Assessing the Quality of Software Requirements Specifications for Automotive Software Systems

Akiyuki Takoshima
Electronics Platform R & D Div.
DENSO CORPORATION.
Kariya, Japan
akiyuki_takoshima@denso.co.jp

Mikio Aoyama
Department of Software Engineering
Nanzan University
Nagoya, Japan
mikio.aoyama@nifty.com

***Abstract*— As the size and complexity of automotive software increases, software requirements specification (SRS) is gaining importance in automotive software system development. However, the diversity in SRSs, which comes from the difference between product characteristics, is becoming an impediment for systematic inspection. We propose a two-stage inspection that combines first-stage inspection by third-party inspectors and second-stage inspection by the project team, which would allow SRSs written in different formats to be inspected in a uniform way. The scope of this paper is limited to third-party inspection. The proposed methodology leads to the following three benefits: 1) systematic inspection of SRSs from different product domains 2) analysis of the inspection results from multiple viewpoints 3) benchmark of inspection results and SRS quality. We applied the proposed method to two actual products' SRSs and evaluated the effect of the method.**

Keywords*— *Software Requirements Specifications; Quality Characteristics; Inspection; Automotive

I. INTRODUCTION

As software adds its weight in automobiles, automotive software increases in size and complexity [2][6]. The introduction of Requirements Engineering (RE) is a key success factor to the development of a complex product within the predefined costs and scheduling [5]. Since the automotive industry has traditionally been organized in a highly vertical manner [9], product development involves many different organizations, i.e. companies or divisions. Therefore, software requirements specification (SRS) plays a particularly important role as a means to ensure that the different organizations have the shared understanding of the development goals.

It is known that SRS quality is the most significant factor in a development project's success and failure [11]. Automotive SPICE [1] is in widespread use as the de facto standard development process in the automotive industry and many organizations conduct SRS inspection. On the contrary, an industrial survey revealed that requirements inspection practices are still rather ad hoc with many projects and organizations defining the rules and procedures for the requirements inspection [4].

One possible factor that prevents proper inspection in the automotive industry is the variety in SRSs. Automotive software can be clustered according to the application domains and the associated non-functional requirements: In-vehicle infotainment, Body/comfort, Safety, Power train and Chassis control, and Infrastructure [9]. DENSO has product lineups in all of the five product domains and SRS structures vary in each domain reflecting the specific characteristics of the domain requirements. Inspection at each development project consequently has the tendency to become an ad hoc activity rather than a company-wide and uniform activity. The performance of an inspection, as a result, largely depends upon the inspector's experience and domain knowledge.

We propose a two-stage inspection that combines first-stage inspection by third-party inspectors and second-stage inspection by the project team, which consists of a project manager, developers, testers, and some other roles. These two stages have different purposes. The first-stage is design to acquire documentation quality and the second stage is designed to acquire requirements quality. It is assumed that third-party inspectors do not have much domain knowledge and project members naturally have it. However, the scope of this paper is limited to third-party inspection. This paper answers the following research questions through an industrial case study at DENSO.

RQ1. Is it possible to apply a uniform inspection method to SRSs written in different formats?

RQ2. Is it possible to make a relative comparison of inspection results for different SRSs?

RQ3. Is IEEE Std. 830-1998 template of SRS adequate for automotive software system requirements?

RQ4. Are IEEE Std. 830-1998 quality characteristics adequate for automotive software system requirements?

The rest of this paper is structured as follows. Chapter II discusses related works. The approach of this research is illustrated in Chapter III. Chapter IV provides the overview of our proposed methodology. New quality characteristics are derived in Chapter V and Chapter VI defines the inspection

1530-1362/15 $31.00 © 2015 IEEE
DOI 10.1109/APSEC.2015.57

guideline for measuring those quality characteristics. Chapter VII discusses the experiences of the methodology. Chapter VIII discusses the contribution of this work, followed by the conclusions and future works in Chapter IX.

II. Related Works

A. Quality Measurement of SRS

As for SRS quality characteristics, eight quality characteristics defined in IEEE Std. 830-1998 [7] are primarily used. However, IEEE Std. 830-1998 does not specify how to measure these quality characteristics. Davis et al. proposed 24 SRS quality attributes and techniques for measuring these attributes [3]. However, these techniques are rather vague when used to inspect an SRS written in natural languages and therefore not good enough for applying to product development.

B. Design Methodology of Systematic Inspection Methods

Saito et al. [10] proposed a methodology (RISDM) to design inspection methods in which the quality of an SRS can be quantitatively measured and thus compared relatively to that of another SRS. RISDM has a design process and an inspection process. In the design process, the standard SRS and pragmatic quality characteristics (PQC), which measures the quality of the standard SRS, are defined. In inspection process, the project SRS created by the development team by tailoring the standard SRS is inspected in line with the designed inspection system. Therefore, RISDM's scope of inspection is limited to the standard SRS or project SRSs tailored from the standard SRS. However, a new product's SRS is often created by updating (modifying existing features and adding new features) the previous product's SRS in the automotive industry. This makes it quite difficult to replace the SRS template. From this reason, RISDM comes short of the ability to inspect SRSs for automotive software that has different formats in different product domains.

