Enhancing the Quality of Software Requirements Artifacts with Scoring Rubrics-Assisted Reading Technique

Emmanuel O.C. Mkpojiogu, Nor LailyHashim, Azham Hussain

Abstract: ***The quality level of software requirements artifacts depends on how adequate esuch artifacts are read. Several reading techniques exist and the most frequently used is the checklist reading technique. Though checklist is an improvement over the ad hoc approach, it is however, laden with some setbacks that impacts on its usability and efficacy. It presents before readers questions that lack detailed guides as to how to carry out the requirements artifacts reading process. In this study, scoring rubrics-assisted reading (SRAR), which over the years has been used in Universiti Utara Malaysia's School of Computing (Software Engineering sub-department) in the assessment of students' requirements' artifacts, was empirically evaluated. SRAR is a checklist-like reading approach that can be used for the evaluation of the quality of software requirements artifacts. The main objective of the study is to evaluate SRAR. Scoring rubrics were employed in the reading of requirements artifacts. The review was done by four independent experts in two iterations. The difference in the mean rubric scores between roundsof reading was used as the performance metrics to measure the improvement in the quality of the artifacts. Initial results indicate that SRAR has efficacy as there was decrease in defects after refinement using SRAR. The paper will be useful to requirements engineers and the research community as the evaluated technique is conceptually a superior alternative to the checklist and ad hoc reading techniques.***

Keywords: ***quality improvement, software evaluation, work products***

I. INTRODUCTION

Software reading is an integral part of software quality assurance process and its main objective is the detection of defects (Oladele &Adedayo, 2014) and the achievement of software quality (Mohammed et al., 2015). It is the textual analysis of software artifacts, designs and models with the sole objective of detecting and finding out defects or errors that are underlying therein (Basili et al., 1996). It is the series of steps aimed at guiding inspectors in acquiring a deep understanding of the inspected software product(s) (Oladele &Adedayo, 2014; Mohammed et al., 2015).

Revised Manuscript Received on May 22, 2019.

Emmanuel O.C Mkpojiogu, Department of Computer and Information Technology, Veritas University, Abuja, Nigeria. School of Computing, Universiti Utara Malaysia, 06010 Sintok, Malaysia

Nor LailyHashim, School of Computing, Universiti Utara Malaysia, 06010 Sintok, Malaysia

Azham Hussain, School of Computing, Universiti Utara Malaysia, 06010 Sintok, Malaysia

Reading involves evaluating the form, structure, content and documentation of software artifacts. The individual analysis of textual software work productsand artifacts is a major activity in several software engineering tasks such as verification and validation, maintenance, evolution and reuse,among others. The aim and reason for software reading is to accomplish the enhancement of the quality of the given software artifacts and in due course, the entire software. Software reading is a way of developing high quality software. It provides assurance on the quality of a software work product (Berling&Runeson, 2003). The quality of work products such as the software requirements specification, use case documents, test plans, test cases, etc, is the key to the success of software development.

Software reading is a primary technical activity in the software quality assurance process and is needed throughout the software development lifecycle. Software artifacts (such as list of requirements, requirement specification, design, code, test plan) associatedwith the various phases of software development, are often subjected to iterative reading, regular reviews, constant modifications and continual understanding. In reading work documents and artifacts, the artifacts are readand analyzed to assess the various qualities and characteristics so as to enhance software product quality (Basili et al., 1996). Beginning the software quality assurance process early and successfully detecting defects in requirements work documents at the early stages of the software development lifecycle assists in reducing the probability of defects slipping into the later phases of the development lifecycle. Defects in the later phases of software development are more expensive to correct (Basili et al., 1996). This is in the areas of rework, effort, finance, manpower and end product delivery delays, among others. Therefore, it is more beneficial to start the software quality assurance efforts earlyin the software development process. Inspections are used to find defects early in the development process, and have shown to be cost effective (Wohlin et al., 2012).

Software quality assurance of work documents and artifacts is typically in the form of formal design review, inspection or walkthrough with varying levels of formality, structure, and team participation. According to Summerville (2007), software inspection is a commonly used method in software verification.

It can be carried out in different ways and used throughout the software development process for the following reasons: 1) understanding, 2) finding defects, and 3) as a basis for making decisions (Wohlin et al., 2012).The pivotal part of the reading process is the detection of defects carried out by individual reviewers reading the documents and recording the defects (Wohlin et al., 2012).In recent times, there are empirical evidences that defect detection is more of an individual activity than a group activity. Inspection results are completely determined by the inspectors (Oladele &Adedayo, 2014; Mohammed et al., 2015).

