Analyzing the Impact of Assessing Requirements Specifications on the Software Development Life Cycle

Samah W. G. AbuSalim[1], Rosziati Ibrahim[1(✉)], Salama A. Mostafa[1], and Jahari Abdul Wahab[2]

[1] Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Johor, Malaysia
samahwgabusalim@gmail.com, {rosziati, salama}@uthm.edu.my
[2] Department of Engineering R&D, SENA Traffic Systems Sdn. Bhd., Kuala Lumpur, Malaysia
jahari@senatraffic.com.my

Abstract. Developing an efficient and quality Software Requirements Specification (SRS) is based on software quality characteristics assessment such as completeness, consistency, feasibility and testability. These characteristics or attributes provide reasonably accurate predictions about system-free bias requirements and hidden assumptions and limit subsequent redesign. They additionally give realistic estimates for costs, risks, and timing of the product. This paper aims to identify possible rules and methods for measuring SRS quality in order to help the engineers to improve the quality of their SRS. The impact of these rules and methods on the software development lifecycle is also reviewed. In this paper, some methods of SRS quality assessment were analyzed from the literature and how to measure the impact of these SRS quality assessment methods on the software development lifecycle are also presented.

Keywords: Requirement Engineering · Software Requirements Specification (SRS) · SRS quality assessment

1 Introduction

Requirement Engineering (RE) is a process of organized and disciplined techniques to handle the identification and management of software products developments and achievement of goals. In SE, Software Requirement Specification (SRS) is considered as the most important factor and outcome of the RE process [1]. Without exception, the process of SRS development is considered a crucial process to get favorable results of a medium to large project [2]. SRS extraction and analysis are performed throughout the first stages of the software development life cycle. The SRS contains a group of activities that are gathering, analyzing, specifying and validating users' needs in a document that is written in a natural language [3]. Hence, the main goal of the SRS is to attempt to fully satisfy users' needs [4]. Therefore, several methods and techniques for SRS development have been used to extract these users' needs depending on the

O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12254, pp. 632–648, 2020.
https://doi.org/10.1007/978-3-030-58817-5_46

software complexity [5]. That is a set of documentation that captures the complete description of the features and properties of the software product. It has many advantages to the developers as it lays out the serviceable and no serviceable specification of the product, defines project scope, minimizes development effort and eliminates any confusion or misunderstanding on the initial stage [6]. An important role of difficulties for SRS is because of complex writing structures that describe the requirements that protect the features of good SRS [7]. A poor specification of software requirements can lead to failed quality of products because any product's quality depends on SRS quality itself.

Software Quality Assessment (SQA) used to protect the quality of software delivered by observing the methods and processes of software engineering that ultimately results, or at least gives confidence, in the quality of software products. SQA expands on whole SDLC which is depends on the design of software, coding, testing, and release management. SRS quality evaluation is very critical to evaluate the quality level and faults in very starting steps of the SDLC process [6, 7]. The standard features of SRS play an important role to create the software with respect to cost-effectiveness and users' real needs. Some major good features of SRS such as (1) Correctness. The software requirement specification is said to be correct if the software has the ability to perform tasks that are actually expected from the system as specified in the requirements. (2) Completeness implies that all the segments of software are completely presented and developed accordingly. (3) Consistency which means that the requirement should be understood in exactly the same way if they are read by more than one person. Requirements in SRS are said to be consistent if the features do not clash with one another or with main features and aims. (4) Feasibility which includes performance and practical satisfaction developed by system verification. An SRS is said to be feasible if the features are done in the context of benefits the life cycle and it will also extend the cost of the life cycle.

Further, the remaining sections are as follow; Sect. 2 represents overview of Software Development Life Cycle (SDLC). Section 3 presents the research methodology including the research questions and answered. Conclusion, as well as future work, is mentioned in Sect. 4.

2 Software Development Life Cycle (SDLC)

Software Development Life Cycle (SDLC) is a systematic process, used in the development or maintenance of any software product [8]. This process aims at detailing the procedures and methods used to guide the work of the software development team, which is an essential part of building any software project. The life cycle of software systems consists of a series of successive phases, where the system is built evolutionary, that is, the system evolves after each phase until it reaches the final system required. These phases include feasibility study, analysis, design, implementation, testing and maintenance. It also consists of models and methodologies used by the software development team that provides a framework for the entire development process to be designed and managed. The purpose of designing and building a software system is to carry out a variety of tasks. The collection of tasks the program can carry

out also yields well-defined results based on complex calculations and manipulations. Therefore, overseeing the entire development cycle is a difficult and recurring challenge to ensure a high degree of quality and durability for the end product, as well as consumer acceptance. Therefore, to ensure the success of the system, it is necessary to use a comprehensive and systematic development process. At present, software developers are using two SDLC methodologies, namely conventional or traditional development and agile development. The following section will discuss and compare in detail these two methodologies and propose some improvements. [8, 9].