III. Approach

A. Assessment of SRS Quality

We establish a methodology for assessing automotive software system SRSs that vary in formats for different product domains. We expand RISDM [10] methodology to establish such a methodology.

B. Defining Reference SRS and Translation Matrix

We define a reference SRS in order to accommodate the differences in the formats of project SRSs and to inspect different project SRSs in a uniform process. The relation between a project SRS and the reference SRS is defined as a translation matrix. The translation matrix associates the contents of a project SRS to those of the reference SRS. Table 1 illustrates the translation matrix.

C. Techniques for Measuring SRS Quality Characteristics

To define measurable quality characteristics, we derive new quality characteristics from the eight quality characteristics defined in IEEE Std. 830-1998. We also define a question set as a practical means to measure these derived quality characteristics.

IV. Framework of Proposed Method

Fig. 1 illustrates the framework of the proposed methodology. This paper defines the following six items: five elements in the framework and the process to conduct inspection in line with the defined elements. A Translation matrix is defined in Chapter III. The Process of the proposed inspection methodology, and Reference SRS are defined in this Chapter. A set of quality characteristics is defined in Chapter V and an inspection matrix and associated inspection guideline are defined in Chapter VI, respectively.

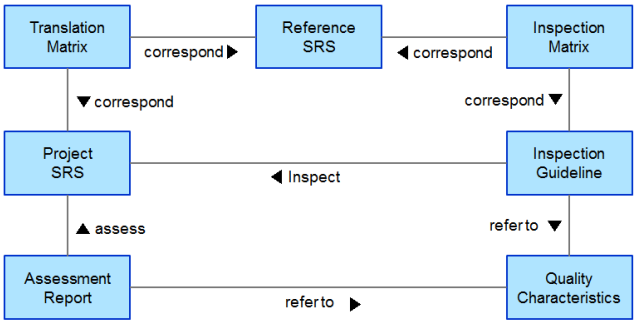
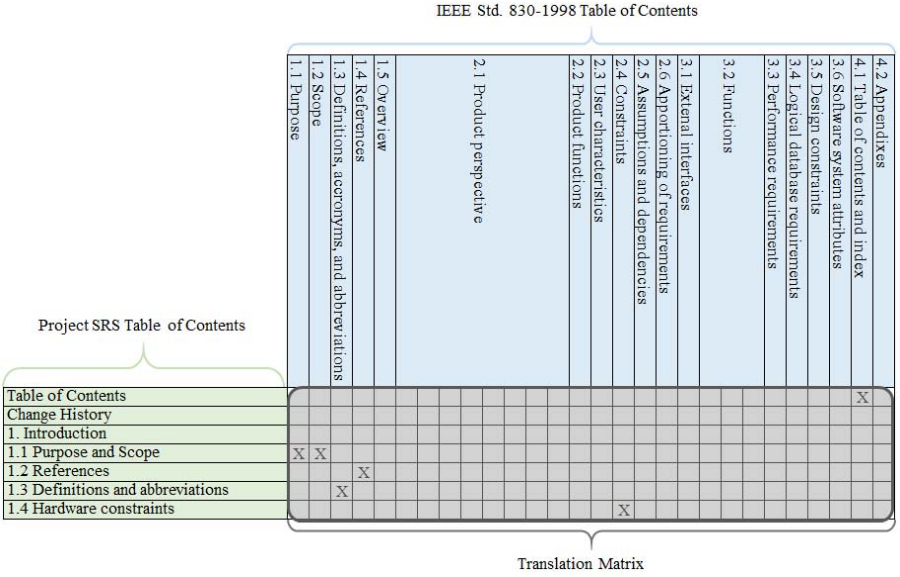


Fig. 1 The Framework of Proposed Methodology

Table 1. Translation Matrix



A. *The Process of the Proposed Inspection Methodology*

In our proposed methodology, inspection is conducted in accordance with the process shown in Fig. 2.

(1) Receipt of SRS by the third-party inspector: The third-party inspector independent from the project, which consists of a requirements analyst and a development team, receives the project SRS drafted by the requirements analyst.
(2) First-stage inspection – Inspection conducted by the third-party inspector: The third-party inspector independent from the project conducts SRS inspection.
(3) Evaluation of the first-stage inspection:
  1) The inspector forwards the SRS to the project team if the inspection result satisfies the acceptance criteria.
  2) The inspector returns the SRS to the requirements analyst with an assessment report that provides the quality score and the advice for improvement if the inspection result does not satisfy the acceptance criteria.
(4) Receipt of SRS by the project: The project team receives the SRS that the third-party inspector judged to satisfy the acceptance criteria.
(5) Second-stage inspection – Inspection conducted by the project: A project manager, developers, testers, and some other roles that constitute the project team inspect the SRS utilizing their domain knowledge.
(6) Evaluation of the second-stage inspection:
  1) The project establishes a requirements baseline and begins software development if the inspection result satisfies the acceptance criteria.
  2) The project returns the SRS to the requirements analyst with an analysis report that lists the findings from the inspection.
(7) Make improvement:
  1) When the requirements analyst receives an assessment report from the third-party inspector, the analyst makes an improvement on the SRS based on the improvement advice.
  2) When the requirements analyst received an analysis report from the project, the analyst answers the findings and makes an improvement on the SRS if necessary.