Reading techniques help in the reviewing, up-grading and enhancement of the quality of software work documents and artifacts. These techniques among others are as follows: ad hoc reading, checklist-based reading, scenario-based reading, defect-based reading, and perspective-based reading. Defect-based reading and perspective-based reading are both types of scenario-based reading (Basili et al., 1996; Shull et al., 2000; Halling et al., 2001; Kollanus& Koskinen, 2007; Wohlin et al., 2012; Alshazly et al., 2014). Reading techniques range from intuitive, non-systematic procedures such as ad hoc or checklist-based techniques, to explicit or highly systematic procedures such as formal proof of correctness (Oladele &Adedayo, 2014). However, the quality enhancement capacities of the various techniques are at varying levels of efficacy.

The most frequently and commonly usedreading technique is the checklist reading technique (Basili et al., 1996; Shull et al., 2000). Even though this technique is an enhancement over the ad hoc reading technique, it is nonetheless, encumbered with some drawbacks that impacts its usability, effectiveness, efficiency, and efficacy (Shull et al., 2000; Basili et al., 1996). Ad hoc and checklist-based reading techniques remain the dominant reading techniques in the industry. In fact, checklist-based reading is considered the standard in software organizations. In ad hoc reading, technical support is normally provided to readers. Reviewers rely on their own skills, knowledge, and experience to identify defects and may be taken through some session of training before the reading exercise (Oladele &Adedayo, 2014).

In checklist-based reading, the artifact is examined with the aid of a pre-defined list of questions (i.e., checklists) (Alshazly et al., 2014). The technique is more structured and is believed to offer more support to reviewers when compared to the ad hoc approach (Oladele &Adedayo, 2014). However, this checklist-based reading approach containsquestions that do not have precise guidelinesand directions on how readers can go aboutcarrying out the reading activity and process. This present study evaluates scoring rubric-assisted reading (SRAR), a checklist-like reading technique for the quality assessment of software requirements documents and artifacts. The objective of the study is to evaluate the quality enhancement capacity and efficacy of SRAR. The structure of the paper is as follows: Section 2 presents the background; section 3 presents the scoring rubrics-assisted reading approach; section 4 is the methods; section 5 is the results and discussion section; and lastly, section 6 concludes with conclusion and future work.

II. BACKGROUND

Software reading is an old idea whose main focus is the detection of defects in software artifacts. Prior research identifies several reading techniques. These techniques include: ad hoc reading, checklist-based reading, scenario-based reading, defect-based reading, and perspective-based reading. Defect-based reading and perspective-based reading are both types of scenario-based reading (Basili et al., 1996; Shull et al., 2000; Halling et al., 2001; Kollanus& Koskinen, 2007; Wohlin et al., 2012). Several techniques exist for individually reading requirements/software documents. The techniques are also used in or by review, inspection and walkthrough teams. At one extreme is the ad hoc reading technique. This technique has no formal or systematic procedure and is based solely on the reviewers' experience and intuition (Shull et al., 2000). It is an unstructured technique which provides no guidance, implying that reviewers detect defects based on their personal knowledge and experience (Wohlin et al., 2012). The checklist reading technique offers a little improvement over the ad hoc approach. It makes the inspection process slightly better defined through the provision of a list of items on which to focus on, to the readers. It captures the knowledge of past inspections, and helps reviewers to focus on their reading (Wohlin et al., 2012).

Defect-based reading technique, a scenario-based reading technique, provides a set of systematic steps and procedures which readers can follow as guides. In scenario-based technique, different reviewers have different responsibilities and are guided in their reading by specific scenarios which aim at constructing a model, instead of just passive reading (Wohlin et al., 2012).Defect-based and perspective reading techniques are variants of scenario-based reading. While the former concentrates on specific defect classes, the latter focuses on the points of view of the user of the document (Wohlin et al., 2012). Perspective-based reading provides a step by step guidance that is tailored and suited to requirements expressed in a natural language like English. Software readers' involvement is based on some perspectives which they focus on. Perspective-based reading requires the identification of users of specific software artifact (such as requirements) (Shull et al., 2000).

The ad hoc reading approach allows readers to use their knowledge and judgment in the detection of defect. It does not provide any guide or assistance to the reader(s). In this approach, there are no well defined procedures and reviewers learn largely by practice. Readers gain expertise gradually with continuous repeated practice (Shull et al., 2000). Users of ad hoc reading solely depend on their intuition and past experience.