Traditional Software Development

This methodology is known as heavyweight because it relies on a set of heavy aspects which is to identify and record a complete set of requirements, followed by the development and examination of design and program performance. This methodology goes through four stages. The first stage is to determine the specifications and requirements of the project and specify the time needed to perform the different stages of the project development. After the requirements have been identified, the next step is to design and plan the program in the form of diagrams and models. From these things, we can anticipate the problems that the program may face during its development. The implementation process begins after the team approves the design and architectural plan of the project. The project continues in the development phase where the project is developed and built until the basic objectives are achieved. Sometimes, system development phases are divided into smaller tasks that are distributed between teams based on the skills of each team. Examples of this methodology include the Waterfall method, V-Model, and RUP. Sometimes the testing phase is performed in parallel with the development phase to ensure that problems are detected and resolved early. Once the project is completed and all the project requirements are implemented by the developers, the customer will engage in the project evaluation and feedback process and deliver the project after the client's satisfaction. The success of the project, which depends on the implementation of this method, depends on the identification of all requirements before the start of the project implementation, which means that any change during the development process can cause a problem. However, it is also easy to identify project costs and schedule resources and allocate them accordingly. It also helps to calculate project costs, schedule and allocate resources accordingly [7–10].

Agile Software Development

A set of steps through which to build software projects in several stages and short periods of time, where each stage generates a product distinct from the previous with additional characteristics. This process is often incremental and iterative until the entire project develops. This process focuses on the creation and formation of a team characterized by the process and intelligent accomplishment of the tasks, planning and continuous cooperation between team members. This helps to develop the project efficiently, deliver it on time and respond to any change. It is a conceptual framework that enhances the interactions expected during the development cycle. The key to the success of a software project is communication, flexibility and good analysis. that's why adopting agile approaches to software development. According to agile manifesto, the following four points are the main agile factors: (1) iterative development, (2) early customer involvement, (3) adaptation to change and (4) self-organizing teams.

There are currently six approaches known as agile methods of development, including crystal methodologies, dynamic software development method, feature-driven development, lean software development, scrum, and extreme programming [8–11].

Traditional Vs Agile Software Development
An important difference between agile and traditional development approaches is that in complex projects with unclear requirements, the former approach has the ability to deliver a result rapidly and cost-effectively. Agile approaches emphasize teams, work technology, customer interaction, and change response; whereas conventional methods emphasize agreements, schedules, procedures, records, and resources. Table 1 shows the differences in different aspects of agile and traditional approaches [7–12].

Table 1. Traditional vs agile software development [13]

Attribute	Agile	Traditional
User requirement	Iterative acquisition	Detailed user requirements are well defined before coding/implementation
Rework cost	Low	High
Development direction	Readily changeable	Fixed
Testing	On every iteration	After the coding phase completed
Customer involvement	High	Low
Extra quality required for developers	Interpersonal skills & basic business knowledge	Nothing in particular
Suitable Project scale	low to medium-scaled	Large-scaled

3 Research Methodology

The aim of this review paper is to analyze and discuss the quality assessments tools and methods used in SRS. The generic methodology used in this work consists of five main phases, which are research questions, literature review, filter papers, data extraction and result and discussion as shown in Fig. 1.

3.1 Definition of Research Questions

Software Requirements Specification is a way to gather and maintain user requirements. A good SRS should be clear and correct. There are a few approaches to go about ensuring your SRS is strong and complete and Some approaches are also not flexible in terms of usage but require background knowledge before using them. The research questions focus on the identification of the most recent methods and tools used for SRS

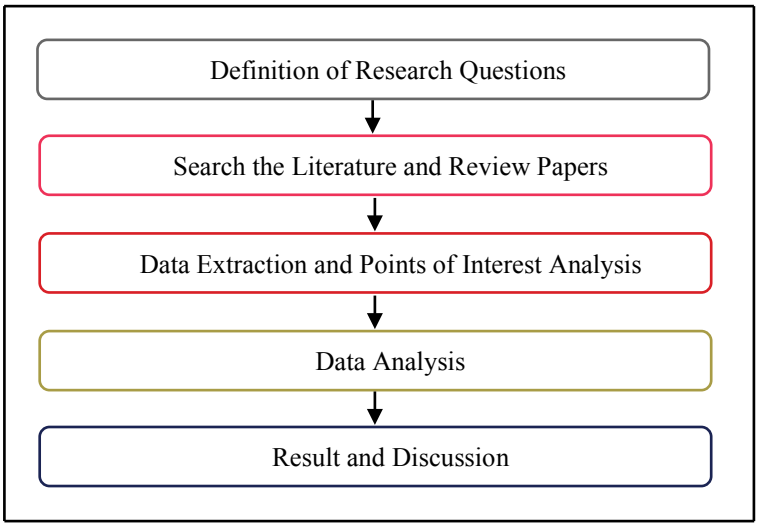

Fig. 1. Research methodology

quality assessment. Also, the impact of SRS quality on project success. Generally, this research aimed to answer the following questions:

