 Graph summarize quality properties for functional requirement

For instance, based on the case study, the total number of 153 functional requirements inputted into the system. As stated in the previous section, the prototype only allows the stakeholder to be inputted before the functional requirement cleansing process started. Four numbers of stakeholders were inserted. A list of the stakeholder comprises a system, user, company, and debtor. The measurement of consistency properties focused on the total stakeholder and total non-similar role. This is due to only 111 number of functional requirements identified has the stakeholder similar as stated for the case study. As for the similarity, there is none of the functional requirements had a similar role.

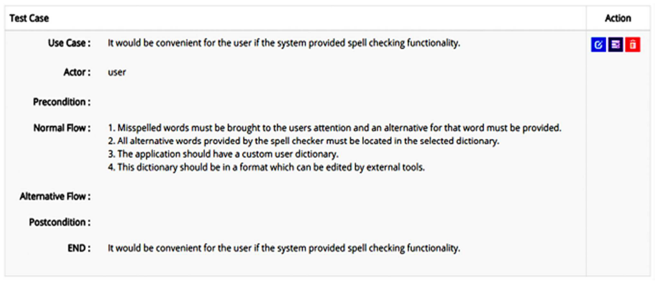

A comment is a free text field for the user to write abstract comments.

This apply to any number that represent a currency

Copy and paste functionality is described here

Fig. 6 Sample snapshot identified stakeholder

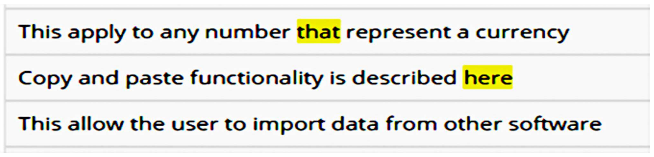
Figure 6 above shows a sample snapshot of how the stakeholder identified each functional requirement. From the sample shown in Figure 6, only the first sample of the functional requirement where the stakeholder had been highlighted. As for the other, there is no stakeholder identified based on the inputted stakeholder. The measurement of correctness properties focused on the total valid test case. Out of 153 number of functional requirements, only 135 number of test cases inserted. As the system run test on each of the inserted test case, only 67 number of the test case are valid. The validity of the test case is based on the presence of the actor in the actor segment and the similarity of the identified actor with the normal flow in the normal flow segment. The sample of valid test cases shown in Figure 7 below. The test case is considered valid if the actor present and the normal flow is identified with the presence of the similarity with the actor.



Test Case		Action
Use Case :	It would be convenient for the user if the system provided spell checking functionality.	
Actor :	user	
Precondition :		
Normal Flow :	1. Misspelled words must be brought to the users attention and an alternative for that word must be provided. 2. All alternative words provided by the spell checker must be located in the selected dictionary. 3. The application should have a custom user dictionary. 4. This dictionary should be in a format which can be edited by external tools.	
Alternative Flow :		
Postcondition :		
END :	It would be convenient for the user if the system provided spell checking functionality.	

Fig. 7 Sample valid test case

Meanwhile, for the measurement of preciseness properties, it is based on total vague word and total datatype found in the functional requirement. Given that 62 number of functional requirements have a vague word. Moreover, only 52 number of functional requirements have a possible datatype. The sample of the functional requirement contains vague words shown in Fig. 8 below. The identification of the vague word would decrease the preciseness of the functional requirement.

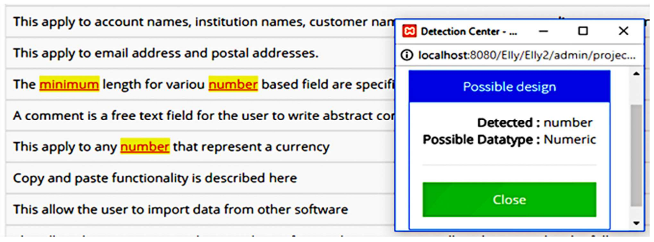

This apply to any number that represent a currency

Copy and paste functionality is described here

This allow the user to import data from other software

Fig. 8 Sample snapshot identified vague word

The identified word in Figure 9 is highlighted and clickable shown the term represent the possible datatype. The possible datatype design would be suggested once the identified word clicked.


This apply to account names, institution names, customer nan

This apply to email address and postal addresses.

The minimum length for variou number based field are specifi

A comment is a free text field for the user to write abstract cor

This apply to any number that represent a currency

Copy and paste functionality is described here

This allow the user to import data from other software

Detection Center - ...

localhost:8080/Elly/Elly2/admin/projec...

Possible design

Detected : number
Possible Datatype : Numeric

Close

Fig. 9 Sample snapshot identified datatype

The overall percentage for each quality properties is shown in Fig. 10. Based on the graph in Fig. 10, the overall percentage of the quality completeness properties is 26.1%, the quality correctness properties are 43.8%, the quality consistency properties 72.5%, and the quality preciseness properties is 37.3%. The overall percentage measurement for the structural requirement is 26%. Meanwhile, the overall measurement for the functional requirement is 51%. To conclude the overall measurement of the quality properties of the case study, it is 45%.