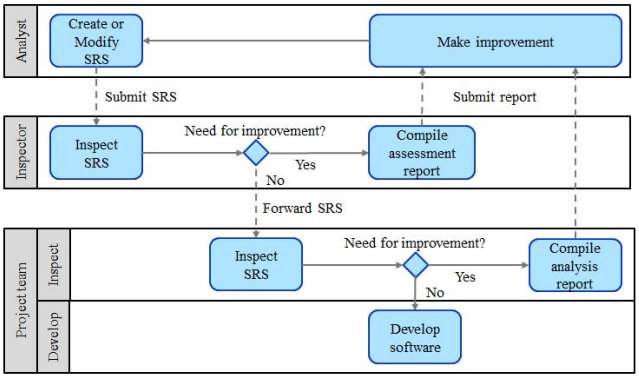


Fig. 2 Inspection Process

B. *Reference SRS*

The appropriate structure of a reference SRS varies in each product domain. This paper adopts the IEEE Std. 830-1998 SRS template as the reference SRS. We verify how well the IEEE Std. 830-1998 SRS template fits to automotive software system requirements in the course of our trial of the proposed method.

V. Deriving Quality Characteristics

We adopt the eight quality characteristics from IEEE Std. 830-1998. We also adopt another quality characteristic, achievable, which is listed one of the 24 quality characteristics in [3] and defined these nine quality characteristics as the reference quality characteristics. It is ideal that third-party inspectors can measure all the reference SRS quality characteristics. But the reality is that a part of the quality characteristics requires domain knowledge related to the requirements the SRS describes. Therefore, we separate the reference quality characteristics into third-party measurable and third-party unmeasurable. In the following sections, we present how we derive a set of new quality characteristics from the reference quality characteristics.

A. *Correct*

An SRS is correct if, and only if, every requirement stated therein is one that the software shall meet [7]. The third-party inspector cannot judge whether an SRS meets this criterion. Thus we define a new quality characteristic that assures the possibility of measuring correctness during the second-stage inspection by the project. The third-party inspector measures the new quality characteristics during the first-stage inspection.

We define accountability as a new quality characteristic. An SRS is accountable if and only if, the objective of developing the software system and the purpose of software features are clearly stated. If accountability is satisfied, it is reasonable to assume that the project members with domain knowledge can verify each requirement in the context of the development objectives and the purpose of features and verify whether each requirement is really necessary i.e. correct.

B. *Unambiguous*

An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation [7]. It is impossible to completely remove the ambiguity from an SRS written in natural language. However, the third-party inspector can determine the existence of ambiguity by answering the predefined closed questions.

C. *Complete*

An SRS is complete if, and only if, it includes the following elements [7]:

1 All significant requirements, whether relating to functionality, performance, design constraints, attributes, or external interfaces
2 Definition of the responses of the software to all realizable classes of input data in all realizable situations

3 Full labels and references to all figures, tables, and diagrams in the SRS and definition of all terms and units of measure

IEEE Std. 830-1998 lists the above three perspectives regarding completeness. The first and second perspectives relate to completeness of requirements. On the other hand, the third perspective relates to completeness of description. Third-party inspector can evaluate only the completeness of descriptions. It is reasonable to assume that the completeness of requirements, as with correctness, can be evaluated during the second-stage inspection by the project if accountability is satisfied. Thus we define two new quality characteristics, namely: descriptive completeness and requirement completeness.

D. *Consistent*

An SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict [7]. It is virtually impossible for third-party inspectors without domain knowledge to find inconsistencies in an SRS written in a natural language. Therefore, we do not derive new quality characteristics from this quality characteristic. We simply assign this quality characteristic to the second-stage inspection by the project.

E. *Ranked for importance and/or stability*

An SRS is ranked for importance and/or stability if each requirement in it has an identifier to indicate either the importance or stability of that particular requirement [7]. However the initial scope of requirements is seldom shrunk or changed for automotive software development. An industrial survey [5] shows that there are few unnecessary or optional needs, and these needs are generally not prioritized in aerospace or energy domain. Thus we eliminated this quality characteristic from the target of measurement.

F. *Verifiable*

An SRS is verifiable if, and only if, every requirement stated therein is verifiable. A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement. In general, any ambiguous requirement is not verifiable [7]. As IEEE Std. 830-1998 explains, requirements are not verifiable if they are ambiguous. To put it another way, requirements are verifiable if they do no have ambiguity. Consequently, we combine unambiguous and verifiable and derive a new quality characteristic: Definiteness.