This approach is prune to faults, and it is not very effective and efficient (Mkpojiogu & Hashim, 2017a) (see Mkpojiogu & Hashim, 2017b; 2017c and Mkpojiogu& Hussain, 2017a; 2017b). More so, the difficulty in offering training for a poorly defined or undefined process like the ad hoc reading process compounds the problem. Therefore, the process cannot be improved upon and it is not repeatable (Shull et al., 2000) (see Table 2).

In checklist-based reading, checklists are used. Checklists comprises of a set of general questions concerning types of defects or the possible signs (pointing to defects) to checkout for or look out for in a given type of document or artifact. Checklist assists readers in reading, remembering and recollecting the aspects that are to be checked and reviewed. Nevertheless, it provides little guidance on what particularly the reviewer is to do. The reader has to map checklist questions to tasks and plan how to traverse the document to be reviewed. Checklist does not allow for the reading process to be repeatable and it is also open to faults and changes/variations (Halling et al., 2001) (see Table 2). One checklist in most cases is used by all the readers in a team, but does not enable a focused coordination of the work of the various members of the team. This may lead to wastage of efforts by team members. Checklist can cover a wide range of areas/issues, but it still requires readers reading through the document in sequence several times. Therefore, this tends tolimits the applicability of checklists to only documents and artifacts with limited size. The employment of checklists may make the reviewers to eventuallygo back to ad hoc reading by using their personal intuition and experience (Halling et al., 2001).

Scenario-based reading uses procedures (step-wise processes) to find and detect specific classes of defects. This technique guides and directs software readers through a given work product or artifact with a particular point of view or emphasis. It provides scenarios that offers step by step guideline and stimulate readers to work actively with the document through note taking, annotating of the document and building a mental framework or model that can lead to more consistency in the given view of a reader (Halling et al., 2001). Scenarios provides guidance and indications on different levels of details, starting from organizing entities, and instructing readers on how to recognize them and to abstract information that are relevant, and include them to the analysis. These steps are repeated on several levels of detail (Halling et al., 2001). This technique has the following merits: the process can be repeated (i.e., it is reliable), audited and canoffer the planners of inspection or review, a means to assign readers to where to focus, in a prescribed form for particular defects (Halling et al., 2001). The earlier mentioned descriptions on ad hoc and checklist reading techniques show their deficiencies and inadequacies (see Table 2). In the light of this, this paper seeks to evaluate a checklist-type reading technique (i.e., SRAR) that overcomes the drawbacks and demerits of the checklist and ad hoc reading techniques.

III. SCORING RUBRICS-ASSISTED READING

This paper evaluates the SRAR technique that assists readers in requirements work documents/artifact reading, defect detection, defect removal and in the enhancement of the quality of work documents and artifacts. Checklists are simplified rubrics. However, rubrics are better off than checklists because they provide detailed clues based on the attributes of the given work document and artifact along all dimensions of the work document, rather than just providing general and broad based questions that do not offer to readers any precise and detailed guidance and direction,as it is in the case of checklists. Scoring rubrics also scores the quality of the attributes of the artifact as well as the entire work document. Hence, it provides a platform to measure and monitor progress in the quality enhancement of a specified work product or artifact. SRARis reliable, valid and consistent. Its assessment process can be repeated with a good degree of consistency unlike the checklist or ad hoc approach (see Table 2).

Scoring rubrics are two dimensional, with rows and columns. The rows are the attributes of a given work document or artifact while the column is a Likert-type rating scale that clearly describes the various expected quality level of the row-wise work documentor artifact attributes. The qualities graduate and grow in ordinal form from no quality to the most expected quality. Scoring rubrics are traditionally used in the field of education to assess the performance of students' work and projects and how the students are improving in repeated assessments. It is used for assessment and evaluation. It offers a uniform and consistent evaluation platform (Jonsson &Svingby, 2007; Dietrich, 2008; Sherman & Martin, 2015; McKenzie & Wood-Bradley, 2014; Dessai et al., 2014).