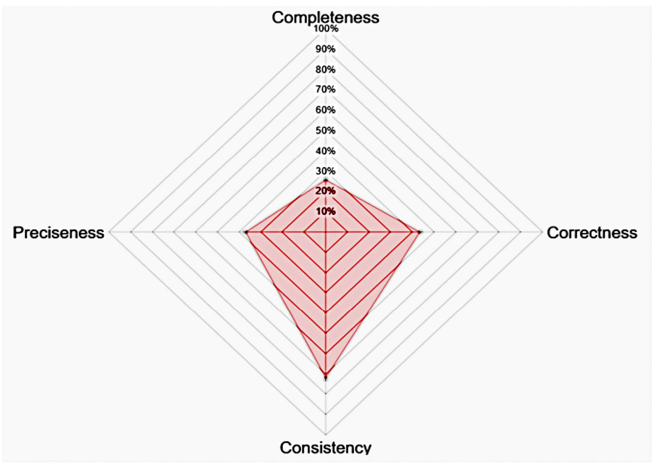


Fig. 10 Graph of overall percentage for each quality properties

The case study in the previous section yields up the overall result of 45% when measured using the prototype. Four quality properties are being assessed. Each of the qualities contributes up to 25%, and with four qualities, it resulted in a maximum of 100%. The measurement is divided into two categories, which were the structural and functional requirements. The result from the case study would be discussed further below in Table 2.

TABLE II
CASE STUDY RESULT

Quality Properties		Expected	Actual
Structural	Completeness	25%	6.5%
Functional Requirement	Correctness	25%	11%
	Consistency	25%	18.25%
	Preciseness	25%	9.25%
Overall Quality			45%

For the structural, 26.1% resulted in the completeness properties. This is due to only 6 number of topics out of 23 were detected similar to the IEEE 830 table of content. 11 number of detected topics had possible similarity toward the IEEE table of content. The result may increase up to 56.5% if the developer is willing to amend the SRS document.

Method to calculate the topic similarity with IEEE 830 topic is adopted in study [2]. However, this prototype automates the pattern-matching process which limit the human error.

Meanwhile, for the functional requirement, three quality properties contribute toward it. 43.8% resulted in correctness properties. This resulted from the missing stakeholder for each defined functional requirement. It is also would reflect on the defined test case. Weiger and Beatty [17] highlight on the capability of the developer to come out with possible test case for each functional requirement. Those resulted in the development of prototype component which validate the present of test case for each functional requirement. The validity of the test case depends on the presence of the stakeholder in the actor segment and the normal flow segment.

The quality properties of consistency resulted in 72.5%. The percentage is quite high due to no similar functional requirement detected. 42 number of functional requirements have no stakeholder as defined previously before the functional requirement can be added to the prototype. The identification of the stakeholder is important as the prototype is meant to adopt the Requirement Boilerplate template [3], [5], [8].

Lastly, for the result of the quality properties of preciseness, it is 37.3%. The low percentage has resulted from the detection of the possible vague word in the functional requirement. As the SRS is generated in natural language, the ambiguity is an issue. The adoption of the semi-formal template as such as Requirement Boilerplate can help reduce the possibility of vague word presence [3], [18]. Aside from that, the detection of a possible datatype is also low, which also resulted in a low percentage of preciseness in the case study. Method to identify the vague word also adopted due to it is part of ambiguity [19], [20]. Requirement Boilerplate template stress on the restriction of term usage in sentences. Pohl and Rupp [18], highlight on the information must be precise to serve as input in next development phase. Those promote the identification of term representing possible datatype.

The case study is an SRS document that is available on the website. If the real case scenario provided, the prototype is expected to evaluate the maturity of the SRS before proceed to the next phase. The prototype reflects the importance of standard usage of the IEEE 830 table of content structure and implementation of Requirement Boilerplate. The prototype is intended to measure the general quality properties that exist in the SRS so different SRS domains can be evaluated using it.

IV. Conclusion

In this research, the usage of the web-based system to define the library for each quality properties was a significant challenge. Each of the libraries needs to be updated periodically to increase the reliability of the prototype. The separate table for each library needs to be built in the database. The pattern-matching between the library and the inputted functional requirement, as well as the SRS document, need to separate so that the redundancy would not occur. The update function in the prototype gives flexibility toward the developer for amendment.

Aside from that, the rules for each of the quality properties have been well defined to fit the quality to be assessed. However, the amendment of the correctness rules has been done due to restrictions of client involvement. The work concluded where the prototype had been developed to prove the concept or the rules define for each of the quality properties. Prototype built is designed where it can be used in a different domain. The quality properties to be assessed are general and not specific toward certain domains.

