Computers and Electrical Engineering 96 (2021) 107445

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Metrics for software requirements specification quality quantification

M.R. Raja Ramesh *, Ch. Satyananda Reddy

Department of Computer Science and Systems Engineering, AU College of Engineering, Andhra University, Visakhapatnam, Andhra Pradesh, India

ARTICLE INFO

Editor: Dr. M. Malek

Keywords:
Software development life cycle
Software requirement specifications
IEEE quality metrics
Natural language processing
Software engineering
Text processing

ABSTRACT

The initial and crucial phase of the automation and software development is identifying requirements and documenting them in an appropriate format that denotes software requirement specification (SRS). The quality and productivity at different phases of the software development depend on requirements specification. The quality of the end product of software development is proportionate to the quality of SRS. Hence, estimating the quality of the SRS is essential. Though the numerous metrics have been defined in contemporary research, the IEEE standard metrics are considered authentic to scale SRS quality. This manuscript endeavored to redefine the IEEE standard metrics' measuring approaches to improvise SRS quality assessment. The experimental study exhibiting the significance and robustness of the proposed approach that compared to the contemporary contributions of the recent literature

1. Introduction

The most significant part of the engineering process for complex software systems is the software requirements. Stakeholders of the project would be allowed since they act as a medium of exchange for communicating their interests or requirements for implementation. For communicating the formal requirements, it is better to communicate in natural language with involved parties. Therefore, in the airbus, the requirements are written in the English language. Usually, both money and time spent on software system implementation might be impacted based on quality requirements. The work in [1] presents that it impacts the likelihood of project success as an outcome.

With this association, several efforts have been kept to determine, evaluating, and enhancing the quality of requirements [1]. Initially, the primary models have based on rules or policies formulated by experts in that particular domain. Most of the rules are lacking completeness, ambiguity, unstable, and other parameters such as validation, test, verification, and trace. Moreover, the vocabulary of the domain often misguides in understanding the SRS context. Further, these parameters turn the maintenance, version updates, and project realization into expensive in terms of money and process time.

For handling these problems, the evaluation of SRS towards its effectiveness is assessed before execution. Currently, the term machine learning is an interesting domain in decision making, predictive analysis, and suggesting systems. Here, tasks related to

Abbreviations: SRS, software requirement specification; NLP, natural language processing; SDLC, software development life cycle; NFR, non-functional requirements; QR, quality requirements; TF-IDF, term frequency-inverse document frequency; EHR, electronic health records; RFP, request for proposals; EM, expectation maximization; ASRD, ambiguous software requirement specification detection.

This paper is for special section VSI-rtcc. Reviews processed and recommended for publication by Guest Editor Prof. Sundhararajan.

* Corresponding author.
E-mail address: rajarameshphd2020@gmail.com (M.R.R. Ramesh).

https://doi.org/10.1016/j.compeleceng.2021.107445
Received 25 November 2020; Received in revised form 8 May 2021; Accepted 1 September 2021
Available online 29 October 2021
0045-7906/

natural language processing (NLP) like opinion mining, emotion recognition, and sentiment analysis were also having prominent research on machine learning techniques [2]. Since these techniques are data-driven approaches, the ML models do not require ongoing human expertise. As an alternative, if supplied with an adequate amount of inputs labeled, it learns to automatically identify related data patterns [3] for forecasting the label of specified input that is unlabeled. Therefore, the challenge of assessing the SRS quality has extensive scope to ML. Nevertheless, any machine learning model of supervised learning often misrepresents the decision because of improper training patterns. For addressing this problem, an optimum learning scheme is significant for implementing ML in predictive analytics [4].

The notation SRS, which is signified as software requirements specifications document reports all functionalities and abilities a software has to provide and states any pre-requisite confine through which the system has to stand. From the definition, the term requirement is stated as an objective, which one has to be met, whereas specification explains how this objective would be attained [5]. On the other hand, the specification document explains how particular tasks have to be done. Here, proactive involvement acts as a major role in quality assurance at the time of the system's requirements specification stage. Earlier, various researches have been examined that companies or industries need to pay fewer amounts for fixing the issues, which are identified at an early stage in the software development life cycle (SDLC) process [5]. For measuring the quality of the SDLC process, the list of quality features, which are expected the software requirements specifications, has to be compiled as presented in contribution [6]. Various required characteristics for the SRS document have been recognized in former contributions [6].

- *Scope of completeness* overall requirements specifications have to determine real-time circumstances and incorporate all required capability features.
- *Scope of consistency* No conflicting statements should be there among consistent requirements specifications.
- *Scope of accuracy* Proper requirement specification has to recognize the individual confines and conditions precisely and accurately for all the instances where the required capability might encounter and determine the response capability of such cases properly.
- *Scope to modify* Relevant requirements specifications should be clustered to modify, whereas requirements specifications that are not related have to be isolated.
- *Scope to rank* Based on their significance and stability, the ranking of Requirements specifications has to be performed.
- *Scope to trace* For attaining traceability, every requirement specification has to be recognized uniquely. The term uniqueness would be attained by using a logical and consistent strategy to assign every SRS specification.
- *Scope to interpret (unambiguous)* every specification can be interpreted only in one approach. The utilization of poor sentence or weak phrase structure has to be evaded.
- *Scope to understand* The requirement specification document is understandable when the meaning of every statement has been understood easily by the readers.
- *Scope to test* The requirement specification is testable in such a way either pass or fail.
- *Scope to verify* To verify, every requirement specification at one of the abstraction levels should be constant with other level abstraction.
- *Scope to validate* The valid requirement specification is the one, which has been examined, accepted, validated, understood, and approved by the participants of the project, engineers, customer representatives, and managers.

The above-stated characteristics are closely related to each other, while some do not exist without others. At the time of audit review and analysis of requirement specifications, various primitive indicators that offer some indication that required attributes existed or not existed were being associated with the attributes mentioned above of quality. This paper intends to devise robust measures to assess the ratio of quality of the software requirement specifications. Here, the SRS's quality quantification is carried by the enhanced measures proposed to IEEE standard SRS quality assessment metrics.

In this manuscript, Section 1 explores the Introduction of Requirements engineering (RE), Section 2 describes the related work, which comprises several specifications found in contemporary contribution. Section 3 discusses the research methodology of this contribution. Section 4 explains the results and experimental study of this work. Section 5 discusses the Conclusion, followed by references.

2. Related research

The significance of software quality requirements has been extensively identified; however, the term is ambiguity. The work in [7–10] presents that non-functional requirements (NFR) have been utilized for a longer time, where quality requirements have been pretended to replace, and ambiguity is evaded, which derives from negative definition with "non" prefix. However, sometimes, the term "quality requirements (QR)" is ambiguous with the term "requirements quality" that indicates specifications.

In contribution [10], three challenges have been recognized in introducing the IEEE software issue on the QR as (i) how to extract QRs, (ii) how to resolute trade-offs among QRs, and (iii) how to evaluate QRs. The other important challenge is how to implement QRs into products. Many of the QRs research concentrates on one or manifold of above stated four challenges such as the resolution of trade-off [11,12], measurement of QR [13,14], implementation of QR [15], and elicitation [9]. Nevertheless, before recognizing these issues, it might be worthy to grasp how QR was dealt with in practice and examine them [16]. Here, it explores the ways for representing, implementing, and eliciting QR. We require a model for recognizing where SRS quality requirements were mentioned and which each characteristic requirement class belongs to. When such detection is implemented easily, we can have a better view of how much quantity of QRs was defined when compared with functional requirements, how they were distributed over different classes such

as compatibility, security, performance, and many more, and further how they were structured in the overall specification document.

