# A Two-Stage Inspection Method for Automotive Software Systems and Its Practical Applications

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* - Requirements to automotive software is ever increasing its size and complexity due to rapid advancement of internet connection and advanced driver assistance system (ADAS). To assure the quality of software requirements specification (SRS) is a challenge. The quality assurance of SRS mostly relies on peer-review, which is error prone. To improve the quality of SRS, numerous techniques have been proposed, including a variety of review and inspection techniques. However, previous works indicate the diversity and confusion of requirements at different abstraction levels in automotive SRS makes it difficult to apply conventional review techniques.**

**We propose a two-stage inspection method in order to separate perspectives in inspection. We define the reference SRS and the associated document quality characteristics, and mapping method from the reference SRS to multiple domain SRSs. We also develop a method to quantitatively measure the document quality of an SRS by third-party inspection.**

**The contributions of this research include: 1) proposal of two-stage inspection process, 2) method of mapping the reference automotive SRS to project SRSs in multiple different product domains, 3) method for quantitative measurement and benchmark of SRS quality, 4) evaluation of the effectiveness of the method through applications to five actual automotive SRSs in four different product domains, and 5) adequacy validation of the IEEE Std. 830-1998 SRS template for automotive SRSs.**

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

## I. INTRODUCTION

Software is adding its weight to automobiles, becoming 10 times larger and more complex every ten years [3], [6], [10]. The introduction of Requirements Engineering (RE) is a key success factor to the development of such a complex product within the predefined costs and scheduling [8]. Automotive software development organizations conduct SRS inspection as Automotive SPICE requires [22]. On the contrary, an industrial survey revealed that requirements inspection practices are still rather ad hoc although many projects and organizations define the rules and procedures for the requirements inspection [8].

Since the automotive industry has traditionally been organized in a highly vertical manner [18], product development involves many different organizations, e.g. companies or divisions. Therefore, software requirements specification (SRS) plays a particularly important role as a means to ensure that the different organizations have a shared understanding of development goals. Active research, e.g. ([13], [17]) has been made in the automotive industry to improve the inspection process and the quality of SRSs.

There are two factors that prevent proper inspection in the automotive industry. The first factor is that proper SRS structure and quality characteristics for automotive software systems have not been established. The second factor is the diversity in automotive software. Pretschner et al. clustered automotive software into five product domains [18]. Different kinds of knowledge and technique are required to develop automotive software in each product domain. Therefore, software development is carried out in separated business units in general. For example, DENSO develops products in five business units. Development projects in different business units consequently need to define their own SRS and quality measurement method [18]. We found that the differences are becoming an impediment to systematic inspection [21]. The performance of an inspection, as a result, largely depends upon the inspector's experience and domain knowledge. However, domain experts who possess such credential is sparse resource and it is difficult to assign them for all inspections.

### A. Research Objective

A unified inspection method ought to be used among all software development units regardless of the product domains. As a previous work, a third-party inspection method is proposed for enterprise information systems [20]. However, automotive SRSs have different constructs in different product domains, reflecting the specific characteristics in each product domain [21]. Therefore, our research objective is twofold: to extend the capability of third-party inspection process to comply with automotive software systems, and to develop an SRS translation method that can accommodate the differences

2332-6441/16 $31.00 © 2016 IEEE
DOI 10.1109/RE.2016.55



IEEE computer society

RE 2016, Beijing, China
Industry Paper

between product domains. These research objectives are based on the needs of the case company DENSO.

### B. Contribution

We made the following contributions to improve inspection of requirements for automotive software systems:

(1) Proposal of a two-stage inspection method, which can be uniformly applied in all business units

(2) Method of mapping a reference automotive SRS to SRSs in multiple different product domains

(3) Method for quantitative measurement and benchmark of SRS quality

(4) Evaluation of the effectiveness of the method through applications to five actual automotive SRSs in four different product domains

(5) Adequacy validation of the IEEE Std. 830-1998 SRS template for automotive SRSs

## II. Problem Context and Research Questions

### A. Problem Context

The study focuses on automotive software systems and was executed at DENSO, a major supplier of automotive parts. Suppliers accept orders from automakers, which are called OEM (Original Equipment Manufacturer), for an ECU (Electronic Control Unit) or a system composed of multiple ECUs. OEMs integrate and test the components delivered by multiple suppliers to develop a complete automobile [11].

DENSO receives requirements specifications for a complete product and then creates internal specifications, including SRS, and hardware requirements specification (HRS), prior to product development. This study targets SRS from among these specification documents. Software is developed either on the premises of DENSO or at partner companies, including group companies.

Most SRSs are written in natural languages. Some parts where natural languages leave ambiguity are supplemented with diagrams including UML, DFD, and decision tables. Simulink models are also used for describing some requirements in control systems such as powertrains. The most commonly used languages are Japanese and English.

### B. Research Questions

This paper answers the following research questions through an industrial case study at DENSO:

RQ1 Is the third-party inspection suggested in this paper effective at improving the quality of SRSs?

RQ2 Is IEEE Std. 830-1998 SRS template adequate for automotive software system requirements?

RQ2.1 Is IEEE Std. 830-1998 SRS template excessive for automotive software system requirements?

RQ2.2 Is IEEE Std. 830-1998 SRS template sufficient for automotive software system requirements?