Scoring rubric adopted and employed in this study to assess the quality of requirements work products and artifacts weretwo-dimensional and Likert-like in design. The columns represent a 4-point Likert-type rating scale (for example, 1. Not acceptable, 2. Below expectations, 3. Meet expectations, 4. Exceeds expectations); a fifth column is added for "not applicable" (see Table 3 and Table 4). The rows consist of the attributes of the given work products. In some cases, the attributes are further defined into criteria. In addition, the cells were created as the intersection of the rows and columns to represent a clear description of the work products' attributes with respect to the corresponding rating scale. Each attribute is scored and the scores of all attributes are totaled to form a total score for the given work product that represents its relative quality. The scores of the rubrics are used as performance metrics to assess the quality of the requirements documents. Higher rubric scores show higher quality (i.e., lower defects) and lower rubric scores indicate lower quality of software requirements documents (implying, higher defects).

The steps involved in the SRAR technique include the following: 1) preparation or development of a given artifact; 2) development of a customized rubric for the specified requirements/software artifact; 3) validation of the rubric; 4) assignment of reading responsibilities to reviewer(s); 5) step-wise reading of the artifact, tenaciously and closely following the information on the rubric corresponding to the artifact attribute's level of quality (here, defects are detected); 6) present the reviewed artifact together with the rubric results to the analyst(s) for refinement 7) step-wise refinement and removal of defects (here, detected defects are located and removed); 8) after refinement, the refined artifact is re-presented to the reviewer(s) for further step-wise reading (step 5 is repeated); 9) the re-read artifact is again presented to the analyst for further refinement (steps 6 & 7are repeated); 10) the iteration continues, repeating steps 5 to 7 cyclically until the desired quality expected/ desired from the artifact is achieved or almost attained.

Generally speaking, engineering designs, system designs, artifacts, codes, and work products can be assessed for quality using SRAR. In software engineering particularly, software reading can be facilitated using SRAR as the technique detects and assists in the removal of defects. In software reading, the focus is on improving the quality of work products/artifacts and not necessarily on the reader because the purpose of software reading is to read software documents and detect defects in them leading to the eventual removal of such errors/defects and the subsequent improvement of the quality of the given work document or artifact (Mkpojiogu & Hashim, 2017). The technical competence of reviewers is nonetheless mandatory. This present study does not compare methods but as a preliminary study, it shows how requirements work documents' quality can be enhanced using SRAR. Table 1 shows how the SRAR technique compared with other software reading techniques.

Table. 1 A comparison of reading techniques

Reading Type	Systematic?	Focused?	Controlled?	Customizable?	Training?
Ad hoc reading	No	No	No	No	No
Checklist reading	Partly	No	Partly	Partly	Partly
Defect-based reading	Yes (Complex)	Yes	Yes	Yes	Yes (Detailed)
Perspective-based reading	Yes (Complex)	Yes	Yes	Yes	Yes (Detailed)
Scoring rubrics assisted reading	Yes (Simpler)	Yes	Yes	Yes	Yes (Simpler)

Table. 2 Comparison of Ad hoc, and Checklist Reading Techniques with SRAR

Reading Type	Tool Used	Means of Reading	Defined Procedure?	Repeatable Process?	Details Provided?	Fault Prone?
Ad hoc reading	No tool	Intuition /Past experience	No	No	No	Yes
Checklist reading	Checklist	List of questions (with no details)	No	No	No	Yes
Scoring rubrics-assisted reading	Scoring Rubric (improved checklist)	Artifacts' attributes with graduated quality levels	Yes	Yes	Yes	No

As can be seen in Table 1, SRAR has the following features: i) systematic: the items in the rubrics follow a systematic ordering of the expected attributes in the work product. It is systematic and yet simpler when compared to the two scenario-based reading techniques; ii) focused: like the two scenario-based reading techniques, SRAR is focused in the sense that it focuses on the expected attributes of the work product, of which a deviation from, implies a defect. It has an in-built standard that guides. It focuses on the expected structure, form, standard and documentation of the particular document; iii) controlled: SRAR is also controlled in the sense that its attributes and dimensions are properly defined in the rubrics; iv) customizable: SRAR is customizable and can be tailored to a particular software/requirements document or artifact to be reviewed. It can also be tailored to any organization and situation; v) training: SRAR require simple and minimal training for users as it is a simple technique, however, defect-based and perspective-based reading approaches are more complex having more detailed training. In all these characteristics, SRAR towers above both ad hoc and checklist reading techniques and thus, is conceptually a better reading and defect finding alternative to them.

IV. METHODS

In this paper, scoring rubrics were used to assess the quality of requirements work products and artifacts. There was a scoring rubric specifically designed for each artifact. As mentioned before, each scoring rubric is two-dimensional and Likert-like in design.