The work in [17] calls requirements sentences distribution across QR characteristics as a spectrum and project a model for identifying spectra. Nevertheless, their rule of differentiating QR from the FR is not obvious. Generally, they consider functional requirements statements such as printing should be assisted and arranged into quality such as interoperability. The work in [17] presents that they categorize the quality of requirements characteristics manually for executing the sort.

The process has been mechanized partially by utilizing a keyword-quality matrix [18]; however, the matrix is designed for every system with the help of a human, even though there is a probability of reusing the current one when the domain of application is similar. The work in [12] had done extensive research on QR analysis. They have considered greater than 2000 requirements for the company systems, and each requirement has been coded elaborately with characteristics and types and aggregated and examined them. However, these were based on the manual work; an automated way for classifying and detecting QR is more significant as stated in contribution [19]. They have utilized a weighted term frequency measure, usually similar to term frequency-inverse document frequency (TF-IDF), for computing the similarity among documents and categorizing the pre-requisites with such measure. MS students developed 15 requirement specifications and are used to assess their projected method and huge requirements document from the company.

The same work is followed. One is carried out by a similar group, implemented for healthcare domains [20]. The other instance is in [21], which is also implemented to the healthcare system. They have utilized eleven documents associated to EHR (electronic health records), incorporating two RFP (request for proposals) and two are open source manual projects. They categorize requirements by the k-nearest neighbor model. Further, they also assessed SVM and NB learning models.

The work in [22] projected a semi-supervised learning model. However, it begins with a set of labeled data, and then the

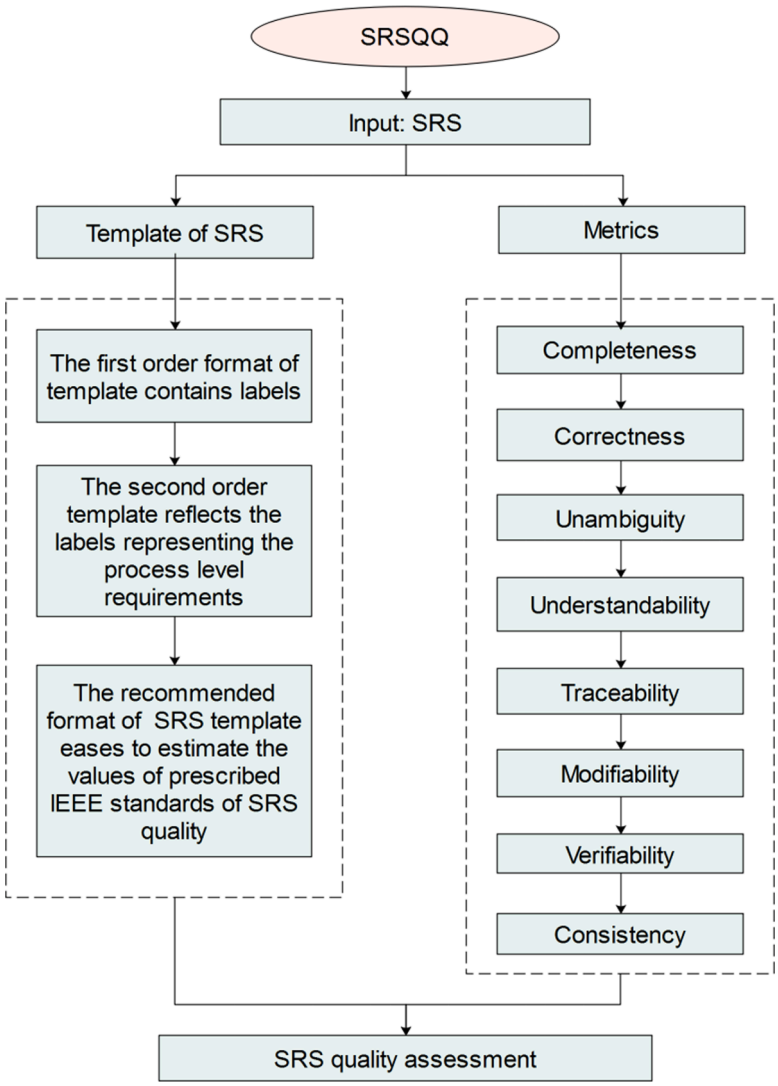

Fig. 1. The block diagram representation of SRSQQ.

Expectation maximization (EM) model has been utilized iteratively for augmenting the learning. The work in [19] utilized similar data accessible at the repository of PROMISE software engineering.

The term natural language processing (NLP) schemes have been extensively utilized for engineering requirements. However, many of them were related to SRS improvement and evaluation. One of the successful domains in implementing NLP in identifying ambiguity of phrases and words in SRS is explored in [23]. The work in [24] is also an instance of implementing NLP for using cases in identifying ambiguity. The characteristic domain is a recommendation of code [25]. The work in [26] presents that NLP has often been used to analyze traceability in RE community.

The contemporary methods "Ambiguous software requirement specification detection" (ASRD) [27], and "Stability prediction of the software requirements Specification" (SPSRS) [28] have portrayed methods, which combined the text mining and supervised learning approaches. However, these methods have focused on an automated approach to discover the quality measure, limited only to an estimation of ambiguity and stability.

`In order to address the issues noted with contemporary methods, this manuscript portrayed a novel approach that modularized the assessment of IEEE standards to scale the competency of the SRS document.

3. Methods and materials

Unlike contemporary methods, the proposal has derived the IEEE metrics of SRS quality assessment standards as more modular to the requirements collection. This section details the description of the SRS's recommended template and the assessment strategy of SRS quality metrics portrayed in this manuscript. The block diagram representation of SRSQQ is shown in Fig. 1.

3.1. Template of software requirement specification

The proposed format of software requirement specification denotes the following. The first order format of the template contains the labels denoting each requirement collected from the target, a list of corresponding inputs and expected outcomes of each requirement, sequences of the requirements, and sources of inputs. The second-order template reflects the labels representing the process level requirements, the inputs, and outcomes of the process level requirements. The SRS template's recommended format eases to estimate the values of prescribed IEEE standards of the SRS quality. The traditional SRS quality assessment approaches consider all SRS requirements as one unit to estimate the values of the quality standards. Unlike the traditional methods, the proposal considers the SRS into multiple modules to improve SRS quality assessment accuracy. The given SRS shall partition into multiple modules either by annotating by human resource or automated. Each set of requirements has dependency or interoperability to complete a certain task of the target software.

3.2. The metrics

The following description explores the improvisation of IEEE standards recommended to estimate the quality of SRS. The metrics completeness, correctness, unambiguity, understandability, traceability, modifiability, verifiability, and consistency have been redefined (more modular).