## III. Related Works

### A. SRS Structure

IEEE Std. 830-1998 [11] is an international standard for software requirements specifications. Other standards include Software Requirements Specification (DI-IPSC-81433A) [5] by the U.S. Department of Defense and NASA Software Documentation Standard (NASA-Std-2100-91) [16] by NASA. Other than standards, Robertson and Robertson proposed Volere Requirements Specification Template [19], which is based on their extensive experience consulting for software development companies. Wiegers and Beatty also proposed another SRS template in their book [24]. As far as we know, however, any prior research has evaluated the adequacy of these templates for automotive software system requirements.

### B. SRS Quality Characteristics and Measurement Methods

As for SRS quality characteristics, eight quality characteristics defined in IEEE Std. 830-1998 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 [4]. 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.

### C. Inspection Methods

Since Fagan proposed the initial inspection method, Fagan inspection, in 1976 [7], various inspection methods have been proposed in order to improve Fagan inspection [2].

#### 1) N-fold Inspection

It is reported that inspection by multiple small teams (*N*-fold inspection) can detect more defects than inspection by one big team [15]. Another study, however, reported that the performance of *N*-fold inspection depends on the level of expertise of the team members and the number of teams [12]. Therefore, the method is not good enough to employ as a corporate-wide inspection method beyond product domains.

#### 2) Phased Inspection

Phased inspection [14] focuses on a particular characteristic, e.g. portability, reusability, or maintainability, in each phase and does not separate the phases based on the need for domain knowledge. Moreover, the effectiveness of the method was evaluated only for source codes and not confirmed for requirements.

#### 3) RISDM (Requirements Inspection System Design Method)

RISDM is proposed to design inspection methods in which the quality of an SRS can be quantitatively measured and thus compared to that of another SRS [20]. RISDM's scope of the

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, i.e. modifying existing features and adding new features to, the previous product's SRS in the automotive industry. This makes it quite difficult to replace the SRS template. Our previous work [21] revealed that RISDM comes short of ability to inspect automotive SRSs that have different constructs in different product domains. Therefore, an inspection method that can accommodate these differences is necessary.

## IV. Framework of Two-stage Inspection Method

### A. *Two-stage Inspection Process*

In the two-stage inspection method, inspection is conducted in accordance with the process shown in Fig. 1.

(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) Third-party inspection: The third-party inspector, independent from the project, conducts SRS inspection.
(3) Judgment on the third-party inspection result:
  3.a) The inspector forwards the SRS to the project team if the inspection result meets the acceptance criteria.
  3.b) 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 meet the acceptance criteria.
(4) Receipt of SRS by the project: The project team receives the SRS that the third-party inspector judged to meet the acceptance criteria.
(5) Project inspection: A project manager, developers, testers, and some other roles that constitute the project team inspect the SRS utilizing their domain knowledge.
(6) Judgment on the project inspection result:
  6.a) The project establishes a requirements baseline and begins software development if the inspection result meets the acceptance criteria.
  6.b) The project returns the SRS to the requirements analyst with an analysis report that lists the findings from the inspection.
(7) Make improvement:
  7.a) 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.
  7.b) When the requirements analyst receives an analysis report from the project, the analyst answers the findings and makes an improvement on the SRS if he or she agrees with the findings.

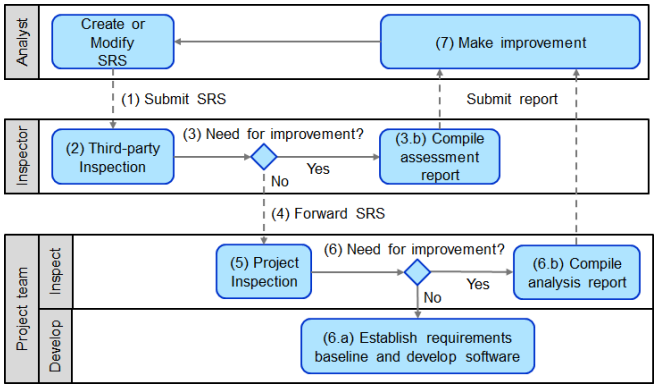



Fig. 1. Two-stage Inspection Process

### B. *Third-party Inspection Meta-model*

Figure 2 illustrates the third-party inspection meta-model. The reference SRS defining the structure of SRS and the document quality characteristics are necessary for quantitative measurements of SRS document quality. The reference SRS consists of an SRS template that defines elements needed to be included in the SRS and their organization. The document quality characteristics are defined as part of the SRS quality characteristics, which third-party inspectors who do not have domain knowledge can measure. The requirements quality characteristics, on the other hand, are the remainder of the SRS quality characteristics after deducting the document quality characteristics.

The inspection guideline consists of the inspection matrix and the question set. The guideline supports third-party inspectors who inspect a project SRS. An inspection matrix associates the table of contents of the reference SRS to document quality characteristics to be measured. The question set is a practical means to measure the document quality characteristics.

The translation matrix associates the contents of the reference SRS to those of a project SRS. The mapping provided by the translation matrix enables a uniform inspection process applicable to diverse automotive SRSs.