G. *Modifiable*

An SRS is modifiable if, and only if, its structure and style are such that any changes to the requirements can be made easily, completely, and consistently while retaining the structure and style [7]. Because modifiable can be measured by answering the predefined closed questions, we do not derive new quality characteristics from this quality characteristic.

H. *Traceable*

An SRS is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation [7]. Because traceable can be measured by answering the predefined closed questions, we do not derive new quality characteristics from this quality characteristic.

I. *Achievable*

An SRS is achievable if and only if there could exist at least one system design and implementation that correctly implements all the requirements stated in the SRS [3]. It is virtually impossible for third-party inspectors without domain knowledge to conclude if there could exist at least one possible system design. Therefore, we do not derive new quality characteristics from this quality characteristic. We simply assign this quality characteristic to the second-stage inspection by the project.

Fig. 3 illustrates the relation between the reference quality characteristics and the derived quality characteristics. As already mentioned, third-party inspection cannot cover all of the reference quality characteristics. However, it is reasonable to assume that all of the reference quality characteristics except for ranked for importance and/or stability can be measured by deriving new quality characteristics and assigning them to either third-party inspection or project inspection.

VI. Inspection Method

We define inspection viewpoints for each inspection point. Inspection viewpoints are defined as closed questions that can be objectively answered in the form of yes or no by third-party inspectors who do not have domain knowledge. 15 questions comprising question set and corresponding to each quality sub-characteristic are shown in Table 3.

We call the paring of the inspection matrix and the question set the inspection guideline. This inspection guideline and aforementioned translation matrix help third-party inspectors who do not have domain knowledge uniquely identify which quality characteristics to measure for each SRS

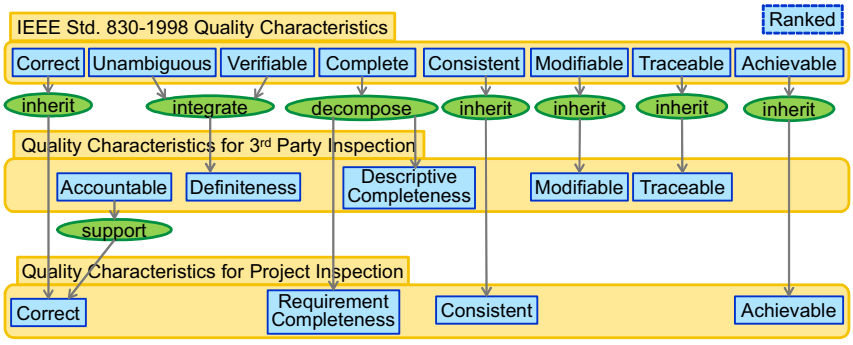


Fig. 3 Derived Quality Characteristics

part. Moreover, the identified quality characteristics can be objectively measured by answering the closed questions.

A. *Techniques for measuring the quality characteristics*

Because all parts of the reference SRS are not related to all of the quality characteristics, we define an inspection matrix that associates the table of contents of the reference SRS to quality characteristics to be measured (Table 2). The cells marked with 'X' specify the inspection points. Take 1.1 Purpose as an example, the matrix tells you that quality sub-characteristics C2-6 and C3-3 need to be measured for this section. In our case, the total number of inspection points became 145 points.

We also consider the techniques to quantify the quality characteristics measurement results to compare the SRS quality between different SRSs. We adopt the same grading method to RISDM [10]. Since each inspection point has a simple closed question that can be answered by yes or no, we give one point to the inspection point answered with yes and zero points to the point answered with no.

VII. TRIAL AND EVALUATION WITH ACTUAL SRS

A. *Verification Method*

We applied the proposed methodology to two actual SRSs. Both of the SRSs are for products in the body/comfort domain, but they belong to different product categories, so they are created from different SRS templates. We describe these SRSs as SRS_A and SRS_B in the rest of this paper.

Table 2 Inspection Matrix

Quality Sub-characteristics

IEEE Std. 830-1998 Table of Contents	C1-1	C1-2	C2-1	C2-2	C2-3	C2-4	C2-5	C2-6	C3-1	C3-2	C3-3	C4-1	C4-2	C5-1	C5-2
1. Introduction															
1.1 Purpose								X			X				
1.2 Scope	X							X			X				
1.3 Definitions, accronyms, and abbreviations								X			X				
1.4 References											X				
1.5 Overview								X			X				
2. Overall description															
2.1 Product perspective															
2.1.1 System interfaces				X				X	X	X	X				
2.1.2 User interfaces				X				X	X	X	X				
2.1.3 Hardware interfaces				X				X	X	X	X				
2.1.4 Software interfaces				X				X	X	X	X				
2.1.5 Communications intrfaces				X				X	X	X	X				
2.1.6 Memory constraints				X	X			X	X	X	X				
2.1.7 Operations		X		X				X	X	X	X				
2.1.8 Site adaptation requirements		X		X				X	X	X	X				
2.2 Product functions				X				X	X	X	X				
2.3 User characteristics				X				X	X	X	X				
2.4 Constraints				X				X	X	X					
2.5 Assumptions and dependencies				X				X	X	X	X				
2.6 Apportioning of requirements				X				X	X	X	X				
3. Specific requirements															
3.1 Extenal interfaces				X				X	X	X	X				
3.2 Functions															
3.2.1 System Feature															
3.2.1.1 Introduction/Purpose of feature		X		X				X	X	X					
3.2.1.2 Stimulus/Response sequence				X	X	X		X	X	X	X				
3.2.1.3 Associated functional requirements			X	X	X	X	X	X	X	X		X		X	X
3.3 Performance requirements			X	X	X	X	X	X	X	X		X		X	X
3.4 Logical database requirements			X	X	X	X	X	X	X	X		X		X	X
3.5 Design constraints				X		X		X	X	X					
3.6 Software system attributes			X	X	X	X	X	X	X	X	X	X		X	X
4. Supporting information													X		
4.1 Table of contents and index															
4.2 Appendixes															