3.2.1. Completeness

The metric completeness denotes the description for every requirement labeled in the first order of the SRS template. However, having a description for every requirement is not enough to state the completeness. Since the description of a requirement shall include details related to functional and non-functional factors, properties of the inputs, and outcomes of the corresponding requirement. Hence, this contribution modularized the metric completeness of a requirement as (i) The ratio of requirements having a description, (ii) The ratio of requirements reflecting the description of functional factors, (iii) The ratio of requirements reflecting the description of non-functional factors, (iv) the ratio of requirements reflecting the description of input parameters, (v) The ratio of requirements reflecting the description of properties relating the outcomes, and (vi) The ratio of requirements obeying all measures of documenting format

3.2.2. Correctness

Every requirement shall represent something required of the system to be built; the other way, every requirement in the SRS shall contribute to some need satisfaction. However, the accuracy of this metric value is proportionate to the SRS quality estimation skills of the human resource, which is highly vulnerable to the false alarming of SRS correctness assessment. In order to address this constraint, the requirement's correctness estimation has modularized as (i) The ratio of requirements listed in one or more sequences of requirement patterns denotes that requirements are mandatory to complete the SRS one or more tasks, (ii) The ratio of requirements seeking available and eligible functional factors to use in the target domain of the proposed software, (iii) The non-functional factors of the requirements shall be valid under the system domain of the proposed software. Hence, the ratio of requirements with eligible non-functional factors has been considered estimating the correctness, (iv) The sources of inputs and formats of outcomes of the requirements shall be valid and eligible. Hence the frequency of requirements having accessible sources of input and eligible formats of outcomes has considered estimating the SRS's correctness, and (v) the violation of business rules by the inputs and outcomes of the requirements degrades the SRS quality. Hence, the ratio of requirements reflecting the inputs and outcomes fall under the business rules defined for target software has considered estimating the SRS's correctness.

3.2.3. Unambiguity

Every requirement denotes one interpretation (An SRS is unambiguous if and only if every requirement stated therein has only one possible interpretation). However, more than one interpretation of the requirement appears for different inputs or for different modules. Hence, the unambiguity has modularized as (i) The ratio of requirements having a unique interpretation, (ii) The ratio requirements having multiple interpretations with different formats of input, and (iii) The ratio of requirements having multiple interpretations related to different sequences or modules.

3.2.4. Understandability

The traditional definition of metric understandability is the description of a requirement that shall fall into the scope of all classes of people related to the target software. However, it is impractical to justify the perception of different classes of people related to that requirement. Hence, the understandability has modularized based on completeness and correctness, which are (i) The ratio of requirements stated as understandable, (ii) The frequency of requirements stated as understandable in relation to process flow, (iii) The frequency of requirements stated as understandable in relating functional factors, (iv) The frequency of requirements stated as understandable in relation to non-functional factors, and (v) The frequency of requirements stated as understandable in relation to inputs and outcomes of requirements.

3.2.5. Traceability

Documenting factors are critical to trace the requirements and corresponding descriptions. The Requirements well-referred to in descriptions, vice versa descriptions shall have a unique identity and well referred by descriptions are optimum to trace. In addition, requirements and corresponding descriptions well referred in baselined documents include system-level requirements specifications, statements of work (SOW), white papers, an earlier version of the SRS to which this new SRS must be upward compatible, and requirement specifications for other systems to which this system must interface. Hence the estimation of traceability has modularized as (i) The ratio of requirements referred to in descriptions, (ii) The ratio of requirements referred to in baseline documents, (iii) Frequency of requirements referred to in the description of system-level requirements, and (iv) Frequency of requirements referred in the description of statement of work.

3.2.6. Modifiability

The requirements having dependents and requirements dependent on other requirements are complexed to modify. However, the dependency of specific operable sources also detracts from the scope of modifiability. Hence, the estimation of the metric modifiability is modularized as (i) The frequency of requirements not dependent on other requirements, (ii) The frequency of requirements not having dependent requirements, and (iii) The ratio of requirements not dependent on specific operable sources.

3.2.7. Verifiability

Verifiability is poor if the descriptions are ambiguous, not worth to cost (ratio of excess cost), no sources to test, not possible to test, poor template measures, and fault references or references. The metric verifiability depends on two other metrics, non-ambiguous and traceability, since unambiguous descriptions of the requirements and rich referencing in SRS documentation (traceability) maximizes the verifiability of the SRS. In addition, the scope to perform testing and the availability of sources to perform testing improve the requirements' verifiability as (i) Unambiguity of the SRS, (ii) Traceability of the SRS, (iii) The ratio of requirements having scope to perform testing, and (iv) The ratio of requirements having sources to perform testing.

3.2.8. Consistency

The consistency of a requirement denotes no conflicts between the statements stated in the corresponding requirement. However, the statements of the requirement description with no conflicts scale only the internal consistency. An SRS shall be consistent concerning system requirements connecting the current requirements as an interface to other applications, descriptions projected in baseline documentation, statement of work, white papers, and an earlier version of the software requirement specification. The overall consistency of the software requirements specification has modularized as (i) The ratio of requirements not having conflicts between statements projected in requirement descriptions, (ii) The ratio of requirements not reflecting any conflicts in baseline documentation, (iii) The ratio of requirements not conflicting with system requirements, (iv) The ratio of requirements not conflicting with earlier versions of the SRS, and (v) The ratio of requirements stated in white papers with no conflicts.

3.3. Estimating the metric values

This section explores the mathematical model of the metric value assessment. The assessment performs in two phases; one estimates metric values at the modules level and measures overall document level metric values.

Let the notation$M = \{m_1,m_2,m_3,, m_{|M|}\}$denotes the set of modules of a count$|M|$, such that each module$m_i = \{r_1,r_2,r_3,..., r_n\}$represents a set of requirements of a different count, and a requirement shall exist in one or more modules. The values of the module level quality assessment metrics stated in the earlier section (Section 3.2) measures as follows:

3.3.1. The completeness

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$Begin // for each module$m_i$

$com_d(m_i) = dm_i{}^*[|m_i| + 1]^{-1}$// the notation $com_d(m_i)$ denotes the ratio of requirements of the module m_i having description; the notations dm_i,$|m_i|$ denote the number of requirements having descriptions and a total number of requirements in respective order.

$com_{ff}(m_i) = fdm_i{}^*[|m_i| + 1]^{-1}$// the notation $com_{ff}(m_i)$ denotes the ratio of requirements of the module m_i having a description for all required functional factors; the notations fdm_i,$|m_i|$ denote the number of requirements having descriptions of the required functional factors and a total number of requirements in respective order.

$com_{nf}(m_i) = ndm_i{}^*[|m_i| + 1]^{-1}$// the notation $com_{nf}(m_i)$ denotes the ratio of requirements of the module m_i having descriptions for all required non-functional factors; the notations ndm_i,$|m_i|$ denote the number of requirements having descriptions for required non-functional factors and a total number of requirements in respective order.

$com_{ip}(m_i) = idm_i{}^*[|m_i| + 1]^{-1}$// the notation $com_{ip}(m_i)$ denotes the ratio of requirements of the module m_i having a description of all required input parameters; the notations idm_i,$|m_i|$ denote the number of requirements having a description of all input parameters and a total number of requirements in respective order.

$com_{oc}(m_i) = odm_i{}^*[|m_i| + 1]^{-1}$// the notation $com_{oc}(m_i)$ denotes the ratio of requirements of the module m_i having a description of outcomes; the notations odm_i,$|m_i|$ denote the number of requirements having a description of outcomes and a total number of requirements in respective order.

$com_{df}(m_i) = dfm_i{}^*[|m_i| + 1]^{-1}$// the notation $com_{df}(m_i)$ denotes the ratio of requirements of a module m_i having documenting formats; the notations dfm_i,$|m_i|$ denote the number of requirements having documenting formats and a total number of requirements in respective order.

$com(m_i) = 1 - \begin{bmatrix} com_d(m_i) * com_{ff}(m_i) * com_{nf}(m_i) * \\ com_{ip}(m_i) * com_{oc}(m_i) * com_{df}(m_i) \end{bmatrix}$ // assessing the completeness of the module m_i, which is the normalized value of the product of ratios of completeness measures.

End //of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.2. The correctness

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$ Begin // for each module m_i

$cor_{sp}(m_i) = spm_i{}^*[|m_i| + 1]^{-1}$// the notation $cor_{sp}(m_i)$ denotes the ratio of requirements listed in one or more sequences of requirement patterns; the notations spm_i,$|m_i|$ denote the number of requirements listed in one or more sequences of requirement patterns and a total number of requirements in respective order.