The columns represent a 4-point Likert-type rating scale (for example, 1. Not acceptable, 2. Below expectations, 3. Meet expectations, 4. Exceeds expectations); a fifth column is added for "not applicable" (see Table 3 and Table 4). The rows consist of the attributes of the given work products. In some cases, the attributes are further defined into criteria. In addition, the cells created by the intersection of the rows and columns represent a clear description of the work products' attributes with respect to the corresponding rating scale. Each attribute is scored and the scores of all attributes are totaled to form a total score for the given work product that represents its relative quality. The scores of the rubrics are used as metrics to assess the quality of the requirements documents. Higher rubric scores show higher quality (i.e., lower defects) and lower rubric scores indicate lower quality of software requirements documents (implying, higher defects).

Table. 3 Scoring Rubric Framework

Attributes	Criteria	Scale of Score					Score
	(optional)	1.(Not acceptable)	2.(Below expectations)	3.(Meets expectations)	4. (Exceeds expectations)	N/A (Not available)	
$Attribute_1$	…	$Cell_{11}$	$Cell_{12}$	$Cell_{13}$	$Cell_{14}$	…	…
…	…	$Cell_{21}$	$Cell_{22}$	$Cell_{23}$	$Cell_{24}$	…	…
…	…	$Cell_{31}$	$Cell_{32}$	$Cell_{33}$	$Cell_{34}$	…	…
Attribute	…	$Cell_{n1}$	$Cell_{n2}$	$Cell_{n3}$	$Cell_{n4}$	…	…
Total Score							(%)

Table. 4 An Example of a Simple Scoring Rubric for Activity Diagram Report

Attributes	Scale of Score					Score
	1. (Not acceptable)	2.(Below expectations)	3.(Meets expectations)	4. (Exceeds expectations)	N/A (Not available)	
Appropriate symbol of representation	Incorrect notation	Some notations incorrect or misused	Correct notations	So clear and complete	…	…
Flow of process presented	Unclear or poorly designed	Incomplete or not well designed	Clear and complete	So simple and clear. The design is understandable to all intended readers	…	…
Total Score						(%)

The research question that guided the study is as follows: Does SRAR technique improves the quality of requirements work documents and artifacts? As this study is an exploratory one, descriptive statistics was used in answering this research question. The scores of the scoring rubric were used as performance metrics in the study. The study's data (rubric scores) were presented as averages and percentages. These rubrics scores can also aid in comparing the quality of two or more software or requirements artifacts. In essence, the quality of two or more artifacts can be compared using the scores of scoring rubrics.

This study was conducted at the School of Computing, Universiti Utara Malaysia (UUM). The Software Engineering sub-department of the School has over time been using SRAR in assessing the quality of students' software requirements work products, artifacts and models. However, the approach has not been empirically validated. This study is part of an attempt to offer an empirical assessment of the efficacy and quality improving capability of the technique. The study was part of the wider study carried out in developing an e-health awareness system (a student's project) (Hussain &Mkpojiogu, 2016; Hussain, Mkpojiogu&Nawi, 2016).

The artifacts (requirements work products/documents and models) used in the study were prepared by one of the researchers. There were 11 artifacts prepared on the whole. They include: 1) vision and scope document; 2) list of requirements; 3) use case description document; 4) software requirements specification (SRS) document; 5) test plan; 6) test cases; 7) use case diagram; 8) activity diagram; 9) sequence diagram; 10) collaboration diagram; and 11) class diagram. After these artifacts were produced, they were reviewed in two iterations (rounds) by four (4) reviewers (experts) who checked out for and detected defects in the work products and scored the artifacts appropriately for quality. Each expert read and scored the requirements artifacts and models independently and separately.

After each round of reading, the requirements work documents were refined and the defects removed. The refining exercise was carried out by one of the researchers who located the detected defects captured by the readers and removed them as much as possible before re-presenting the artifacts to the readers for any further reading and defect detection. As was earlier said, the reading of the requirements artifacts was done by four (4) reviewers.

They independently evaluated the artifacts. The evaluation was paper-based. These four evaluators were of the rank of Senior Lecturer in the School of Computing, Universiti Utara Malaysia. They all have PhD qualification and are experts in software engineering with several years of teaching and research experience. These experts were purposively selected and recruited for the study for the purpose of reading the requirements artifacts. On the average, they took the following time per artifact for each round of reading to carry out the reading task: vision and scope document (9 minutes for round 1and 4 minutes for round 2); use case description (10 minutes for round 1 and 5 minutes for round 2); software requirements specification (11.50 minutes for round 1 and 4 minutes for round 2); test plan (10 minutes for round 1 and 5 minutes for round 2); test cases (10 minutes for round 1 and 7.50 minutes for round 2); requirements models (12.50 minutes for round 1 and 5 minutes for round 2).