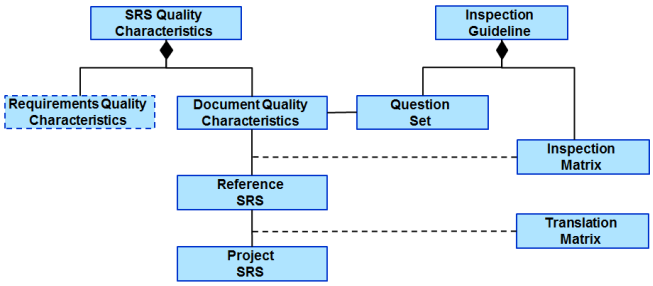



Fig. 2. Third-party Inspection Meta-model

## V. Details of Third-party Inspection Method

### A. Reference SRS

Any reference SRSs that describe the requirements for automotive software systems have not been proposed [23]. We adopt the IEEE Std. 830-1998 SRS template, one of the most commonly used SRS templates, as a tentative reference SRS. We verify how well the IEEE Std. 830-1998 SRS template fits automotive software system requirements in the course of our trial of the proposed method.

### B. SRS Quality Characteristics

Since we adopted IEEE Std. 830-1998 SRS template as a reference SRS, we also consult the SRS quality characteristics defined in IEEE Std. 830-1998 to define our SRS quality characteristics. We removed *ranked for importance and/or stability* from the eight original quality characteristics defined in IEEE Std. 830-1998, i.e. *correct, unambiguous, complete, consistent, ranked for importance and/or stability, verifiable, modifiable,* and *traceable* and added a new quality characteristic *achievable* instead, and defined the new set of the eight quality characteristics as the SRS quality characteristics. We explain the rationales for removing *ranked for importance and/or stability* and adding *achievable* in the next section.

### C. Document and Requirement Quality Characteristics

To conduct the two-stage inspection consisting of the third-party inspection and the project inspection, SRS quality characteristics have to be categorized in terms of the need for domain knowledge, and to be allocated to either inspection stage. The parts of SRS quality characteristics dependent on domain knowledge are allocated to the third-party inspection stage as document quality characteristics. Similarly, the parts of SRS quality characteristics independent of the domain are allocated to the project inspection stage as requirement quality characteristics. When an SRS quality characteristic cannot be simply allocated to either inspection stage, the quality characteristic is decomposed, or integrated into another quality characteristic. The newly derived quality characteristics are then allocated to either stage. The following sections explain how each SRS quality characteristic leads to new quality characteristics and allocation to an inspection stage. The relations between SRS quality characteristic and newly defined quality characteristics are illustrated in Fig.3.

#### 1) Correct

An SRS is *correct* if, and only if, every requirement stated therein is one that the software shall meet [11]. 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 project inspection by the project team. The third-party inspector measures the new quality characteristics during the third-party 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*.

#### 2) Unambiguous

An SRS is *unambiguous* if, and only if, every requirement stated therein has only one interpretation [11]. 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 applying predefined checklists, e.g. a list of ambiguous terms.

#### 3) Complete

An SRS is *complete* if, and only if, it includes the following elements [11]:

a) All significant requirements, whether relating to functionality, performance, design constraints, attributes, or external interfaces
b) Definition of the responses of the software to all realizable classes of input data in all realizable situations
c) 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 3 perspectives regarding completeness. The first and second perspectives relate to the completeness of requirements. On the other hand, the third perspective relates to the completeness of the description. Third-party inspector can evaluate only the completeness of

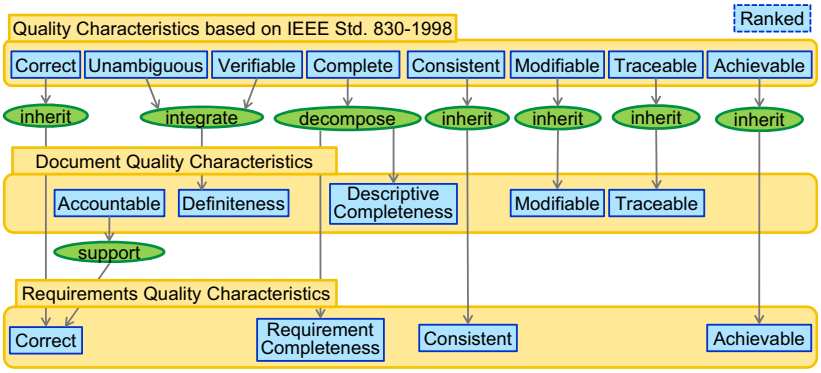



Fig. 3. Allocation of SRS Quality Characteristics

descriptions. It is reasonable to assume that the completeness of requirements, as with correctness, can be evaluated during the project inspection if *accountability* is satisfied. Thus, we define two new quality characteristics, namely: *descriptive completeness* and *requirement completeness*.

*4) Consistent*

An SRS is internally *consistent* if, and only if, no subset of individual requirements described in it conflict [11]. 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 project inspection stage.

*5) 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 [11]. However, the initial scope of requirements is seldom shrunk or changed for automotive software development. An industrial survey [9] shows that there are few unnecessary or optional needs, and these needs are generally not prioritized in automotive, aerospace, and energy domains. Thus, we eliminated this quality characteristic from the target of measurement.

*6) 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* [11]. 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 not have ambiguity. Consequently, we combine *unambiguous* and *verifiable* and derive a new quality characteristic: *Definiteness*.

*7) 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 [11]. Because the property of *modifiable* can be measured by applying some predefined checklists, we do not derive new quality characteristics from this quality characteristic.