$cor_{ff}(m_i) = rfm_i{}^*[|m_i| + 1]^{-1}$// the notation $cor_{ff}(m_i)$ denotes the ratio of requirements seeking available and eligible functional factors of the module m_i; the notations rfm_i,$|m_i|$ denote the number of requirements seeking available and eligible functional factors and a total number of requirements in respective order.

$cor_{nf}(m_i) = rnm_i{}^*[|m_i| + 1]^{-1}$// the notation $cor_{nf}(m_i)$ denotes the ratio of requirements seeking available and eligible non-functional factors of the module m_i; the notations rnm_i,$|m_i|$ denote the number of requirements seeking available and eligible non-functional factors and a total number of requirements in respective order.

$cor_{si}(m_i) = sim_i{}^*[|m_i| + 1]^{-1}$// the notation $com_{si}(m_i)$ denotes the ratio of requirements of the module m_i having valid sources of all required input parameters; the notations sim_i,$|m_i|$ denote the number of requirements having valid sources of all required input parameters and a total number of requirements in respective order.

$cor_{voc}(m_i) = vom_i{}^*[|m_i| + 1]^{-1}$// the notation $cor_{voc}(m_i)$ denotes the ratio of requirements of the module m_i having the valid format of outcomes; the notations vom_i,$|m_i|$ denote the number of requirements having a valid format of outcomes and a total number of requirements in respective order.

$cor(m_i) = 1 - \begin{bmatrix} cor_{sp}(m_i) * cor_{ff}(m_i) * cor_{nf}(m_i) * \\ cor_{si}(m_i) * cor_{voc}(m_i) \end{bmatrix}$// assessing the correctness of the module m_i, which is the normalized value of the product of ratios of correctness measures.

End //of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.3. The unambiguity

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$Begin // for each module m_i

$ua_{uit}(m_i) = uim_i{}^*[|m_i| + 1]^{-1}$// the notation $ua_{uit}(m_i)$ denotes the ratio of requirements of the module m_i having unique interpretation; the notations uim_i,$|m_i|$ denote the number of requirements having unique interpretation and a total number of requirements in respective order.

$ua_{uif}(m_i) = ifm_i{}^*[|m_i| + 1]^{-1}$// the notation $ua_{uif}(m_i)$ denotes the ratio of requirements of the module m_i having unique input formats; the notations ifm_i,$|m_i|$ denote the number of requirements having unique input formats and a total number of requirements in respective order.

$ua_{ims}(m_i) = msm_i{}^*[|m_i| + 1]^{-1}$// the notation $ua_{ims}(m_i)$ denotes the ratio of requirements interpreted in different sequences of the module m_i; the notations msm_i,$|m_i|$ denote the number of requirements interpreted in multiple sequences and the total number of requirements in respective order.

$ua(m_i) = 1 - [ua_{uit}(m_i)^*ua_{uif}(m_i)^*ua_{ims}(m_i)]$// assessing the unambiguity of the module m_i, which is the normalized value of the product of ratios of unambiguity measures.

End // of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.4. The understandability

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$ begin // for each modulem_i

$us_d(m_i) = udm_i{}^*[|m_i| + 1]^{-1}$// the notation $us_d(m_i)$ denotes the ratio of requirements of the module m_i having understandable descriptions; the notations udm_i,$|m_i|$ denote the number of requirements having understandable descriptions and a total number of requirements in respective order.

$us_{pf}(m_i) = pfm_i{}^*[|m_i| + 1]^{-1}$// the notation $us_{pf}(m_i)$ denotes the ratio of requirements of the module m_i having understandable process flow; the notations pfm_i,$|m_i|$ denote the number of requirements having understandable process flow and a total number of requirements in respective order.

$us_{uf}(m_i) = ufm_i{}^*[|m_i| + 1]^{-1}$// the notation $us_{uf}(m_i)$ denotes the ratio of requirements of a module m_i having understandable functional factors; the notations ufm_i,$|m_i|$ denote the number of requirements having understandable functional factors and a total number of requirements in respective order.

(*continued on next page*)

(*continued*)

$us_{nf}(m_i) = unm_i{*}[|m_i| + 1]^{-1}$// the notation $us_{nf}(m_i)$ denotes the ratio of requirements of the module m_i having understandability of all required non-functional factors; the notations $ndm_i,|m_i|$ denote the number of requirements having understandable non-functional factors and a total number of requirements in respective order.

$us_{ip}(m_i) = uim_i{*}[|m_i| + 1]^{-1}$// the notation $us_{ip}(m_i)$ denotes the ratio of requirements of the module m_i having understandable input parameters; the notations $uim_i,|m_i|$ denote the number of requirements having understandable input parameters and a total number of requirements in respective order.

$us_{oc}(m_i) = uom_i{*}[|m_i| + 1]^{-1}$// the notation $us_{oc}(m_i)$ denotes the ratio of requirements of the module m_i having understandable outcomes; the notations $uom_i,|m_i|$ denote the number of requirements having understandable outcomes and a total number of requirements in respective order.

$us(m_i) = 1 - \begin{bmatrix} us_d(m_i) * us_{pf}(m_i) * us_{nf}(m_i) * \\ us_{ip}(m_i) * us_{oc}(m_i) \end{bmatrix}$// assessing the understandability of the module m_i, which is the normalized value of the product of ratios of understandability measures.

End //of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.5. The traceability

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$ Begin // for each module m_i

$tr_{rd}(m_i) = rdm_i{*}[|m_i| + 1]^{-1}$// the notation $tr_{rd}(m_i)$ denotes the ratio of requirements of module m_i referred in all necessary descriptions; the notations $rdm_i,|m_i|$ denote the number of requirements referred in all necessary descriptions and a total number of requirements in respective order.

$tr_{bd}(m_i) = bdm_i{*}[|m_i| + 1]^{-1}$// the notation $tr_{bd}(m_i)$ denotes the ratio of requirements of the module m_i referred in baseline documentation; the notations $bdm_i,|m_i|$ denote the number of requirements referred in baseline documentation and a total number of requirements in respective order.

$tr_{sld}(m_i) = slm_i{*}[|m_i| + 1]^{-1}$// the notation $tr_{sld}(m_i)$ denotes the ratio of requirements of the module m_i referred in system-level documentation; the notations $ndm_i,|m_i|$ denote the number of requirements referred in system-level documentation and a total number of requirements in respective order.

$tr_{sw}(m_i) = swm_i{*}[|m_i| + 1]^{-1}$// the notation $tr_{sw}(m_i)$ denotes the ratio of requirements of the module m_i referred in a statement of work; the notations $swm_i,|m_i|$ denote the number of requirements referred in the statement of work and the total number of requirements in respective order.

$tr(m_i) = 1 - [tr_{rd}(m_i){*}tr_{bd}(m_i){*}tr_{sld}(m_i){*}tr_{sw}(m_i)]$// assessing the completeness of the module m_i, which is the normalized value of the product of ratios of completeness measures.

End //of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.6. The modifiability

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$ begin // for each module m_i

$md_{ir}(m_i) = irm_i{*}[|m_i| + 1]^{-1}$// the notation $md_{ir}(m_i)$ denotes the ratio of requirements of the module m_i not dependent on other requirements; the notations $dm_i,|m_i|$ denote the number of requirements not dependent on other requirements and a total number of requirements in respective order.