Acknowledgment

The authors would like to thank Universiti Malaysia Sarawak for providing the funding to conduct this research. Fundamental Research Grant Scheme FRGS/2/2014/ICT07/UNIMAS/02/1 supported this work

References

[1] J. Medeiros, A. Vasconcelos, C. Silva, and M. Goulão, “Quality of software requirements specification in agile projects: A cross-case analysis of six companies,” *Systems and Software*, vol. 142, pp. 171-194, Aug. 2018.

[2] A. Takoshima, and M. Aoyama, “Assessing the Quality of Software Requirements Specifications for Automotive Software Systems,” in *Proc. APSEC,* 2015, p. 393-400, Dec. 2015.

[3] E. Hull, K. Jackson, and J. Dick, *Requirement Engineering*, 3rd ed., New York: Springer, 2011.

[4] M. P. S. Bhatia, A. Kumar, and R. Beniwal, “Ontologies for Software Engineering: Past, Present and Future,” *Science and Technology*, vol. 9(9), pp. 1-16, Mar. 2016.

[5] A. Mustafa, W. M. N. W. Kadir, and N. Ibrahim, “Automated Natural Language Requirements Analysis using General Architecture for Text Engineering (GATE) Framework,” *Telecommunication, Electronic and Computer Engineering*, vol. 9(3-4), pp. 97-101, Aug. 2017.

[6] E. Stephen, and E. Mit, “Framework for Measuring the Quality of Software Specification,” *Telecommunication, Electronic and Computer Engineering*, vol. 9(2-10), pp. 79-84, Aug. 2017.

[7] *IEEE Recommended Practice for Software Requirement Specifications*, IEEE Std. 830, 1998.

[8] S. Ahmad, U. Anuar, and N.A. Emran, “A Tool-based Boilerplate Technique to Improve SRS Quality: An Evaluation,” *Telecommunication, Electronic and Computer Engineering*, vol. 10(2-7), pp. 111-114, Apr. 2018.

[9] E. Stachtiari, A. Mavridou, P. Katsaros, S. Bliudze and J. Sifakis, “Early Validation of System Requirements and Design Through Correctness-by-Construction,” *System and Software*, vol. 145, pp. 52-78, Nov. 2018.

[10] G. Lucassen, F. Dalpiaz, J. M. E. M. v. d. Werf, and S. Brinkkemper, “Improving agile requirements: the Quality User Story framework and tool,” *Requirement Engineering*, vol. 21(3), pp. 383-403, Sep. 2016.

[11] N. A. Moketar, M. Kamalrudin, M. M. Yusof, and S. Sidek, “A study of generating abstract test for requirements validation among requirements engineers,” *Theoretical and Applied Information Technology*, vol. 95(7), pp. 1381-1388, Apr. 2017.

[12] R. Kazmi, D. N. A. Jawawi, R. Mohamad, and I. Ghani, “Effective Regression Test Case Selection: A Systematic Literature Review,” *ACM Computing Surveys (CSUR),* vol. 50(2), pp. 1-32, June 2017.

[13] S. Maalem, and N. Zarour, “Challenge of validation in requirements engineering,” *Innovation in Digital Ecosystems*, vol. 3(1), pp. 15-21, June 2016.

[14] M. Zhang, T. Yue, S. Ali, B. S. O. Okariz, R. Norgre, and K. Intxausti, “Specifying uncertainty in use case models,” *Systems and Software*, vol. 144, pp. 573-603, Oct. 2018.

[15] C. S. Gebizli, and H. Sözer, “Automated refinement of models for model-based testing using exploratory testing,” *Software Quality*, vol. 25(3), pp. 979-1005, Sep. 2017.

[16] P. A. Laplante, *What Every Engineer Should Know About Software Engineering*, 1st ed., Boca Raton, Florida: CRC Press, 2007.

[17] K. Weiger, and J. Beatty, *Software Requirement*, 3rd ed., Redmond, Washington: Microsoft Press, 2013.

[18] K. Pohl, and C. Rupp, *Requirement Engineering Fundamentals*, 2nd ed., Santa Barbara, United States: Rocky Nook, 2015.

[19] A. Nordin, N. H. A. Zaidi, and N. A. Mazlan, “Measuring Software Requirements Specification Quality,” *Telecommunication, Electronic and Computer Engineering*, vol. 9(3-5 Special Issue), pp. 123-128, Aug. 2017.

[20] A. O. J. A. Sabriye, and W. M. N. W. Zainon, “An approach for detecting syntax and syntactic ambiguity in software requirement specification,” *Theoretical and Applied Information Technology*, vol. 96(8), pp. 2275-2284, Apr. 2018.

[21] E. Serna, O. Bachiller, and A. Serna, “Knowledge meaning and management in requirements engineering,” *Information Management*, vol. 37(3), pp. 155-161, June 2017.

[22] H. Ahmed, A. Hussain, and F. Baharom, “Current Challenges of Requirement Change Management,” *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 8(10), pp. 173-176, Dec. 2016.