The SRAR technique is an iterative process. Its step-wise reading and refinement helps in detecting and exponging defects from artifacts. Artifact quality is guaranteed with high eriterations (that is, the more the iterations, the higher the quality). This is true and so in every quality assurance process. However, this quality improvements expectation also depends on the quality of the scoring rubrics and the background of the readers. These two factors were taken care of in the study and thus improves the internal validity of the review. Empiriaclly, improvement in the quality of artifacts was obtained by capturing the difference in the mean rubric scores in the rounds of reading.

V. RESULTS AND DISCUSSION

The following section presents the results of the analysis and the accompanying discussion that follows.

Rubrics Scores

This exploratory reading study was aided by the use of scoring rubrics. The results as shown in Table 5indicate a good improvement in the quality of the requirements work products. Rubric scores (percentage scores) are performance metrics. Among the artifacts read with their corresponding percentage increase in quality include: vision and scope document (19%), list of requirements (100%), use case description (10%), software requirements specification (22%), test plan (39%), test cases (25%), use case diagrams (47%), activity diagrams (25%), collaboration diagrams (22%), and class diagrams (38%).

Table. 5 Mean rubric scores for two rounds of reading: Overall mean: 70.72

Artifacts	Rubric Score (Round 1)	Rubric Score (Round 2)	Difference	% Increase in Quality	Improvement?
Vision & Scope Doc	73.44	87.50	14.06	19	√
List of Reqs	25.00	50.00	25.00	100	√
Use Case Description	81.25	89.29	8.04	10	√
Software Reqs Spec	71.88	87.50	15.62	22	√
Test Plan	62.09	86.25	24.16	39	√
Test Cases	64.29	80.36	16.07	25	√
Mean Work Product (Docs)	56.22	72.71	16.49	29	√
Use Case Diagram	53.57	78.57	25.00	47	√
Activity Diagram	50.00	62.50	12.50	25	√
Sequence Diagram	89.29	78.57	-10.77	-12	X
Collaboration Diagram	64.29	78.57	14.28	22	√
Class Diagram	59.38	82.14	22.76	38	√
Mean Work Products (Models)	63.30	76.07	12.77	20	√
Mean Work Products (Docs & Models)	45.06	56.95	11.89	26	√

From Table 5, the average quality improvement for all 6 textual documents (i.e., vision and scope document, list of requirements, use case description document, software requirements specification, test plan, and test cases) is 29%. Also, the average percentage increase in quality for all the 5 requirements models (use case, activity, sequence, collaboration, and class diagrams) put together is 20%. In addition, the average percentage increase in quality for all work documents read (that is, requirements documents and models) is 26%.

As can be seen, the percentage increase in quality ranged from 10 to 100 percent. Almost all the work products read showed observable improvement in quality. Only one work product had a negative score (-12%, that is, sequence diagrams). Of all the artifacts read, the list of requirements had the highest level of quality enhancement (100%).

However, sequence diagrams were the least in quality (-12%). The lack of improvement of the sequence diagram was

not due to the fact that the employed rubric did not have enough information or that the SRAR process was inadequate to detect the defects. Rather, this may be due to technical oversight from the readers. The overall percentage quality of the entire work documents read is 70.72%. These results indicate a good quality improvement rate. It provides some noticeable evidences to show that the use of SRAR enhanced the quality of the inspected artifacts within the two rounds of reviews on the overall, thus attesting to the SRAR's efficacy.

Rubrics Reliability

The result of the reliability analysis as shown in Table 6 reveals that the Cronbach alpha coefficient (α) of the work products' rubrics scores ranged from 0.867 to 0.986. Vision and scope document with 8 attributes has α = .986. The use case description with 8 attributes has α = 9.78. The software requirements specification (SRS) with 8 attributes has a Cronbach alpha (α) of .970. Also, the test plan documents with 10 attributes has a Cronbach alpha (α) of .040. Other work products with their corresponding number of attributes and α coefficient are as follows: test case documents (7, .931), use case diagram (7, .964), activity diagram (2, .867), sequence diagram (7, .981), collaboration diagram (7, .974), class diagram (8, .966).