$md_{nd}(m_i) = ndm_i{*}[|m_i| + 1]^{-1}$// the notation $com_{ff}(m_i)$ denotes the ratio of requirements of the module m_i not having dependent requirements; the notations $ndm_i,|m_i|$ denote the number of requirements not having dependent requirements and a total number of requirements in respective order.

$md_{sos}(m_i) = osm_i{*}[|m_i| + 1]^{-1}$// the notation $md_{sos}(m_i)$ denotes the ratio of requirements of the module m_i not dependent on specific operable sources; the notations $osm_i,|m_i|$ denote the number of requirements not dependent on specific operable sources and a total number of requirements in respective order.

$md(m_i) = 1 - [md_{ir}(m_i){*}md_{nd}(m_i){*}md_{sos}(m_i)]$// assessing the completeness of the module m_i, which is the normalized value of the product of ratios of modifiability measures.

End //of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.7. The verifiability

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$Begin // for each module m_i

$vr_{pt}(m_i) = ptm_i{*}[|m_i| + 1]^{-1}$//The notation $vr_{pt}(m_i)$ denotes the ratio of requirements of the module m_i having scope to perform testing; the notations $ptm_i,|m_i|$ denote the number of requirements having scope to perform testing and a total number of requirements in respective order.

$vr_{sp}(m_i) = spm_i{*}[|m_i| + 1]^{-1}$//The notation $vr_{sp}(m_i)$ denotes the ratio of requirements of the module m_i having sources to perform testing; the notations $spm_i,|m_i|$ denote the number of requirements having sources to perform testing and a total number of requirements in respective order.

$vr(m_i) = 1 - [vr_{pt}(m_i){*}vr_{sp}(m_i){*}ua(m_i){*}tr(m_i)]$// assessing the verifiability of the module m_i, which is the normalized value of the product of ratios of verifiability measures, unambiguity ratio, and traceability ratio of the corresponding module m_i. Unlike other measures, the metric verifiability depends on the other measures unambiguity and traceability (see Section 3.2.7).

End // of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

3.3.8. The consistency

$\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$ Begin // for each module m_i

$con_{sd}(m_i) = sdm_i{*}[|m_i| + 1]^{-1}$// the notation $con_{sd}(m_i)$ denotes the ratio of requirements of the module m_i not having conflicts between statements projected in requirement descriptions $sdm_i,|m_i|$ denotes the number of requirements not having conflicts between statements projected in requirement descriptions and a total number of requirements in respective order.

(*continued on next page*)

(*continued*)

$con_{bd}(m_i) = bdm_i{}^*[|m_i| + 1]^{-1}$// the notation $con_{bd}(m_i)$ denotes the ratio of requirements of the module m_i not reflecting any conflicts in baseline documentation; the notations $bdm_i, |m_i|$ denote the number of requirements not reflecting any conflicts in baseline documentation and a total number of requirements in respective order.

$con_{sr}(m_i) = srm_i{}^*[|m_i| + 1]^{-1}$// the notation $con_{sr}(m_i)$ denotes the ratio of requirements of the module m_i not conflicting with system requirements; the notations $srm_i, |m_i|$ denote the number of requirements not conflicting with system requirements and a total number of requirements in respective order.

$con_{ov}(m_i) = ovm_i{}^*[|m_i| + 1]^{-1}$//The notation $con_{ov}(m_i)$ denotes the ratio of requirements of the module m_i not conflicting with earlier versions of the SRS; the notations $ovm_i, |m_i|$ denote the number of requirements not conflicting with earlier versions of the SRS and a total number of requirements in respective order.

$con_{wp}(m_i) = wpm_i{}^*[|m_i| + 1]^{-1}$// the notation $con_{wp}(m_i)$ denotes the ratio of requirements of module m_i stated in white papers with no conflicts; the notations $dfm_i, |m_i|$ denote the number of requirements stated in white papers with no conflicts and a total number of requirements in respective order.

$con(m_i) = 1 - \begin{bmatrix} con_{sd}(m_i) * con_{bd}(m_i) * con_{sr}(m_i) * \\ con_{ov}(m_i) * con_{wp}(m_i) \end{bmatrix}$// assessing the consistency of the module m_i, which is the normalized value of the product of ratios of consistency measures.

End //of the $\forall_{i=1}^{|M|}\{m_i \exists m_1 \in M\}$

The document level (overall) quality of the SRS shall assess as follows.

Let the notations*comD*, *corD*, *uaD*, *usD*, *trD*, *mdD*, *vrD*, *conD* representing the quality measures completeness, correctness, unambiguity, understandability, traceability, modifiability, verifiability, and consistency, which initialized by 1.

$\forall_{i=1}^{|M|}\{m_i \exists m_i \in M\}$ Begin // for all modules of the document
$comD = comD{}^*com(m_i)$ // absolute product of the completeness scaled for all the modules
$corD = corD{}^*cor(m_i)$ // absolute product of the correctness scaled for all the modules
$uaD = uaD{}^*ua(m_i)$ // absolute product of the unambiguity scaled for all the modules
$usD = usD{}^*us(m_i)$ // absolute product of the understandability scaled for all the modules
$trD = trD{}^*tr(m_i)$ // absolute product of the traceability scaled for all the modules
$mdD = mdD{}^*md(m_i)$ // absolute product of the modifiability scaled for all the modules
$vrD = vrD{}^*vr(m_i)$ // absolute product of the verifiability scaled for all the modules
$conD = conD{}^*con(m_i)$ // absolute product of the consistency scaled for all the modules
End

Since the values are the product of fractions that results in lesser values, the quality measures shall be normalized. The measures $\{1 - comD, 1 - corD, 1 - uaD, 1 - usD, 1 - trD, 1 - mdD, 1 - vrD, 1 - conD\}$ are the final values of the quality metrics.

4. Experimental study

This section portrays the details of the dataset used, experimental study, and performance analysis of the proposed model by comparing it with contemporary models. The details follow

4.1. The data

The experiments have carried on the dataset nlreqdataset [29], which is reflecting the mixture of the legitimate description of the requirements specifications under IEEE metric standards, which labeled as positive, and the requirement specifications failed under most of the metrics, which labeled as negative, with the latter being reflecting the diversity in faded quality under IEEE metrics of the type stated in Table 1. Each record is a description of a requirement labeled as either positive or negative. Dataset is comprised of requirements, including the class feature attained under the status of requirement quality metrics. The records have been labeled as positive or negative. Among these, the positive records of count 12,186 and the records labeled as negative are 9139.

Table 1
Performance Diversity between SRSQQ and the other contemporary models (a) *T*-score and probability value (*p*-value) observed between SRSQQ and other models for precision and specificity.

Precision			Specificity		
	T-score	*p*-value		*T*-score	*p*-value
SRSQQ & ASRSD	7.01418	0.00003	SRSQQ & ASRSD	6.92955	0.00003
SRSQQ & SPSRS	10.3993	< 0.00001	SRSQQ & SPSRS	11.0692	< 0.00001
(b) *T*-score and probability value (*p*-value) observed between SRSQQ and other models for sensitivity and Accuracy					
Sensitivity			Accuracy		
	T-score	*p*-value		*T*-score	*p*-value
SRSQQ & ASRSD	6.87369	0.00004	SRSQQ & ASRSD	7.2938	0.00002
SRSQQ & SPSRS	4.41768	0.00084	SRSQQ & SPSRS	7.33886	0.00002
(c) *T*-score and probability value (*p*-value) observed between SRSQQ and other models for *F*-measure and MCC					
F-measure	Matthews correlation coefficient				
	T-score	*p*-value		*T*-score	*p*-value
SRSQQ & ASRSD	6.95995	0.00003	SRSQQ & ASRSD	7.22435	0.00002
SRSQQ & SPSRS	10.7768	< 0.00001	SRSQQ & SPSRS	7.37492	0.00002

4.2. The performance analysis

This section explains the measures used to assess the proposed method's performance against the other contemporary methods, and the comparative study about the statistics observed to corresponding metrics from the proposed and contemporary models ASRSD [27], SPSRS [28] has detailed.