*8) 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 [11]. Because *traceable* can be measured by applying some predefined checklists, we do not derive new quality characteristics from this quality characteristic.

*9) 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 define new quality characteristics from this quality characteristic. We simply assign this quality characteristic to the project inspection stage.

### D. Question Set

We define closed questions, which can be objectively answered with either yes or no by third-party inspectors who do not have domain knowledge, as a practical means to measure document quality characteristics. Document quality characteristics are too abstract to be associated with closed questions. Thus, we decomposed them into sub-quality characteristics that are concrete enough to be associated with closed questions. Fifteen questions comprising the question set and corresponding to each quality sub-characteristic are shown in TABLE I.

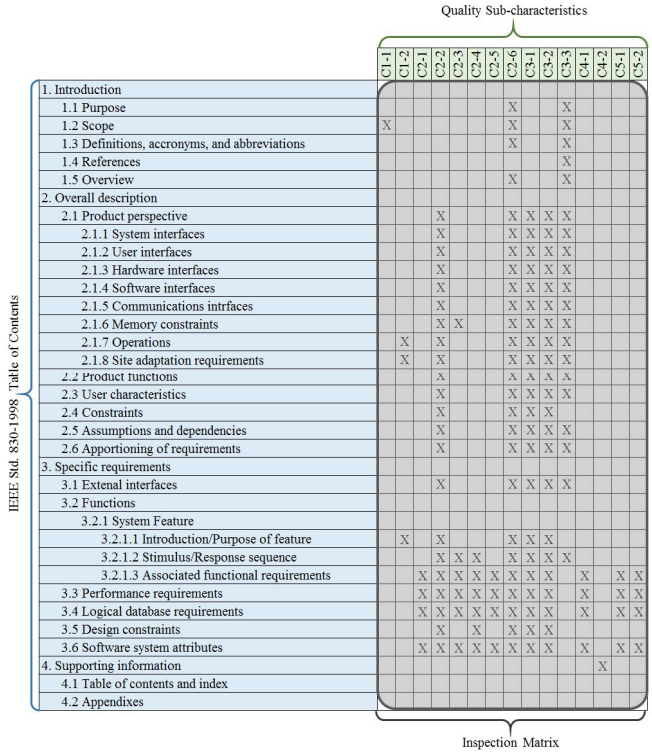
### E. Inspection Matrix

Because not all quality sub-characteristics are needed to be measured for the entire reference SRS, we define an

TABLE I. QUESTION SET

| ID | Quality Characteristics | 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 without defining its meanings? |
| C2-3 | | Quantitative | Qualitative expression is not used where quantitative 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 referred, 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 appropriate requirement when a requirement refer to another requirement? |
| C4-2 | | Searchable | Table of contents and index are created? |
| C5-1 | Traceable | Backward traceability | Traceability matrix that specifies the link between high-level requirements and software requirements? |
| C5-2 | | Forward traceability | Each requirement has a unique identification? |

TABLE II. INSPECTION MATRIX




inspection matrix (TABLE II) that associates the table of contents of the reference SRS to quality sub-characteristics to be measured. The cells marked with 'X' specify the inspection points. Taking '1.1 Purpose and Scope' as an example, the matrix tells you that quality sub-characteristics C2-6 and C3-3 need to be measured in this section. In our case, the total number of inspection points became 149 points. To quantify document quality, we adopt a simple grading method. 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.

### F. *Translation Matrix*

We define a reference SRS in order to absorb the differences in the constructions of project SRSs and to inspect different project SRSs in a uniform process. The relation between the reference SRS and a project SRS is defined as a translation matrix (TABLE III). The translation matrix associates the contents of the reference SRS to those of a project SRS. The cells marked with 'X' specify the translation points. For example, you can understand from the matrix that '2.4 Constraints' of the reference SRS can be mapped to '1.4 Hardware constraints' of a project SRS. An actual translation matrix created for SRS_D is shown in TABLE V.

### G. *Inspection Guideline*

We call the paring of the inspection matrix and the question set the inspection guideline. This inspection guideline and aforementioned translation matrix help the third-party inspectors who do not have domain knowledge uniquely identify which quality characteristics to measure for each SRS part. Moreover, the identified quality characteristics can be objectively measured by answering the closed questions.

## VI. PRACTICAL APPLICATION TO AUTOMOTIVE SRSS

We evaluated five actual DENSO products' SRSs from four different product domains. Each product has multiple specification documents, e.g. SRS and HRS. The subject of this research is SRS. However, the names of specification documents differ between development projects. Thus, the evaluator selected the specification documents that are most appropriate to be considered SRSs after examining the contents and interviewing a project team about the position of the documents in their development process. For system products composed of multiple ECUs, an SRS is created for each ECU. Therefore, the evaluator picked one representative ECU and inspected its SRS. Moreover, some ECUs may have multiple separate SRSs. The purpose of separating an SRS is to manage a major characteristic of automotive software, i.e. various customization caused by the differences in grades and destinations of vehicles. For separated SRSs, we evaluate a collection of the specifications as a single SRS.

TABLE IV lists the product domains and the page counts of five SRSs we evaluated in this study. All SRSs are written in Japanese.