Furthermore, the overall reliability coefficient is 0.987. Overall, there werea total of 72 attributes. The Cronbach alpha scores are indicative of a very good internal consistency. Reliability scores that are 0.70 and above, are normally regarded as good reliability estimates (Nunnaly, 1978); therefore, the used scoring rubrics are highly reliable and their use can produce consistent results. They are thus reliable for the detection of defects and for the improvement of the quality of the documents.

Table. 6 Reliability analysis of artifacts rubrics: Overall Cronbach Alpha coefficient: 0.987

Artifacts	No of Attributes	Cronbach Alpha
Vision & Scope Doc	8	.986
Use Case Desc	8	.978
SRS	8	.970
Test Plan	10	.940
Test Cases	7	.931
Use Case Diagram	7	.964
Activity Diagram	2	.867
Sequence Diagram	7	.981
Collaboration Diagram	7	.974
Class Diagram	8	.966
Overall	72	.987

In sum, it is observed that after round two of the reading, there were observable improvements in the work products and artifacts' quality levels as seen in the metric used. This shows the efficacy of the reading approach. The rubric score for all textual documents put together, improved in quality by 29%. Furthermore, the rubric score of all requirements models combined improved by 20% and the one of requirements models combined with textual documents improved by 26%. The overall rubric score of 70.72% shows that on the overall, the average quality of the work product was good.

V. CONCLUSION AND FUTURE WORK

The quality of requirements documents and of software relies on how satisfactory the documents are read. A number of reading techniques have been proposed in the literature. These techniques help in the reading and improvement of the quality of software artifacts. However, in literature, the quality improvement efficacies of the various techniques are at varying levels. The most common reading technique is checklist reading technique. This technique, even though is an enhancement over the ad hoc reading technique, is nonetheless, encumbered with some drawbacks that affect its usability, effectiveness, efficiency and efficacy as it offers too broad questions to readers devoid of any precise guidance on how to undertake the reading process. This paper evaluates SRAR, a checklist-like reading technique for the detection of defects in and refinement of requirements artifacts. Initial outcomes reveal the efficacy of this technique and showed that it improves the quality of software requirements artifacts.

The main finding of the study is that the use of SRAR helped in the improvement of the quality of requirements artifacts. The quality improvement ranged from 10 to 100% depending on the artifact, except for sequence diagram. Thus, SRAR promises to be a reading technique with good efficacy, which overcomes the inadequacies of the checklist and ad hoc reading techniques. It is simple, guiding, defined, consistent and scoring as well. It has the capacity of assisting readers detect defects in software/requirements work products and improving the quality of the given artifacts. In addition, it can be used to monitor the progress of quality improvements of artifacts. SRAR is conceptually better than checklist and ad hoc techniques in that it provides checklist-like rubrics which these techniques do not provide to readers. The attributes of these rubrics guide readers to achieve high level quality defect detection and subsequently assists the analyst(s) to get a high level artifact refinement and defect removal.

In a nutshell, this study has adequately answered the research question that was posed at the beginning of the study: i.e., Does SRAR technique improves the quality of requirements work documents and artifacts? SRAR does improve the quality of software artifacts and software requirements artifacts as evidenced in the improved quality after the second round of refinement following the use of SRAR. The study will benefit software engineering and software requirements engineering professionals and the research community as this technique is conceptually a better reading approach than checklist and ad hoc as the study alludes to.

However, this study was an exploratory one and was limited in the sense that SRAR was not empirically and

experimentally compared with checklist, ad hoc, or any other reading technique, but the efficacy was judged based on the defect detection capability and quality improvement potentials as observed from the study. Future works will concentrate on evaluating and comparing SRAR with other reading techniques using both professional and non-professional users/readers.