1. What are the most recent SRS quality assessment methods?
2. What is the impact of SRS quality on project success?

3.2 Search the Literature and Review Papers

This research begins with conducting review on literature and previous research work regarding Requirement Engineering. Thus, we selected papers published in the ACM, IEEE, Science Direct, Springer, World Wide Web: Google Scholar and DBLP. In addition, other set of conferences and workshops were searched such as: ICSE, ICEET, AMCSE and ICIT. According to Brereton [14], these libraries are considered one of the most important resources and references in software engineering.

3.3 Data Extraction and Points of Interest Analysis

The goal of this phase is to design data extraction forms with which to accurately record the information obtained from the primary studies [5]. This paper aims at Investigate the current trends in SRS evaluation methods. The main points of interstate are identifying the used evaluation methods, the most used quality attributes and the main components of the related models. The data collected from 12 total papers was done by collating and summarizing the results of the primary studies.

3.4 Data Analysis

This section shows the answer of each research questions obtained after analysis the primary studies. The selected studies provided relevant evidence with which to satisfactorily answer the four RQs, as described below:

RQ1. What is the impact of SRS quality on project success?

During the practice of the software projects, these projects faced some problems and shortcomings and sometimes the failure of the application completely, resulting in significant delays and cost overruns. In fact, the software development life cycle of software systems has been plagued by exceeding budget, delayed or delayed deliveries and frustrated customers. In addition, the technology life cycle of software systems has been plagued by budget overrun, late or delayed delivery, and consumers have been frustrated. The Standish Group [8, 9] undertook a thorough investigation into this issue, they find that many projects do not deliver on time, do not deliver on budget, and do not deliver as planned or needed. The main reason for this is that the managers of the project do not smartly delegate the necessary number of staff and resources to the SDLC's different activities. For this reason, some phases of the SDLC are delayed and other phases are waiting for them to be completed without doing progress in the project. Thus, it creates a gap between the arrival and execution of projects, resulting in a failure to deliver a product on schedule, within the budget and at an acceptable quality level [9–12].

As mentioned earlier the basic task in a project begins with collection, analysis and definition of the requirements. In the requirements phase, a faulty requirement leads to specification errors and errors are induced from a requirement into the system specification. Then in the design phase, design errors occur which are induced from requirements and specification errors. These didn't stop here but in the implementation stage, program errors which in turn are induced errors from the requirement, specification and design occurs.

The worst part will happen however, in the testing and integration stage where known uncorrected errors and unknown errors happen and these leads to total failure of the system. This very showed clearly that failures can happen in each stage, mainly when the beginning starts with errors. Given the key role of specifications in the project life cycle and in assessing project success or failure, as a consequence, the overall quality of the software is directly associated with the requirements reliability. Project specifications should be of high quality to reduce failure of software products such as code and test cases [7, 12]. Arogundade et al. [15] examined and identified concepts that constitute a modeling technique for the safety risk assessment of the Information System (IS) and developed a conceptual model for the achievement of the safety risk assessment of the IS during the requirement analysis phase of the software process.

Ferguson and Lami [16] illustrate the value of quality in specifications for the occurrence of software development lifecycle deficiencies at later stages. Figure 2 uses information gathered by James Martin to highlight this argument, showing that over half of all defects are due to specification issues. Figure 3 shows that more than 80% of the rework effort can be traced to requirement-related defects.

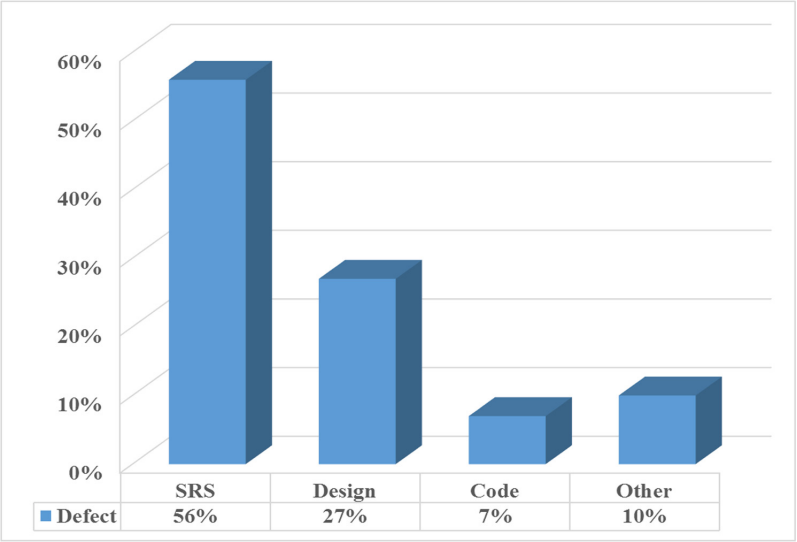

Fig. 2. Defects rate [17]