TABLE III. TRANSLATION MATRIX

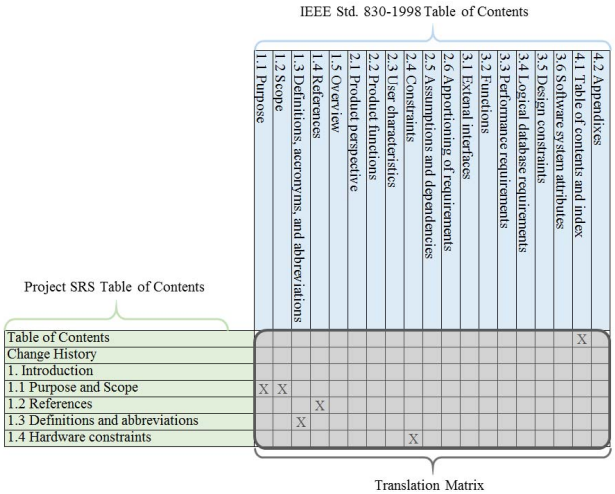


TABLE IV. SRS DATA SET

| ID | Product Domain | # of Pages |
|---|---|---|
| SRS_A | Infrastructure | 76 |
| SRS_B | Body | 34 |
| SRS_C | Body | 39 |
| SRS_D | Safety | 41 |
| SRS_E | Comfort | 53 |
| **Total** | | 243 |

## VII. EVALUATION AND DISCUSSION

### A. *Verifying the Effectiveness of Proposed Method (RQ1)*

We verify whether the quality score information is effective for SRS quality improvement. We analyzed the practical application results for SRS_E to do the verification. The quality score has two categories: non-normalized score and normalized score. The non-normalized score is calculated for a project SRS as is. On the other hand, the normalized score is calculated for the reference SRS after associating a project SRS to it with the aid of a translation matrix. Each quality score category has two different views: quality characteristic view, and structural view.

*1) Non-normalized Score*

If we analyze the quality score from quality characteristic view (Fig.6) and sub-characteristic view (Fig. 8), we can understand which characteristics need to be improved in the SRS. For SRS_E, we can learn that accountability needs to be improved. Next, if we analyze the quality score from the structural view (Fig. 10), we learn that SRS_E has quite a high score overall — only chapter 1 falls below 80% — but there still exists a margin for improvement.

*2) Normalized Score*

From the normalized score, we can grasp the missing elements in a project SRS. We can understand from Fig. 11 that SRS_E lacks the elements corresponding to chapter 1, 2, and 4 in the reference SRS. Another benefit we can obtain from the normalized score is that we can relatively compare quality scores between project SRSs that have different constructions.

*3) Relative Comparison*

The normalized score enables relative comparison between different project SRSs. Fig.12 and Fig.13 show relative comparison results for the five SRSs listed in TABLE IV in the quality characteristic view and the structural view, respectively. We can assume that we would be able to use the quality score benchmark to judge whether the development phase could move from the requirement phase to the next phase if we keep accumulating more quality score data.

Through the above discussion, we confirmed that quality score information gathered from third-party inspection provides useful insight for improving the quality of SRS because the information offers multifaceted perspectives on SRS quality.

### B. *Validating the Reference SRS (RQ2)*

*1) Verifying the Excessive Elements (RQ2.1)*

We conducted a document histogram analysis to verify the existence of excessive elements of the IEEE Std. 830-1998 SRS template for describing automotive SRSs. When we evaluated the five SRSs, we created a translation matrix for each SRS. Thus, we counted the number of times 'X' occurred per section of IEEE Std. 830-1998 and made a histogram (Fig.4).

If the frequency of a section is less than five, it means that at least one SRS does not have any content related to the section because the number of SRSs is five. We can learn that Chapter 1 *Introduction* and Chapter 2 *Overall description* have low frequencies and Chapter 3 *Specific requirements* has high frequency.

One possible reason why Chapter 1 has low frequency is that DENSO is not an OEM but a supplier. When a supplier accepts an ECU product, some information that should be described in Chapter 1, e.g. the purpose of a system, is kept only inside the OEM and is not distributed to the supplier. However, we cannot say with certainty that these elements are excessive for automotive SRSs since low frequency does not necessarily mean low importance.

It can be assumed that low frequency in Chapter 2 is attributed to the characteristics of automotive software and its development process. First, we consider what makes the frequency of 2.3 *User characteristics* so low. Automotive software that directly interfaces with the user is limited to certain products such as car navigation. None of the five SRSs in the data set has direct inputs from the user. Next, we look at 2.5 *Assumptions and dependencies*, and 2.6 *Apportioning of requirements*. Automotive software is incrementally developed. In other words, mass-production software is completed through multiple stages of prototyping. During the development process, updated requirements are handed over from an OEM to the suppliers as a change request [13]. These

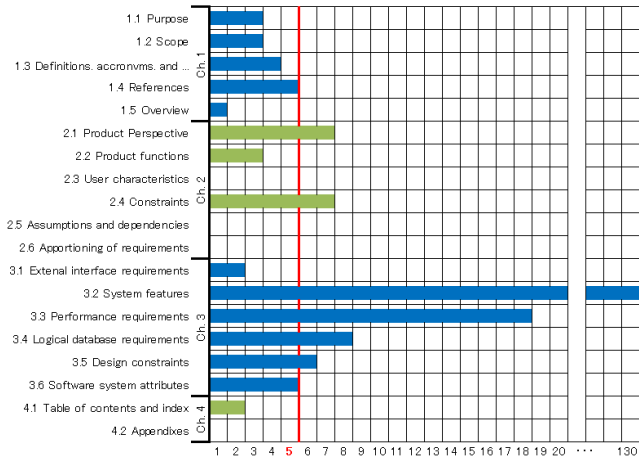


Fig. 4. Document Histogram

elements are not used because addition and modification of requirements are taken for granted in this incremental process. It could be also considered another reason that fluctuation of requirements in automotive software development, at least for now, does not occur as often as in enterprise software development.