4.2.1. Measures

The effective classification measures are designed from confusion-matrix, which offers outcomes of counting incorrectly & correctly identified instances for every events class. Hence, the range of determined measures statistically is deliberated and utilized based on comparative classifier analysis.

There were 4 basic metrics that were utilized in the confusion matrix for explaining measures of performance: TP (true positives) that indicates the number of correct predictions of positive on the set of testing; TN (true negatives) that signifies the count of correct predictions in negative on the set of training; FP (false positives) depicts the amount of incorrect or wrongly positive estimations on the set of testing; FN (false negatives) depicting the amount of incorrect or wrongly negative predictions on a set of testing.

The metrics determined above were utilized for various performance measures, which are implemented for determining selected classifier quality. The most prominent measures were: rate of detection, specificity, the value of ROC, sensitivity & precision. We need to assess the rate of detection & ROC (Receiver operating characteristics) for a pre-requisite of this contribution. Moreover, the detection rate, generally called AC (Accuracy) defines

4.3. Comparative study of the metric values

Fig. 2 shows the graph is plotted between metric precision and ten folds over the proposed SRSQQ model and contemporary models ASRSD and SPSRS. The metric precision is also called a positive predictive value, which signifies the ratio of related instances among retrieved instances. From statistics, it is envisaged that the precision for the proposed SRSQQ model is more significant than contemporary models.

Among several metrics, specificity is one among them, which is defined as the ratio of true negatives to the total number of actual negatives considered. The graph is plotted between this specificity metric and ten folds for the proposed SRSQQ model of this contribution and contemporary models ASRSD, and SPSRS as shown in Fig. 3, which denotes that the proposed model performs better when compared to contemporary models.

Fig. 4 shows the graph is plotted between metric sensitivity and ten folds over the proposed SRSQQ model and contemporary models ASRSD and SPSRS. The metric sensitivity is also called recall, which signifies the ratio of true positives to the actual positives considered. From statistics, it is envisaged that the sensitivity for the proposed SRSQQ model is more significant than contemporary models.

Accuracy is one among several metrics defined as a ratio of the sum of true positives and true negatives to the sum of actual positives and negatives considered. The graph is plotted between this accuracy metric and ten folds for the proposed SRSQQ model of this contribution and contemporary models ASRSD and SPSRS, as shown in Fig. 5. It is envisaged from the statistics that the accuracy for the proposed model performs better when compared to contemporary models.

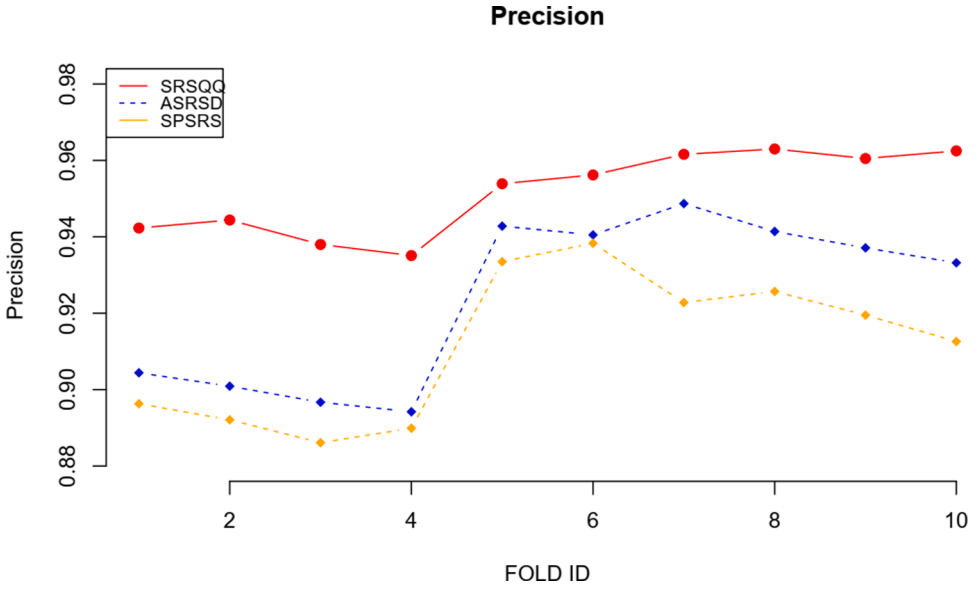

Fig. 2. Precision observed for proposed model SRSQQ, and contemporary models ASRSD, and SPSRS.

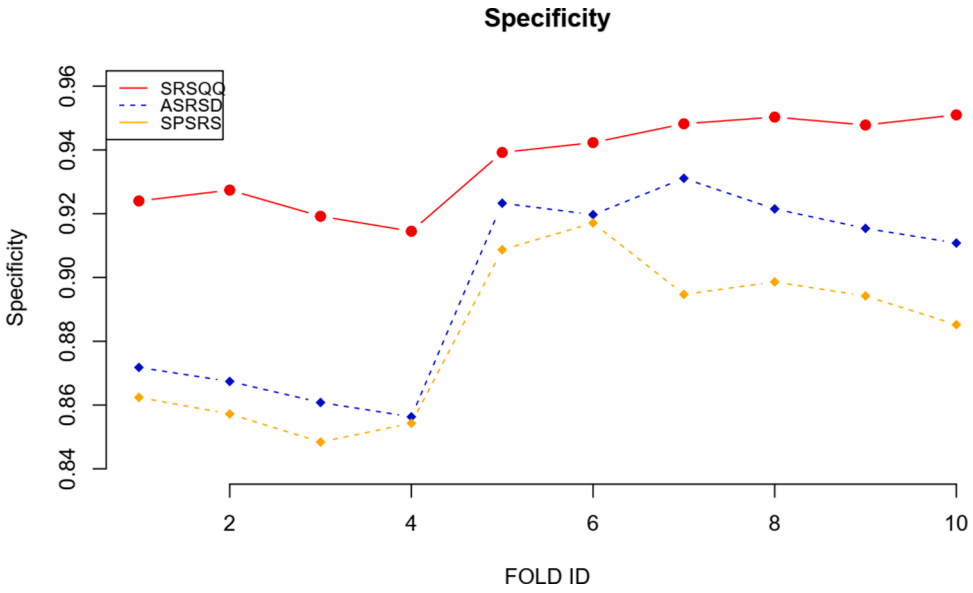

Fig. 3. Specificity observed for proposed model SRSQQ, and contemporary models ASRSD, and SPSRS.