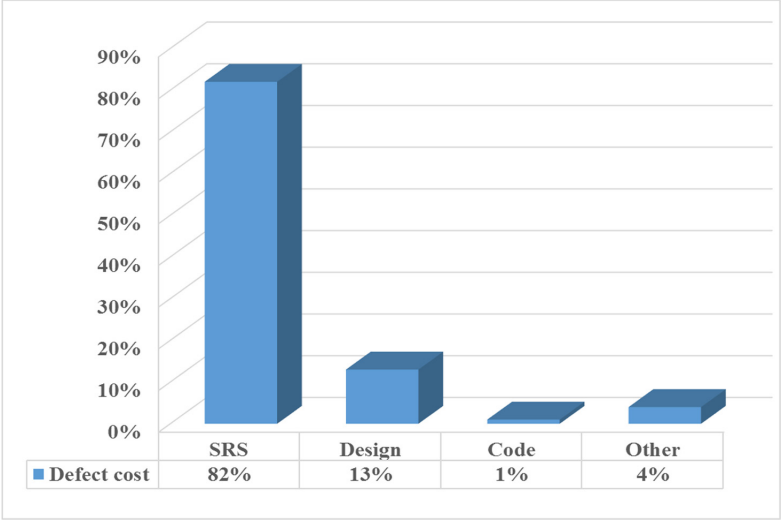

Fig. 3. Effort cost to repair the defects in Fig. 2 [17]

Wingers [18] discussed the need for quality requirements in order to minimize total project costs and mitigate risks that the project can face in the later stages of the software development life cycle. He shows the relative cost at various stages of the development process to fix a specification defect (Fig. 4). He says that issues with specifications can raise the cost of development by up to 50% and can add up to 80% to

rework costs due to mistakes found after delivery. By improving the quality with detecting and correcting defects at the level of requirements rather than developing them later at other stages of project development, the cost will remain low and success will be greater.

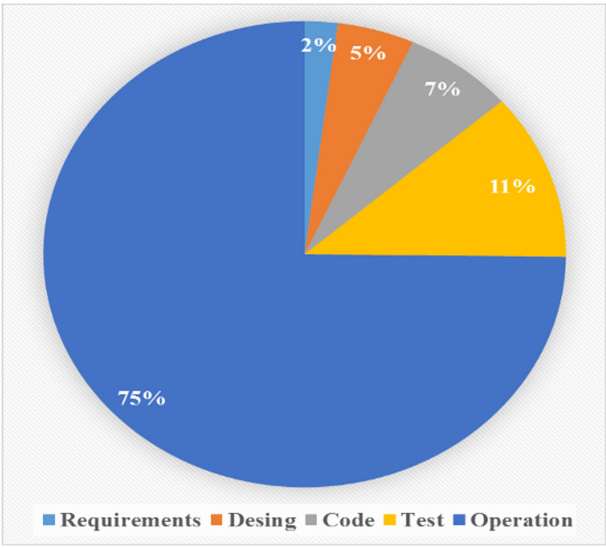

Fig. 4. The cost of correcting requirements defects [17]

RQ2. What are the most recent SRS quality assessment methods?

Several methods have been proposed to boost up the standard of SRS document. This paper analyzed some of the methods and tools used to evaluate SRS quality.

Antinyan and Staron [19] present an automated method to evaluate the understandability of the SRS document. Their method named (Rendex). Rendex assessed the understandability by using four indicators; complexity, Coupling, and size. The assessment results show that the Rendex achieved (73–80) % agreement compare with the manual result.

Nordin *et al.* [20] introduce a performance assessment tool for the SRS document based on two quality perspectives; Requirements Sentence Quality (RSQ) and Requirements Document Quality (RDQ) by developing the SRS Quality Checker tool as proof of the rules. Their paper concludes the methodology can be used as a framework and effort to measure the SRS quality.

Thitisathienkul and Prompoon [21] present an approach to how to analyses the use of common language in SRS, formation of documents, and overall standard of documents by making applicable the process evaluation model as a key tool for assessment of quality and standard of SRS also includes these two techniques to the evaluation process and measurement information. in this study, they conclude that only 3 features of specification requirement software are unambiguous, valid and changeable. Results

of this study that faults in the standard can be removed by having a discussion which could be useful as a technique for future development and quality assessment of upcoming SRS.