From the document histogram analysis, we found *User characteristics*, *Assumptions and dependencies,* and *Apportioning of requirements* could be excessive elements for automotive SRSs. These elements may have to represent similar but different aspects to fit automotive software requirements more precisely. For example, user characteristics could be replaced with environmental characteristics in order to generalize characteristics of the interface counterparts.

*2) Verifying Insufficient Elements (RQ2.2)*

We inspected five project SRSs with IEEE Std. 830-1998 as the reference SRS. We identified two important attributes of automotive SRSs that IEEE Std. 830-1998 falls short of managing, namely: diverse variant requirements, and confusion of requirements at different abstraction levels.

Automotive systems have a significant number of variations to differentiate vehicle grades. A premium car typically has about 80 electronic fittings that can be ordered depending on the country, and so on [18]. Because of this variation, recent automotive software system development is shifting towards software product line engineering (SPLE). However, commonality and variability, which are vitally important for SPLE, are not considered in IEEE Std. 830-1998.

Automotive software has different abstraction levels as exemplified by AUTOSAR layered software architecture [1] shown in Fig.5. Therefore, requirement analysts need to avoid including design constraints in the disguise of requirements to manage requirements at a high abstraction level [4], [11]. On the contrary, requirement analysts need to step into design and manage requirements at a low abstraction level [23]. From the above reasons, an automotive SRS needs to manage requirements written at very different levels of abstraction. However, the SRS template and guideline provided by IEEE Std. 830-1998 is not sufficient to capture requirements at various levels of abstraction.

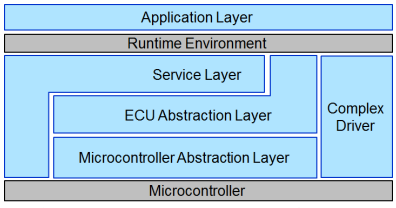


Fig. 5. AUTOSAR Software Architecture

## C. Discussion

*1) Validity of Two-stage Inspection*

Third-party inspectors who do not have domain knowledge measure all quality characteristics with the inspection method proposed by a previous research [20] for IT systems. On the contrary, automotive SRSs contain domain specific requirements and consequently the domain knowledge is a prerequisite for inspection. The inspection method for IT systems cannot be directly applied to automotive SRSs. Therefore, we proposed a two-stage inspection method, in which the inspection process is separated into a domain independent stage and a dependent stage, and these stages are sequentially executed. We defined the detail of a third-party inspection for the first inspection stage and evaluated its effectiveness through practical applications. We confirmed that automotive SRSs for different product domains can be uniformly inspected by a third-party inspector, and quality score information provides useful insight for improving the quality of SRS. Domain knowledge is a prerequisite not only for automotive software but all embedded software. The two-stage inspection method we proposed is applicable to embedded software development in general.

*2) Characteristics of Automotive Software Requirements*

We analyzed the description density of five actual SRSs for each specification items from the document histogram and identified that certain items of the IEEE Std. 830-1998 SRS template could be unsuitable for automotive SRSs. We also identified two important attributes of automotive SRSs that the IEEE Std. 830-1998 SRS template is not sufficient to manage, namely diverse variant requirements, and confusion of requirements at different abstraction levels. No research has proposed optimized SRSs for automotive software. Our findings can be a baseline for the future research in this field.

# VIII. Conclusions and Future Work

We proposed a two-stage inspection method that consists of the third-party inspection stage and the project inspection stage. We defined the details of the third-party inspection stage and evaluated its effectiveness.

This work is based on the previous work in a third-party inspection method for IT systems [20], and our previous work appeared in [21]. This work contains the following new results: 1) method of mapping the reference automotive SRS to project SRSs in multiple different product domains, 2) evaluation of effectiveness of the proposed method through the practical application with five actual SRSs from four different domains, and 3) identification of shortcomings of the IEEE Std. 830-1998 SRS template to accommodate two important aspects of automotive SRSs from the practical applications: diverse variant requirements, and confusion of requirements at different abstraction levels.

For future work, we plan to collect more SRSs from widespread product domains and apply the proposed method to them in order to improve inspection accuracy, and to accumulate data for the SRS quality benchmark. We also plan to evaluate the efficiency of the proposed third-party inspection method, and the effects to the project inspection by

domain experts, in order to clarify when the two-stage inspection method should be employed, and when should not.

We defined the details only for the third-party inspection stage this time. We will concretize the project inspection stage and verify the effectiveness of the proposed method as a whole.

## References


[1] AUTOSAR Consortium, AUTOSAR – Layered Software Architecture. AUTOSAR Consortium, Tech. Rep. Release 4.2.2, Jul. 2015.