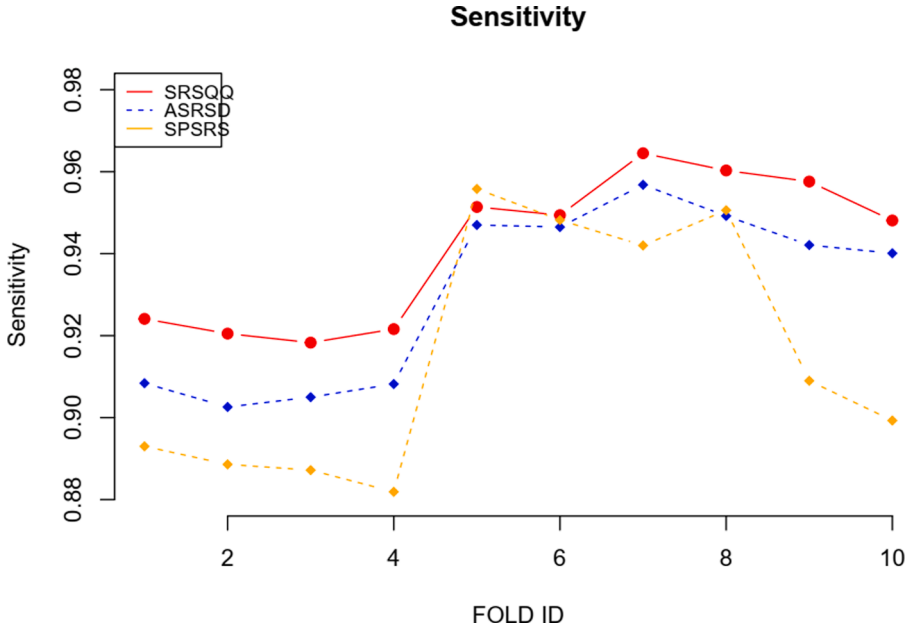

Fig. 4. Sensitivity observed for proposed model SRSQQ, and contemporary models ASRSD, and SPSRS.

Fig. 6 shows the graph is plotted between metric F-measure and ten folds over the proposed SRSQQ model and contemporary models SPSRS and ASRSD. From statistics, it is envisaged that the F-measure for the proposed SRSQQ model is more significant than SPSRS, ASRSD contemporary models.

The metric MCC is one among several metrics, which have been used in machine learning for evaluating binary classification. The graph is plotted between this MCC metric and ten folds for the proposed SRSQQ model of this contribution and contemporary models ASRSD and SPSRS, as shown in Fig. 7. It is envisaged from the statistics that the MCC for the proposed model is better when compared to contemporary models.

The uniformity in performance of the projected method envisioned by *T*-test [30] implemented on the values of performance measures attained from SRSQQ, and contemporary methods ASRSD and SPSRS. Table 1 discussed t-score & corresponding probability degree perceived for distinct metric values between SRSQQ and other contemporary models. The probability degree of correlation observed for the *T*-test between SRSQQ and the other different ways is almost 0 with a positive t-score. Therefore, it could be obvious to

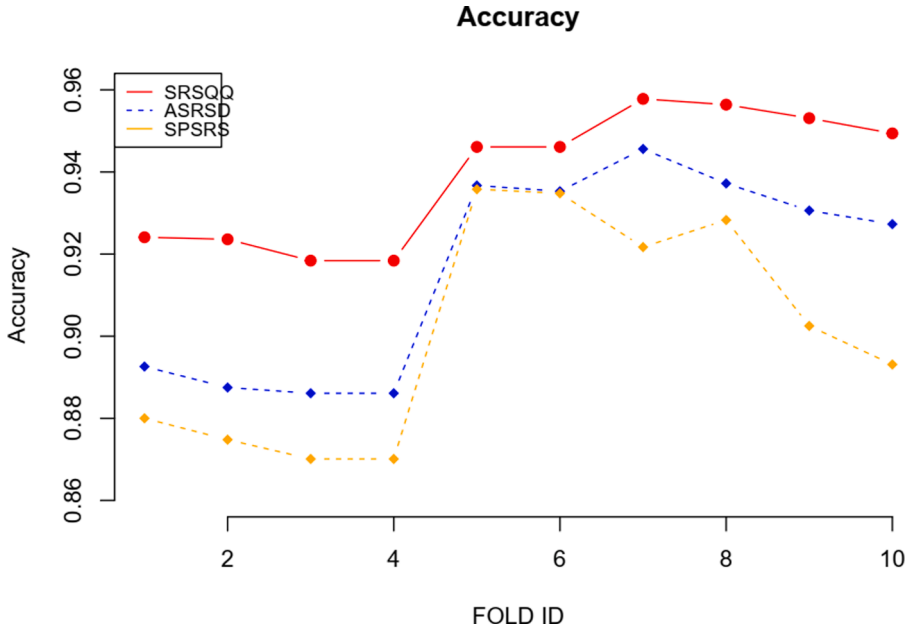

Fig. 5. Accuracy observed for proposed model SRSQQ, and contemporary models ASRSD, and SPSRS.

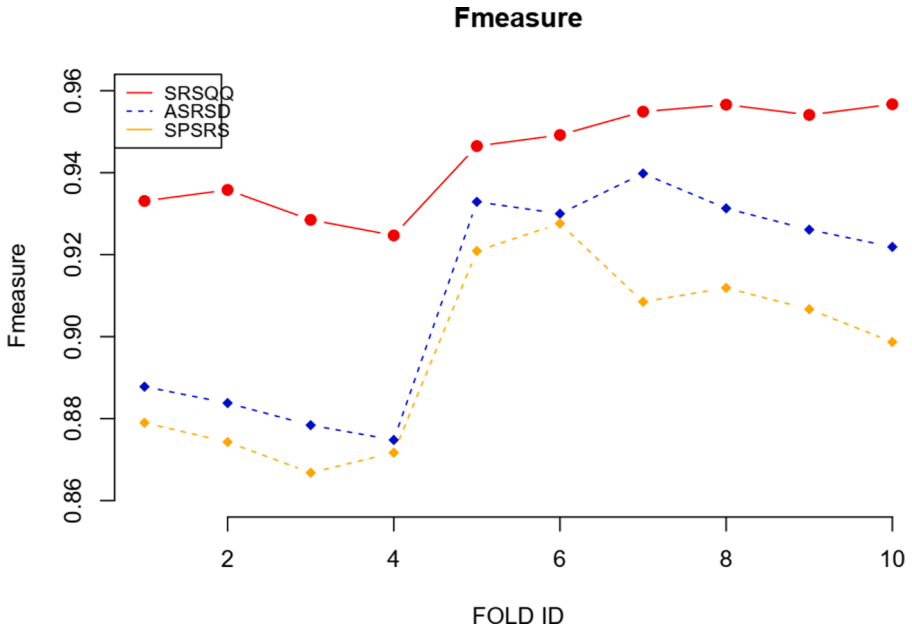

Fig. 6. *F*-measure observed for proposed model SRSQQ, and contemporary models ASRSD, and SPSRS.

finalize that; the projected SRSQQ method significantly performs better than all other existing methods deliberated for the comparison; however, the ASRSD [27] and SPSRS [28] are competent in respective order that compared to the other contemporary methods.