Ali *et al.* [22] have developed a distinctive approach to increase the standard of the document. Their technique comprises of four processes i.e. Parsing Requirement (PR), Requirement Mapping using Matrix (RMM), Addition of demands in specification requirement software template and independent inspection. PR will get the required inputs from the Requirement Engineering Process after the implementation; requirements will be achieved by completing the rules of ontology. Stakeholders concerns could be saved by RMM that will be formed to decrease ambiguousness and errors. Previous results will be added to IEEE quality organize. Independent inspection will be done to cross-check the customers and SRS demands. Inspection model of SRS will be used to assigning Total Quality Score (TQS) the third party will submit a report in detail to a group of Requirement Engineers (RE).

Ahmad *et al.* [23] present an evaluation of a boilerplate technique with the assistance of a tool-based prototype to enhance the Software Requirements Specification (SRS) quality in terms of comprehensibility, correctness, and consistency. The value behind this boilerplate is to ease the process of discovering necessary specification for a common information management system and convert these specifications in SRS. Outcomes of this study present that tools based on common techniques enhance completeness, correctness, and consistency of requirements in SRS.

Stephen and Mit. [24] propose a platform to measure the quality of both structure and functional requirements in SRS. The SRS includes information to make it confirmed with the standard of software. Measurement proposed based on four quality properties namely preciseness, consistency, completeness, and correctness. The completeness properties used a minimum standard IEEE 830 to evaluate and measure the SRS. In the meantime, it is suggested to use the characteristics of consistency, correctness, and accuracy to measure the functional requirement in the document. The general SRS standard measurement calculated based on all standard characteristics. The rules and formula for computing the SRS quality are embedded in the proposed framework which is a basis for a platform for assessing the software quality.

Da Silva [25] proposes an automated validation strategy that can assist with sufficient tool support to alleviate some of these limitations and therefore improve the quality of the specifications of requirements, in particular, those are related to consistency, completeness and unambiguity. Their study extends the RSLingo strategy by considering that the demands in RSL-IL are automatically extracted from the specifications of the natural language or specifically generated by customers.

Yaremchuk *et al.* [26] mentioned ways to enhance the correct demands between the SRM frameworks. This also includes difficult method. To check the complexity of method RCM metric is used. It is handled by prioritizing the requirements and based on complexity. This method is more helpful as compared to check and verify the whole process. Defectiveness and correctness enhanced by applying the improved requirements. Table 2 shows a summary of the methods and tools used to evaluate the quality of the SRS document.

Aguilar et al. [27] enhance the Model-Driven tool, named the WebREd-Tool, in order to enable the web application designer with the NFR specification to make better

Table 2. Summary of the analyzed methods and tools.

Author	Quality attribute	Proposed method	Strength	Limitation
Antinyan and Staron [19]	Complexity Coupling Size	They introduce a new tool called Rendex. The tool is relying on four internal quality measures of the SRS document	Rendex tool produces to facilitate the readability and understandability of the requirement document.	Rendex requires more evaluation in order to grasp its generalization possibilities across different product sizes and domains.
Nordin *et al.* [20]	Requirements Sentence Quality (RSQ) Requirements Document Quality (RDQ)	The proposed tool can be used for demonstrating how the rules were implemented to measure SRS quality	Their tool benefit from the performance assessment of SRS for the attributes applied (1) requirements sentence quality (RSQ) (2) requirements document quality (RDQ) or even to be used in pre-review sessions	Their work is incomprehensive because it is not possible to automate certain quality attributes
Thitisathienkul and Prompoon [21]	Unambiguous Verifiable Modifiable	The suggested approach can be used to critically assess the quality of the SRS to use natural language in the document's overall quality to display the document's value and the defects that arise during the specification process	Measure and report the quality level and the document deficiency portion. SDLC documents identified metrics, validity requirements, performance standard and possible enhancement parts are registered and stored	The insistence on unambiguous and verifiable features due to the layout of the report is directly linked to the modifiable function. Other features are difficult to verify and require human judgment

(*continued*)

Table 2. (*continued*)

Author	Quality attribute	Proposed method	Strength	Limitation
Ali *et al.* [22]	Ambiguities Completeness Correctness Verification Validation and inspection	The proposed solution will reduce the negative factors which can affect the quality performance of software products directly or indirectly	Improves the reliability of the report, increases the number of times needed, ensures completeness of the requirements and tracks the incorrectness of the requirements and continues conditionally after correcting requirements	In the method presented, each step of the system presented functions within its own limitations
Ahmad *et al.* [23]	Comprehensibility Correctness Consistency	The proposed boilerplate technique can improve all three SRS quality attributes	Improves the SRS quality in comprehensibility due to guided sentence structure based on generic essential system functionality (information management system) which helps to reduce ambiguity	Their research is a focus on the second perspective as the boilerplate provides a guide to having essential requirements. And focuses only on internal consistency which is included in the SRS
Stephen and Mit [24]	Preciseness Consistency Completeness Correctness	The suggested framework demonstrates the flow of data and measured the structure and functional requirement	The completeness properties are evaluated on the basis of the structure and reliability, quality and accuracy of the report depending on the functional criterion	Their research focuses only on four quality properties that can be measured as early as the documentation stage of the specifications, which was preciseness, correctness, consistency, and completeness