[2] A. Aurum, H. Petersson, and C. Wohlin, State-of-the-Art: Software Inspections after 25 Years, J. of Software Testing Verification and Reliability, Vol. 12, No. 3, Sep. 2002, pp. 133-154.

[3] P. Braun, et al., 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.

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

[5] DI-IPSC-81433A, Data Item Description: Software Requirements Specification, Dec. 1999.

[6] C. Ebert and C. Jones, Embedded Software: Facts, Figures, and Future, IEEE Computer, Vol. 42, No. 4, Apr. 2009, pp. 42-52.

[7] M. Fagan, Design and Code Inspections to Reduce Errors in Program Development, IBM Systems Journal, Vol. 15. No. 3, 1976, pp. 182-211.

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

[9] G. Fanmuy et al., A Survey on Industrial Practices in Requirements Engineering, INCOSE Int'l Symposium, Vol. 22, No. 1, Jul. 2012, pp. 1021-1040.

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

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

[12] E. Kantorowitz, et al., The Performance of the N-Fold Requirement Inspection Method, J. of Req. Eng., Vol. 2, No. 3, 1997, pp. 152-164.

[13] J. Krisch, and F. Houdek, The Myth of Bad Passive Voice and Weak Words. Proc. of RE 2015, IEEE, Aug. 2015, pp. 344–351.

[14] J. C. Knight, and E. A. Myers, An Improved Inspection Technique, CACM, Vol. 36, No. 11, Nov. 1993, pp. 51-61.

[15] J. Martin, et al., N-Fold Inspection: A Requirements Analysis Technique, CACM, Vol. 33, No. 2, Feb. 1990, pp. 225-232.

[16] NASA-STD-2100-91, NASA Software Documentation Standard, NASA Headquarters Software Engineering Program, Jul. 1991.

[17] D. Ott, Defects in Natural Languages Requirements Specifications at Mercedes Benz, Proc. of RE 2012, IEEE, Sep 2012, pp. 291-296.

[18] A. Pretschner, et al., Software Engineering for Automotive Systems: A Roadmap, Proc. of FOSE '07 2007 Future of Software Engineering, IEEE, May 2007, pp. 55-71.

[19] J. Robertson, and S. Robertson, Volere Requirements Specification Template Edition 17, Atlantic Systems Guild, 2014.

[20] S. Saito, et al., RISDM: A Requirements Inspection Systems Design Methodology, Proc. of RE 2014, IEEE, Aug. 2014, pp. 223-232.

[21] A. Takoshima, and M. Aoyama, Assessing the Quality of Software Requirements Specifications for Automotive Software Systems, Proc.of APSEC 2015, IEEE CPS, Dec. 2015, pp. 393-400.

[22] VDA QMC Working Group 13/ Automotive SIG, Automotive SPICE Process Assessment Model V. 3.0, Jul. 2015.

[23] 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.

[24] K. Wiegers, et al., Software Requirements, 3rd Ed., Microsoft Press, 2013.


TABLE V. TRANSLATION MATRIX FOR SRS_D

| | | | | 1.1 Purpose | 1.2 Scope | 1.3 Definitions, acronyms, and abbreviations | 1.4 References | 1.5 Overview | 2.1 Product Perspective | 2.1.1 System interfaces | 2.1.2 User interfaces | 2.1.3 Hardware interfaces | 2.1.4 Software interfaces | 2.1.5 Communications intfaces | 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 External interfaces | 3.2.1.1 Introduction/Purpose of feature | 3.2.1.2 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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1.2.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1.2.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1.2.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1.3.1 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1.3.2 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 1.3.3 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1.4 | | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 1.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 2.1 | | | | X | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 2.1.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 2.1.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 2.1.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 2.2 | | | | | | | | | | | | | | | | | X | | | | | | | | | | | | | | |
| | 2.3 | | | | | | | | | X | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 3.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 3.1.1 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.1.2 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.1.3 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.1.4 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.1.5 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | 3.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 3.2.1 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.2.2 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.2.3 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | | 3.2.4 | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| | 3.3 | | | | | | | | | | | | | | | | | | | | | | | X | X | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 4.1.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | 4.1.1.1 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | | 4.1.1.2 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | | 4.1.1.3 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | | 4.1.1.4 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.2 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.3 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.4 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.5 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.6 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.7 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| | | 4.1.8 | | | | | | | | | | | | | | | | | | | | | | | | | X | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 5.1.1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| | | 5.1.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| | | 5.1.3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| | | 5.1.4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| | | 5.1.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | X | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 6.1.1 | | | | | | | | | | | | | | | | | | | | | | X | | X | | | | | | |
| | | 6.1.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | 7.1.1 | | | | | | | | | | | | | | | | | | | | | | X | | X | | | | | | |
| | | 7.1.2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