5. Conclusion

Software requirement specification collection is the critical phase among the multiple phases of the Software development life-cycle. The qualitative factors of the software development lifecycle highly correlate to the quality and optimality of the software requirement specification. The software requirement specification quality determines under diversified factors such as completeness, correctness, unambiguity, consistency, modifiability, interpretation, traceability, understandability, and verifiability, often stated as IEEE metrics for SRS quality assessment. However, these IEEE metrics' assessment methodology is incompetent to assess the quality of

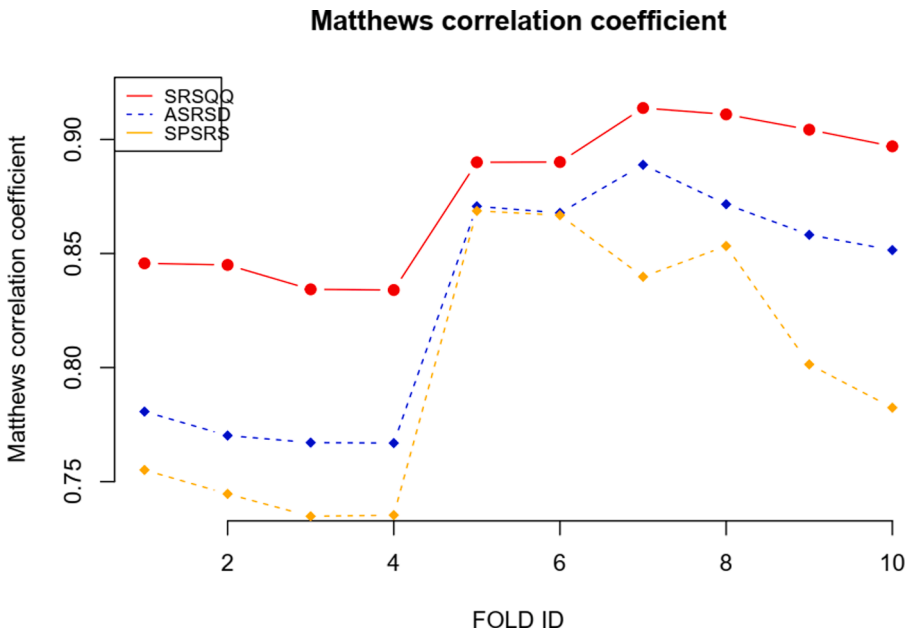

Fig. 7. Matthews's correlation coefficient (MCC) observed for proposed model SRSQQ and contemporary models ASRSD, and SPSRS.

the software requirement specification of modern software requirements. In this regard, this manuscript portrayed novel assessment methodologies to determine the SRS quality assessment factors. The experimental study stated the significance and robustness of the proposed version of SRS quality assessment metrics. Future research can use these assessment strategies to be considered a fitness function in machine learning and soft computing-based SRS quality assessment strategies.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

CRediT authorship contribution statement

M.R. Raja Ramesh: Conceptualization, Funding acquisition, Formal analysis, Data curation, Writing – original draft, Writing – review & editing. **Ch. Satyananda Reddy:** Conceptualization, Funding acquisition, Formal analysis, Data curation, Writing – original draft, Writing – review & editing.

Declaration of Competing Interest

This paper has not communicated anywhere till this moment, now only it is communicated to your esteemed journal for the publication with the knowledge of all co-authors.

References

[1] Konig F, Ballejos LC, Ale MA. A semi-automatic verification tool for software requirements specification documents. In: Proceedings of the Simposio Argentino de Ingeniería de software (ASSE)-JAIIO 46. Córdoba; 2017. p. 75–83. 2017).
[2] Sun S, Luo C, Chen J. A review of natural language processing techniques for opinion mining systems. Inf Fusion 2017;36:10–25.
[3] Alpaydin E. Machine learning: the new AI. MIT press; 2016.
[4] Barocas S, Hardt M, Narayanan A. Fairness in machine learning. NIPS Tutor 2017;1.
[5] Mund JM. Measurement-based quality assessment of requirements specifications for software-intensive systems. Diss. Technische Universität München; 2017.
[6] Nazir F, et al. The applications of natural language processing (NLP) for software requirement engineering-a systematic literature review. In: Proceedings of the international conference on information science and applications. Springer; 2017. p. 485–93.
[7] Mylopoulos J, Chung L, Nixon B. Representing and using non-functional requirements: a process-oriented approach. IEEE Trans Softw Eng 1992;18(6):483–97.
[8] Glinz M. On non-functional requirements. In: Proceedings of the 15th IEEE international requirements engineering conference (RE). IEEE; 2007. p. 21–6. 2007).
[9] Chung L, Sampaio do Prado Leite JC. On non-functional requirements in software engineering. Conceptual modeling: foundations and applications. Springer; 2009. p. 363–79.
[10] Blaine JD, Cleland-Huang J. Software quality requirements: how to balance competing priorities. IEEE Softw 2008;25(2):22–4.
[11] Regnell B, Berntsson Svensson R, Olsson T. Supporting road mapping of quality requirements. IEEE Softw 2008;25(2):42–7.
[12] SSvensson RB, Gorschek T, Regnell B. Quality requirements in practice: an interview study in requirements engineering for embedded systems. In: Proceedings of the international working conference on requirements engineering: foundation for software quality. Springer; 2009. p. 218–32.
[13] Glinz M. A risk-based, value-oriented approach to quality requirements. IEEE Softw 2008;25(2):34–41.

[14] Knauss E, Boustani CE. Assessing the quality of software requirements specifications. 2008. In: Proceedings of the 16th IEEE international requirements engineering conference. IEEE; 2008. p. 341–2.
[15] Wei B, et al. Automated reasoning with goal tree models for software quality requirements. 2012. In: Proceedings of the IEEE 36th annual computer software and applications conference workshops. IEEE; 2012. p. 373–8.
[16] Svensson RB, Olsson T, Regnell B. An investigation of how quality requirements are specified in industrial practice. Inf Softw Technol 2013;55(7):1224–36.
[17] Kaiya H, et al. Toward quality requirements analysis based on domain specific quality spectrum. In: Proceedings of the ACM symposium on applied computing; 2008. p. 596–601.
[18] Kaiya H, Ohnishi A. Improving software quality requirements specifications using spectrum analysis. 2012. In: Proceedings of the IEEE 36th annual computer software and applications conference workshops. IEEE; 2012. p. 379–84.
[19] Cleland-Huang J, et al. The detection and classification of non-functional requirements with application to early aspects. In: Proceedings of the 14th IEEE international requirements engineering conference (RE'06). IEEE; 2006. p. 39–48.
[20] Rahimi M, Mirakhorli M, Cleland-Huang J. Automated extraction and visualization of quality concerns from requirements specifications. 2014. In: Proceedings of the IEEE 22nd international requirements engineering conference (RE). IEEE; 2014. p. 253–62.
[21] Slankas J, Williams L. Automated extraction of non-functional requirements in available documentation. 2013. In: Proceedings of the 1st international workshop on natural language analysis in software engineering (NaturaLiSE). IEEE; 2013. p. 9–16.
[22] Casamayor A, Godoy D, Campo M. Identification of non-functional requirements in textual specifications: a semi-supervised learning approach. Inf Softw Technol 2010;52(4):436–45.
[23] Chantree F, et al. Identifying nocuous ambiguities in natural language requirements. In: Proceedings of the 14th IEEE international requirements engineering conference (RE'06). IEEE; 2006. p. 59–68.
[24] Fantechi A, et al. Applications of linguistic techniques for use case analysis. Requir Eng 2003;8(3):161–70.
[25] Takuya W, Masuhara H. A spontaneous code recommendation tool based on associative search. In: Proceedings of the 3rd international workshop on search-driven development: users, infrastructure, tools, and evaluation; 2011. p. 17–20.
[26] Cleland-Huang J, et al. A machine learning approach for tracing regulatory codes to product specific requirements. In: Proceedings of the 32nd ACM/IEEE international conference on software engineering-volume 1; 2010. p. 155–64.
[27] Osman MH, Zaharin MF. Ambiguous software requirement specification detection: an automated approach. 2018. In: Proceedings of the IEEE/ACM 5th international workshop on requirements engineering and testing (RET). IEEE; 2018. p. 33–40.
[28] del Sagrado J, del Aguila IM. Stability prediction of the software requirements specification. Softw Qual J 2018;26(2):585–605.
[29] http://fmt.isti.cnr.it/nlreqdataset/.
[30] Budak H, Taşabat SE. A modified *T*-score for feature selection. Anadolu Universities Bilim Ve Teknoloji Dergisi A Uygulamalı Bilimlerve Mühendislik 2016;17 (5):845–52.