(*continued*)

Table 2. (*continued*)

Author	Quality attribute	Proposed method	Strength	Limitation
Da Silva [25]	Consistency Completeness Unambiguousness	Their approach helps to mitigate some of the SRS documents limitations, particularly with regard to inconsistency, incompleteness, and ambiguousness	Their RSLingo approaches provide (1) a language for defining linguistic patterns that frequently occur in requirements specifications written in natural language (the RSL-PL language)	RSLingo does not provide yet any guarantees that the RSL-IL specifications have the required quality
Yaremchuk *et al.* [26]	Correctness	The proposed method makes it possible to boost the correctness of specifications by finding a greater number of resource-restricted defects	The approach allows the correctness and defectiveness of specifications to be improved and the achieved index of correctness and defectiveness to be evaluated	Their methodology allowed the quantity of observable actual and potential defects to be increased by 9% on average compared to the total verification of the complexity of the requirements
Aguilar et al. [27]	Non-Functional Requirements (NFRs)	Presented a Model-Driven tool, named WebREd-Tool, extending the requirements metamodel with a NFRs classification	The proposal offers several advantages such as including the specification of Non-Functional Requirements from the requirements analysis stage considering the design decisions from the initial stages of the Web application development process	The proposal supports an automatic derivation of Web conceptual models from a requirements model by means of a set of transformation rules, the derivation of the Web application source code is still in development

design decisions and also to use it to verify the consistency of the final web application. Their proposal promotes the automated derivation of web conceptual models from the requirements model by means of a set of transformation laws.

Table 2 shows that the most evaluated quality attributes are consistency, completeness and correctness. It for the shows that it is difficult to automating the quality attributes as they are written in natural languages and need human experts. The automation of quality attributes evaluation required advanced NLP algorithms such as Text Normalization, Stemming and Lemmatization, Topic modeling, TF-IDF algorithm and Naive Bayes algorithm.

3.5 Discussion and Remarks

The Natural Language (NL) is still the primary way to write the SRS document. The SRS document written by using the NL is considered simple and understood by the stakeholders and developers due to these document does not need specific knowledge and effort. However, the NL still poses different issues like ambiguity and imprecise. The requirements written in a poor way have a negative impact on the overall project. Ambiguous or incomplete requirements require an additional effort in the different stage of the project. Finally, bad requirements lead to misunderstanding and produce the wrong product. To overcome these problems many researchers, attempt to improve the quality of the SRS document, the researchers are scattered into three main ways to evaluate the quality of the requirements; fully automated, semi-automated and non-automated (expert) [28, 29]. Figure 5. Shows the SRS assessment architecture with different automation levels.

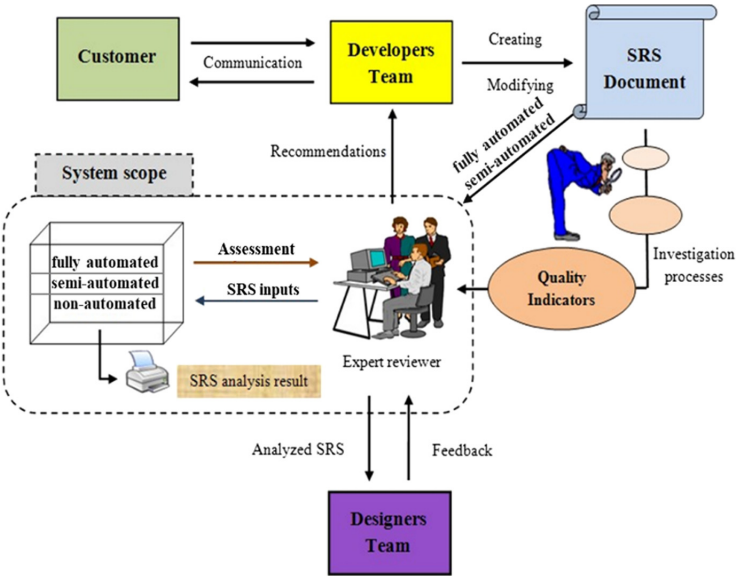

Fig. 5. The SRS assessment architecture with different automation levels

The fully automated way totally depends on the machine to complete whole steps of the assessment (without interacting by expert), the semi-automated the expert and machine are interacted between each other to produce the final assessment of the document, on the other hands, the non-automated is totally depends on the expert person to produce the final decision of the quality of the document. However, the expert still suffering from limited knowledge on the different domains, high cost, and consume time. For this reason, the researchers give more attention to produce an automated tool to solve the above-mentioned problems [10], [30–34]. Some of the advantages of the automated review are listed below:

The Automated Review Is Cheap and Fast: One of the main challenges that facing the projects is assessing the quality of the document by using manual review, this method has a high cost that derived thorough investigation. Consequently, the mechanism that delivers response with the free cost is a promising advantage as well as, the speed of response.