Inspection Matrix

B. *Defining Translation Matrix*

We create a translation matrix to associate the contents of a project SRS to those of the reference SRS. Table 4 illustrates how SRS_A can be associated to the reference SRS.

C. *Evaluation Results*

We answer the four research questions through the evaluation results for our trial of the proposed methodology.

First, we inspected both SRS_A and SRS_B with a uniform method with the aid of translation matrices required by the proposed methodology. The evaluation results of both SRSs are shown from Fig. 4 to Fig. 11.

Second, we verify if relative comparison of inspection results

Table 3 Question Set

ID	Quality Characterisitc	Sub-characteristic	Question
C1-1	Accountability	Project objective	The objective of developping the software is stated?
C1-2		Purpose of feature	The purpose of the function is stated?
C2-1	Definiteness	Non-redundant	Multiple requirements are not included in a single sentence?
C2-2		Non-equivocal	Any equivocal terms are used withoug defining its meanings?
C2-3		Quantative	Qualitative expression is not used where quantative expression should be used?
C2-4		Freedom from ambiguity	Ambiguous expressions are not used?
C2-5		Voice	The sentence is written in the active voice?
C2-6		Reference	If external documents are refered, reference section provides enough information to locate the documents?
C3-1	Descriptive completeness	TBD	Each TBD is accompanied by a) How to resolve b) Due date c) Responsible person d) Tracking ID
C3-2		Label	A unique number is assigned to Figures and tables?
C3-3		Template	All elements required by the standard SRS are included?
C4-1	Modifiable	Cross reference	Cross-reference is provided to the appropreate requirement when a requirement refer to anotherr requiremet?
C4-2		Searchable	Table of contents and index are created?
C5-1	Traceable	Backward traceablility	Traceability matrix that specifies the link between high-level requirements and software requirements?
C5-2		Forward traceability	Each requirement has a unique identification?

between different SRSs is possible. Fig. 8 and Fig. 9 show the quality score per chapter for SRS_A and SRS_B. As you can see from the graphs, SRS_A has seven chapters and SRS_B has ten chapters because they are based on the different SRS templates. This difference prohibits direct comparison of the quality of these two SRSs. Thus, we aggregated each product's quality score per the contents of the reference SRS so as to make relative comparison between project SRSs. Fig. 10 and Fig. 11 illustrate the quality score per chapter of the reference SRS. As just described, relative comparison of quality between project SRSs become possible if measurement results of quality characteristics are aggregated in line with the reference SRS structure.

SRS quality score information gathered from inspection can be shown in different views. We can earn new insight when we make an analysis from a different view. If we analyze the quality score from quality characteristics and sub-characteristics view, we can understand which characteristics need to be improved in all over the SRS. Take SRS_A as an example, we can learn from Fig. 4 that accountability needs to be improved. Next, if we analyze the quality score from chapter-wise view, we can learn from Fig. 8 that SRS_A has quite high score overall but only chapter 1 falls below 80% and there still exists a margin for improvement. Finally, if we analyze the quality score from the reference SRS view, we can compare the score with other projects' SRSs. This comparison clarifies the relative position of the SRS among other project SRS in the organization.

Thirdly, we assess the applicability of IEEE Std. 830-1998 SRS template to automotive software system requirements. We inspected two project SRSs with IEEE Std. 830-1998 as the reference SRS and we have identified two important attributes of automotive software systems that IEEE Std. 830-1998 comes short of the ability to describe the requirements in well-structured way.

1) Variants and Configurations: Automotive systems have significantly many variations to differentiate vehicle grades. A premium car typically has about 80 electronic fittings that can be orderd depending on the country, etc [9]. Because of this variation, recent automotive software system development is shifting towards software product line engineering (SPLE). However, commonality and variability, which are vitaly important for SPLE, are not considered in IEEE Std. 830-1998.