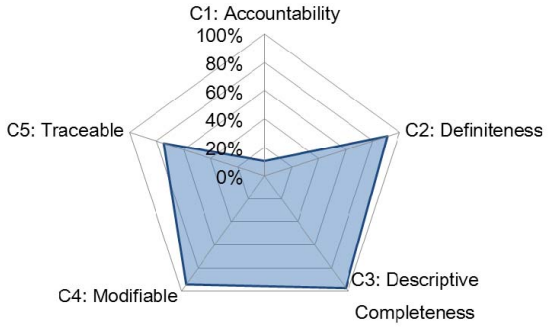



Fig. 6. Non-normalized Quality Characteristic View for SRS_E

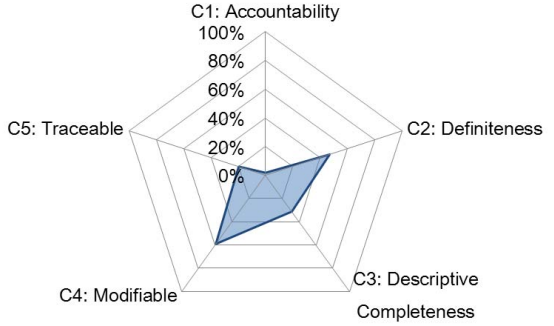



Fig. 7. Normalized Quality Characteristic View for SRS_E

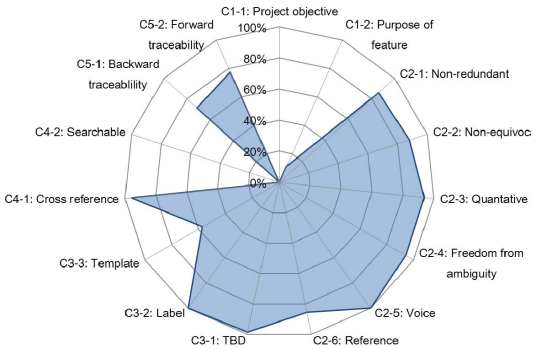



Fig. 8. Non-normalized Quality Sub-characteristic View for SRS_E

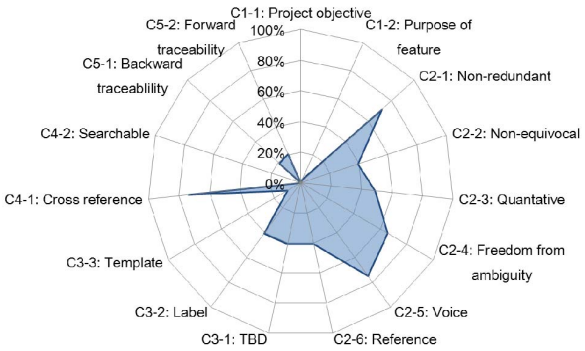



Fig. 9. Normalized Quality Sub-characteristic View for SRS_E

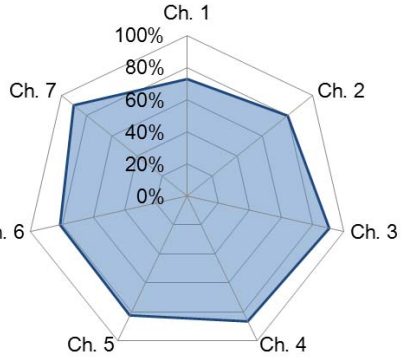



Fig. 10. Non-normalized Structural View for SRS_E

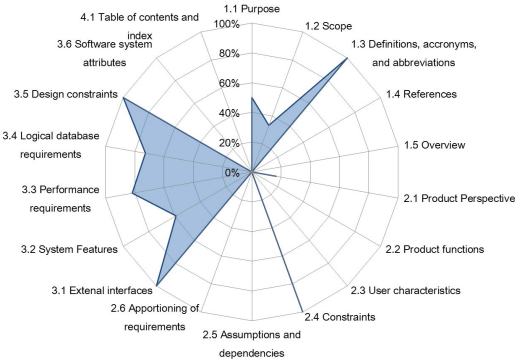



Fig. 11. Normalized Structural View for SRS_E

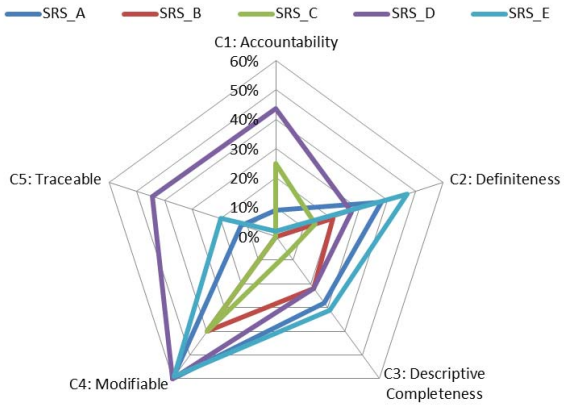



Fig. 12. Relative Comparison in Quality Characteristic View

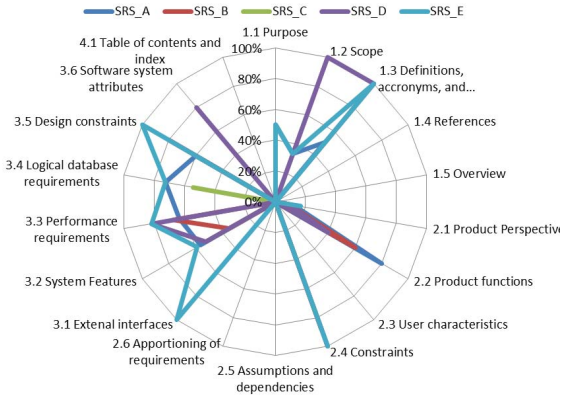



Fig. 13. Relative Comparison in Structural View