The Automatic Reviews are Consistent: Almost the expert reviewers are inconsistence because they effected by different private factors such; the state of his mind, or the current inputs of the reviewers. While the automated method is considered as a consistence.

Several methods and tools are used to assess the quality of SRS documents (in an automated manner) based on various quality SRS attributes and indicators like; consistency, completeness, complexity, and unambiguousness. These have been highlighted in Table 2. Since there are many resources for collecting knowledge and are used to identify requirements, there is no lack of knowledge and methods. But the problem is that many companies are unable to provide software products that satisfy the actual customer requirements due to flaws frequently found in SRS. Many studies concentrated on several quality characteristics. That can be evaluated as soon as the documentation phase of demands, which were preciseness, correctness, consistency, and completeness. Based from SRS, the system or software can be implemented. Once requirements are captured properly, the implementation is very straightforward. The SRS can also be used for formalization prior to implementing the system. Examples of research for formalization includes [35, 36].

4 Conclusion and Future Work

The project could be successfully completed if the required quality specification is clear to the team for development. Measuring the quality of SRS is based on software quality attributes like traceability, consistency, and completeness. This paper has analyzed several techniques and tools used to measure the SRS quality according to different quality attributes. Some researchers determined the SRS quality by examining different quality attributes such as accuracy, correctness and unambiguous. In the future, we can analyze and measure other quality attributes that have limited research coverage like cost. It will help to cost saving for persons that detection and correction of requirements failures.

Acknowledgement. This project is funded by the Ministry of Education Malaysia under the Malaysian Technical University Network (MTUN) grant scheme Vote K234 and SENA Traffic Systems Sdn. Bhd.

References

1. Aguilar, J.A., Zaldívar-Colado, A., Tripp-Barba, C., Misra, S., Bernal, R., Ocegueda, A.: An analysis of techniques and tools for requirements elicitation in model-driven web engineering methods. In: Gervasi, O., et al. (eds.) ICCSA 2015. LNCS, vol. 9158, pp. 518–527. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21410-8_40
2. Pekar, V., Felderer, M., Breu, R.: Improvement methods for software requirement specifications: a mapping study. In: 2014 9th International Conference on the Quality of Information and Communications Technology (2014)
3. Aguilar, J.A., Zaldívar-Colado, A., Tripp-Barba, C., Espinosa, R., Misra, S., Zurita, C.E.: A survey about the impact of requirements engineering practice in small-sized software factories in Sinaloa, Mexico. In: Gervasi, O., et al. (eds.) ICCSA 2018. LNCS, vol. 10963, pp. 331–340. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95171-3_26
4. Aguilar, J.A., Garrigo's, I., Mazo'n, J.N., Trujillo, J.: An MDA approach for goal-oriented requirement analysis in web engineering. J. UCS **16**(17), 2475–2494 (2010)
5. Zamudio, L., Aguilar, J.A., Tripp, C., Misra, S.: A requirements engineering techniques review in agile software development methods. In: Gervasi, O., et al. (eds.) ICCSA 2017. LNCS, vol. 10408, pp. 683–698. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62404-4_50
6. Souza, R.G.M., Stadzisz, P.C.: Problem-based software requirements specification. Revista Eletrônica de Sistemas de Informação **15**(2) (2016)
7. Jani, H.M., Mostafa, S.A.: Implementing case-based reasoning technique to software requirements specifications quality analysis. Int. J. Adv. Comput. Technol. **3**(1) (2011)
8. Leau, Y.B., Loo, W.K., Tham, W.Y., Tan, S.F.: Software development life cycle AGILE vs traditional approaches. In: International Conference on Information and Network Technology, vol. 37, no. 1, pp. 162–167 (2012)
9. Carlson, N., Laplante, P.: The NASA automated requirements measurement tool: a reconstruction. Innov. Syst. Softw. Eng. **10**(2), 77–91 (2013). https://doi.org/10.1007/s11334-013-0225-8
10. Jubair, M.A., Mostafa, S.A., Mustapha, A., Hannani, A., Hassan, M.H:. Fully automated quality assessment metrics for software requirement specifications, pp. 177–187 (2019)
11. Bassil, Y.: A simulation model for the waterfall software development life cycle. arXiv preprint arXiv:1205.6904 (2012)
12. Wallmuller, E.: Software Quality Assurance: A Practical Approach. Prentice-Hall BCS Practitioner. Prentice-Hall, Upper Saddle River (1994)
13. Stoica, M., Mircea, M., Ghilic-Micu, B.: Software development: agile vs. traditional. Informatica Economica **17**(4) (2013)
14. Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M., Khalil, M.: Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw. **80**(4), 571–583 (2007)
15. Arogundade, O.T., Misra, S., Abayomi-Alli, O.O., Fernandez-Sanz, L.: Enhancing misuse cases with risk assessment for safety requirements. IEEE Access **8**, 12001–12014 (2020)