2) Requirements in Different Abstraction Levels: In automotive software system development, car manufacturers provide requirements to suppliers written in different abstraction levels. Some are in very high level and explain what problem the software system should solve and other are in very low level and described in the form of solution that strictly constrains design. IEEE Std. 830-1998 advises that SRS should not normally specify design. Davis et al. also listed design independent as a quality characteristic [3]. One

Table 4 Translation Matrix for SRS_A

				1. Purpose	1.2 Scope	1.3 Definitions, acronyms, and abbreviation	1.4 References	1.5 Overview	2.1 System interfaces	2.1.1 User interfaces	2.1.2 Hardware interfaces	2.1.3 Software interfaces	2.1.4 Communications interfaces	2.1.5 Communications interfaces	2.1.6 Memory constraints	2.1.7 Operations	2.1.8 Site adaptation requirements	2.2 Product functions	2.3 User characteristics	2.4 Constraints	2.5 Assumptions and dependencies	2.6 Apportioning of requirements	3.1.1 Introduction/Purpose of feature	3.1.1 External Interfaces	3.1.2 Stimulus/Response sequence	3.2.1.3 Associated functional requirements	3.3 Performance requirements	3.4 Logical database requirements	3.5 Design constraints	3.6 Software system attributes	4.1 Table of contents and index	4.2 Appendixes
																															X
1																															
	1.1				X	X																									
	1.2						X																								
	1.3					X																									
	1.4																				X										
2																															
	2.1											X																				
	2.2																															
	2.3																															
3																															
	3.1																							X		X						
	3.2																							X								
		3.2.1																														
			3.2.1.1																							X						
			3.2.1.2																													
			3.2.1.3																							X						
			3.2.1.4																							X						
			3.2.1.5																													
			3.2.1.6																													
			3.2.1.7																													
			3.2.1.8																							X						
			3.2.1.9																							X						
			3.2.1.10																							X						
			3.2.1.11																							X						
			3.2.1.12																							X						
			3.2.1.13																													
			3.2.1.14																							X						
		3.2.2																						X								
			3.2.2.1																							X						
			3.2.2.1																													
		3.2.3																						X								
			3.2.3.1																							X						
			3.2.3.2																							X						
	3.3																							X		X						
		3.3.1																								X						
		3.3.2																								X						
		3.3.3																								X						
		3.3.4																								X						
	3.4																															
	3.5																							X		X						
		3.5.1																								X						
	3.6																							X								
		3.6.1																								X						
		3.6.2																								X						
4																															
	4.1																							X								
		4.1.1																								X						
		4.1.2																								X						
	4.2																															
		4.2.1																								X						
		4.2.2																								X						
	4.3																															
		4.3.1																								X						
		4.3.2																								X						
		4.3.3																								X						
	4.4																															
		4.4.1																								X						
		4.4.2																	X							X						
	4.5																															
		4.5.1																														
			4.5.1.1																											X		
			4.5.1.2																											X		
		4.5.2																						X								
			4.5.2.1																								X						
			4.5.2.2																								X						
		4.5.3																						X								
			4.5.3.1																								X						
			4.5.3.2																								X						
		4.5.4																						X		X						
	4.6																							X		X						
	4.7																							X		X						
	4.8																							X		X						
5																															
	5.1																									X						
	5.2																									X						
	5.3																									X						
	5.4																									X						
	5.5																					X										
	5.6																					X										
	5.7																															
	5.8																										X					
	5.9																															
	5.1																															
6																															
	6.1																											X				
	6.2																															
	6.3																										X					
	6.4																															
7																															
	7.1																															
	7.2																							X								
		7.2.1																						X								
		7.2.2																														
		7.2.3																														
		7.2.4																														
		7.2.5																										X				
	7.3																															
	7.4																															
		7.4.1																														
		7.4.2																														
		7.4.3																														
	7.5																															

reason that requirements should not constrain design is that it prevents developer from optimizing the design. Moreover, an industrial survey revealed that the most common defect in requirements is expressing needs in the form of solution [5]. Therefore, it is vitaly important to separate requirements (problem space) and design (solution space). On the other hand, Weber et al. enumerated reasons why an SRS for automotive software system should step into the solution space [12]. From above reasons, an SRS for automotive software system needs to manage requirements written in very different level of abstraction. However, the SRS template and guidline provided by IEEE Std. 830-1998 is not sufficient to capture requirements in various level of abstraction.

Lastly, we assess the applicability of IEEE Std. 830-1998 quality characteristics to evaluate automotive software system requirements. We started with IEEE Std. 830-1998 quality characteristics to redefine appropriate quality characteristics to measure the quality of automotive software system requirements. We found that IEEE Std. 830-1998 lacked one quality characteristic, i.e. achievable, which is important in developer's perspective.

We found that IEEE Std. 830-1998 lacks some aspects to describe and evaluate requirements in automotive software systems. On the other hand, IEEE Std. 830-1998 may have some aspects that have nothing to do with automotive software systems. Since our research have examined only two SRSs so far, we need to verify more SRSs gathered from wide variety of product domains to identify the unnecessary aspects if any.