REFERENCES

1. Alshazly, A.A., Elfatatry, A.M., & Abougabal, M.S. (2014). Detecting defects in software requirements specification, *Alexandria Engineering Journal*, 53(3), 513-527.
2. Basili, V., Caldiera. G., Lanubile, F., & Shull F. (1996). Studies on reading techniques. Institute of Advanced Computer Studies, University of Maryland, USA.
3. Berling T., Runeson P. (2003). Evaluation of a perspective based review method applied in an industrial setting, *IEE Proceedings on Software*, 150(3), 177-184.
4. Dessai, K.G.G., Kamat, V.V., & Wagh, R.S. (2014). Effective use of rubrics in computer assissted subjective answer script evaluation. *2014 IEEE 6thInternational Conference on Technology for Education (T4E)*, Clappana, 18-21 Dec. 2014, p. 95-98.
5. Dietrich, S. (2008). Quality-based assessment of papers and projects in computer science. *J Comp Sci Colleg*. 23(3), 16-22.
6. Halling, M., Biffl, S., Grechenig, T., & Kohle, M. (2001). Using reading techniques to focus inspection performance. Proceedings of the 27th 2001 IEEE Euromicro Conference, Warsaw, 4-6 September, 2001,pp.248-257.
7. Jonsson, A., & Svingby, G. (2007). The use of scoring rubrics: reliability, validity and educational consequences. *Education Research Review*,2, 130-144.
8. Hussain, A., &Mkpojiogu, E.O.C. (2016). An application of Kano method in the elicitation of stakeholder satisfying requirements for an e-Ebola awareness system. *International Journal of Systems Applications, Engineering and Development*, 10, 169-178.
9. Hussain, A., Mkpojiogu, E.O.C., &Nawi, M.N.M. (2016). Requirements model for an e-health awareness portal. *Proceedings of the International Conference on Applied Science and Technology (ICAST'16),* Kedah, Malaysia, AIP Conf. Proc., 1761 (1), 020048
10. Kollanus, S., &Koskinen, J. (2007). Survey of sftware inspection research: 1991-2005. Computer Science and Information Systems Reports Working Papers WP-40. Department of Computer Science and Information Systems, University of Jyvaskyla, Finland.
11. McKenzie, S., &Wood-Bradley, G. (2014). Using rubrics in IT: Experiences of assessment and feedback at Deakin University. *2014 International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, Welligton, 8-10 Dec., 2014, pp. 474-479.
12. Mkpojiogu, E.O.C., &Hashim, N.L. (2017a). Improving the quality of requirements work products using scoring rubrics-assisted reading.6th *International Conference on Computing and Informatics 2017 (ICOCI'17),* 25-27 April, 2017, Kuala Lumpur, Malaysia.
13. Mkpojiogu, E.O.C. &Hashim, N.L. (2017b). Evaluating the effectiveness of scoring rubrics assisted reading (SRAR) technique in the reading of software requirements work documents. *Journal of Engineering and Applied Sciences (JEAS)*, 12(21), 5548-5553
14. Mkpojiogu, E.O.C. &Hashim, N.L. (2017c). Assessing the efficiency of scoring rubrics assisted reading approach in the review of software requirements work products. *Journal of Engineering and Applied Sciences (JEAS)*, 12(8 SI), 8355-8359.
15. Mkpojiogu, E.O.C. & Hussain, A. (2017a). Can scoring rubrics be used in assessing the performance of students in software requirements engineering education? *Journal of Telecommunication, Electronic & Computer Engineering (JTEC)*.9 (2-11), 115-119
16. Mkpojiogu, E.O.C., & Hussain, A. (2017b). Assessing students' performance in software requirements engineering education using scoring rubrics. *Proceedings of the 2nd International Conference on Applied Science and Technology (ICAST'17),* Langkawi Island, Malaysia, 3-5 April, 2017. AIP Conference Proceedings 1891 (1), 020092, http://doi.org/10.1063/1.5005425
17. Mohammed, A.O., Maatuk, A.M., Abdelnabi, Z.A., & Abdulla, A.S. (2015). An experimental study on detecting semantic defects in objects in object-oriented programs using software reading techniques, *Proceedings of the International Conference on Engineering & MIS (ICEMIS-15)*, September 24-26, 2015, Istanbul. Turkey.
18. Nunnaly, J. (1978). *Psychometric Theory*, New York: McGraw-Hill.
19. Oladele, R.O., &Adedayo, H.O. (2014). On empirical comparison of checklist-based reading and ad hoc reading for code inspection, International Journal of Computer Application, 87(1), 35.
20. Sherman, M., &Martin, F. (2015). The assessment of mobile computational thinking. *J Comp SciColleg*, 30(6), 53-59.
21. Shull, F., Rus, I.., & Basili, V. (2000). How perspective based reading can improve requirements inspections, *Computer*, 33(7), 73-79.
22. Sommerville, I. (2007). *Software Engineering*. Eigth Edition, Addison-Wesley.
23. Wohlin, C., Runeson, P., Host, M., Ohlsson, M.C., Regnell, B., & Wesslen, A. (2012). Are the persepectives really different?: further experimentation on scenario-based reading of requirements, *Experimentation in Software Engineering*, Springer Berling Heidelberg, pp. 175-200.