16. Ferguson, R., Goldenson, D., Fusani, M., Fabbrini, F., Gnesi, S.: Automated natural language analysis of requirements and specifications. In: INCOSE (International Council on System Engineering) International Symposium (2015)
17. Elcock, A., Laplante, P.: Testing software without requirements: using development artifacts to develop test cases. Innov. Syst. Softw. Eng. **2**(3–4), 137–145 (2006). https://doi.org/10.1007/s11334-006-0009-5
18. Wiegers, K.E.: Software Requirements, Chapters 14-Appendix D
19. Antinyan, V., Staron, M.: Rendex: a method for automated reviews of textual requirements. J. Syst. Softw. **131**, 63–77 (2017)
20. Nordin, A., Zaidi, N.H.A., Mazlan, N.A.: Measuring software requirements specification quality. J. Telecommun. Electron. Comput. Eng. **9**(3–5), 123–128 (2017)
21. Thitisathienkul, P., Prompoon, N.: Quality assessment method for software requirements specifications based on document characteristics and its structure. In: 2015 Second International Conference on Trustworthy Systems and Their Applications (2015)
22. Ali, S.W., Ahmed, Q.A., Shafi, I.: Process to enhance the quality of software requirement specification document. In: 2018 International Conference on Engineering and Emerging Technologies (ICEET) (2018)
23. Ahmad, S., Anuar, U., Emran, N.A.: A tool-based boilerplate technique to improve SRS quality: an evaluation. J. Telecommun. Electron. Comput. Eng. **10**(2–7), 111–114 (2018)
24. Stephen, E., Mit, E.: Framework for measuring the quality of software specification. J. Telecommun. Electron. Comput. Eng. **9**(2–10) (2018)
25. da Silva, A.R.: Quality of requirements specifications: a preliminary overview of an automatic validation approach. In: Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 1021–1022. ACM, March 2014
26. Yaremchuk, S., Bardis, N., Vyacheslav, K.: Metric-based method of software requirements correctness improvement. In: ITM Web of Conferences AMCSE (2016)
27. Aguilar, J.A., Misra, S., Zaldívar, A., Bernal, R.: Improving requirements specification in WebREd-Tool by using a NFR's classification. In: Murgante, B., et al. (eds.) ICCSA 2013. LNCS, vol. 7973, pp. 59–69. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39646-5_5
28. Mostafa, S.A., Gunasekaran, S.S., Khaleefah, S.H., Mustapha, A., Jubair, M.A., Has-san, M. H.: A fuzzy case-based reasoning model for software requirements specifications quality assessment. Int. J. Adv. Sci. Eng. Inf. Technol. **9**(6), 2134–2141 (2019)
29. Mostafa, S.A., Gunasekaran, S.S., Khaleefah, S.H.: Integrating fuzzy logic technique in case-based reasoning for improving the inspection quality of software requirements specifications. In: Khalaf, M.I., Al-Jumeily, D., Lisitsa, A. (eds.) ACRIT 2019. CCIS, vol. 1174, pp. 503–513. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38752-5_39
30. Ibrahim, R., Saringat, M.Z., Ibrahim, N., Ismail, N.: An automatic tool for generating test cases from system's requirements. In: CIT 2007: 7th IEEE International Conference on Computer and Information Technology, Article number 4385193, pp. 861–866 (2007)
31. Femmer, H.: Automatic requirements reviews - potentials, limitations and practical tool support. In: Felderer, M., Méndez Fernández, D., Turhan, B., Kalinowski, M., Sarro, F., Winkler, D. (eds.) PROFES 2017. LNCS, vol. 10611, pp. 617–620. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69926-4_53
32. Ashraf, S., Khan, R., Iqbal, K., Chohan, R.: Quality software requirement specification (SRS) and suitable SDLC leads to quality software. J. Appl. Environ. Biol. Sci. **6**(4S), 137–146 (2016)
33. Sabriye, A.O.J.A., Zainon, W.M.N.W.: A framework for detecting ambiguity in software requirement specification. In: 2017 8th International Conference on Information Technology (ICIT), pp. 209–213. IEEE, May 2017

34. MacDonell, S.G., Min, K., Connor, A.M.: Autonomous requirements specification processing using natural language processing. arXiv preprint arXiv:1407.6099 (2014)
35. Aman, H., Ibrahim, R.: Formalization of transformation rules from XML schema to UML class diagram. Int. J. Softw. Eng. Appl. **8**(12), 75–90 (2014)
36. Aman, H., Ibrahim, R.: Formalization of versioning rules for XML schema using UML class diagram. J. Theoret. Appl. Inf. Technol. **95**(15), 3652–3661 (2017)