VIII. DISCUSSIONS

Our proposed methodology is independent from SRS formats therefore can be used in various organizations. Organizations that want to shift the inspection from project-wise ad hoc activity to corporate-wide systematic activity can take advantage of the proposed methodology.

IX. CONCLUSIONS AND FUTURE WORKS

This paper proposed a methodology to uniformly inspect project SRSs written in different formats that reflect product domain characteristics. We applied the methodology to actual SRSs and evaluated the results. The evaluation result proved that the methodology can be used as a framework for uniform inspection of SRSs created from different templates. It was confirmed that quality score information gathered from inspection provides useful insight for improving SRS because the information offers a multifaceted perspectives on SRS quality. We found that IEEE Std. 830-1998 SRS template is not sufficient to manage variants and abstraction level, which are two important aspects in automotive software systems. We also found that IEEE Std. 830-1998 quality characteristics lack a quality characteristic that evaluates the technical feasibility of requirements.

For the future works, we plan to collect more SRSs from widespread product domains and apply the proposed methodology to them in order to improve inspection accuracy and to accumulate data for SRS quality benchmark since we evaluated the method only with two SRSs both from the body/comfort domain. We also plan to enhance IEEE Std. 830-1998 SRS template and establish a reference SRS that is more suitable for requirements in automotive software systems.

REFERENCES

[1] Automotive SPICE Process Assessment Model v.2.5

[2] P. Braun, M. Broy, F. Houdek, M. Kirchmayr, M. Müller, B. Penzenstadler, K. Pohl, and T. Weyer, Guiding Requirements Engineering for Software-Intensive Embedded Systems in the Automotive Industry, Computer Science - Research and Development, Vo. 29, No. 1, Springer, Feb. 2014, pp. 21-43.

[3] A. Davis, et al., Identifying and Measuring Quality in a Software Requirements Specification, Proc. of the First Int'l Software Metrics Symposium, IEEE Computer Society, May 1993, pp. 141-152.

[4] G. Fanmuy, et al., Requirements Verification in the Industry, Proc. of CSDM 2011, Springer, Dec. 2011, pp. 145-160

[5] G. Fanmuy and G. Foughali, A Survey on Industrial Practices in Requirements Engineering, INCOSE International Symposium, Vol. 22, No. 1, Ju. 2012, pp. 1021-1040.

[6] K. Grimm, Software Technology in an Automotive Company - Major Challenges, Proc. ICSE 03, IEEE Computer Society, May. 2003, pp. 498-503.

[7] IEEE, Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications, IEEE Computer Society, 1998.

[8] O. Laitenbergera and J.-M. DeBaud An Encompassing Life Cycle Centric Survey of Software Inspection, J. of Systems and Software, Vol. 50, No. 1, Jan. 2000, pp. 5-31.

[9] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner, Software Engineering for Automotive Systems: A Roadmap, Proc. of FOSE '07 2007 Future of Software Engineering, IEEE Computer Society, May 2007, pp. 55-71.

[10] S. Saito, M. Takeuchi, S. Yamada, and M. Aoyama, RISDM: A Requirements Inspection Systems Design Methodology, Proc. of RE 2014, IEEE Computer Society, Aug. 2014, pp. 223-232.

[11] The Standish Group, The CHAOS Report (2015), 2015. http://www.standishgroup.com/Reports2015.

[12] M. Weber and J. Weisbrod, Requirements Engineering in Automotive Development: Experiences and Challenges, IEEE Software, Vol. 20, No. 1, Jan./Feb. 2003, pp. 16-24.

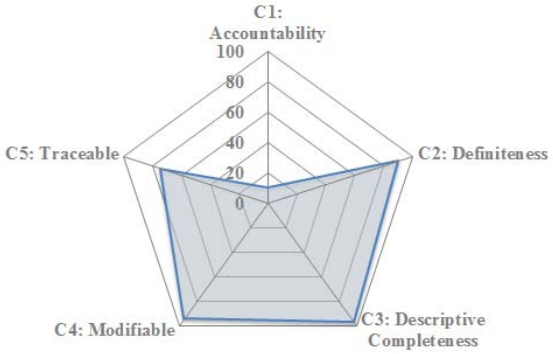


Fig. 4 Quality Characteristic View for SRS_A

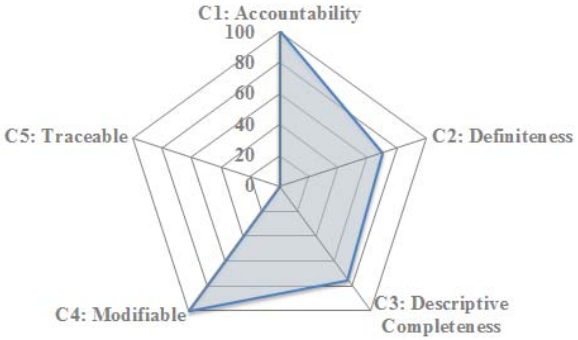


Fig. 5 Quality Characteristic View for SRS_B

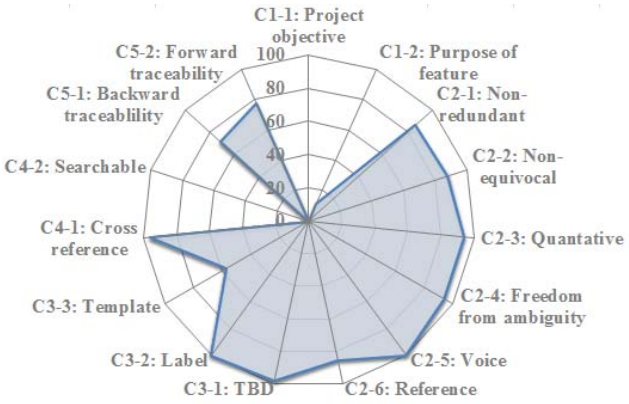


Fig. 6 Quality Sub-characteristic View for SRS_A

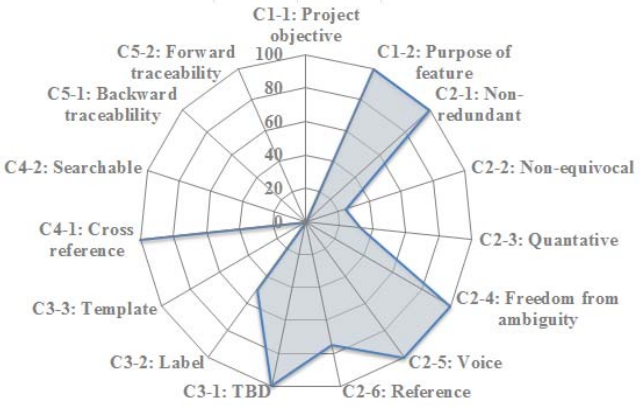


Fig. 7 Quality Sub-characteristic View for SRS_B

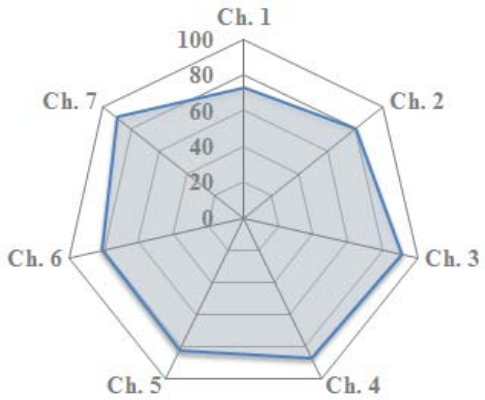


Fig. 8 Chapter View (Project SRS) for SRS_A

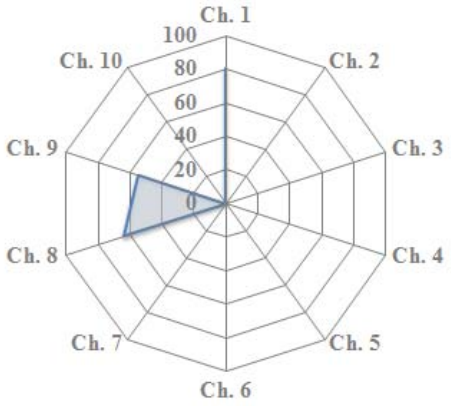


Fig. 9 Chapter View (Project SRS) for SRS_B

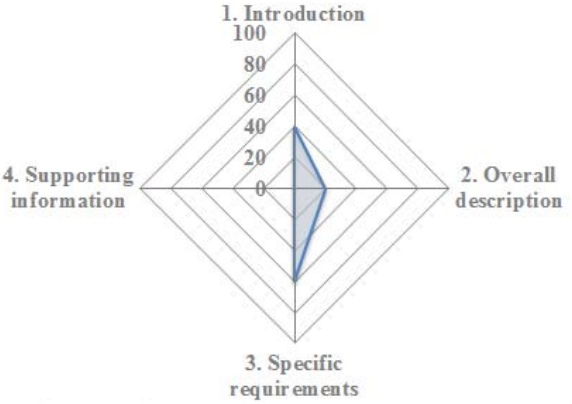


Fig. 10 Chapter View (Reference SRS) for SRS_A

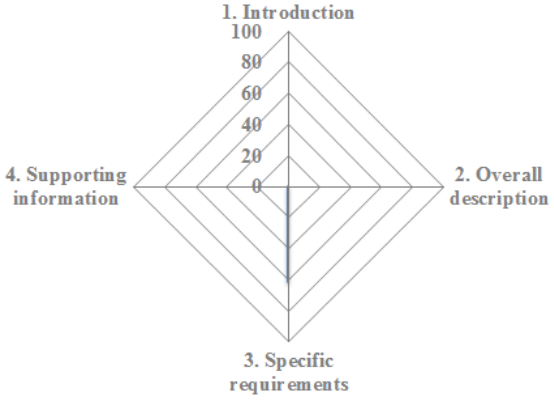


Fig. 11 Chapter View (Reference SRS) for SRS_B