**Proceedings of the ASME 2024**
**International Design Engineering Technical Conferences & Computers and Information in Engineering Conference**
**IDETC-CIE 2024**
**August 25–28, 2024, Washington, DC, USA**

**IDETC/CIE2024-139583**

# WELL-FORMED QUALITY OF SYSTEM REQUIREMENTS FOR VERIFYING TO ISO 29148-2018: A NATURAL LANGUAGE PROCESSING (NLP) BASED FRAMEWORK AND QUANTITATIVE METRIC

**Chandan Kumar Sahu[1,*], Vedant Rai[2], Vinit Roshan[3]**

[1]Department of Automotive Engineering, Clemson University, Greenville, SC
[2]Artificial Intelligence Research Institute in Science and Engineering (AIRISE), Clemson University, Greenville, SC
[3]Production System, Schlumberger, Houston, TX

## ABSTRACT

*The system requirements directly affect the quality, cost, and performance of system architecture developed in the detailed design phase, necessitating a comprehensive approach to ensure accuracy, completeness, and clarity. A well-defined set of requirements lays the foundation for developing effective and efficient systems, guiding the architectural design process toward meeting stakeholder expectations while optimizing resource utilization and minimizing potential risks associated with ambiguous or poorly specified requirements. Methodologies for assessing the quality of individual requirements generated during the initial conceptual system design phase are crucial. In this paper, we outline a rule-based natural language processing (NLP) framework for assessing the well-formed quality of individual requirements that conforms to the ISO/IEC/IEEE 29148:2018 (E) requirement standard. We leverage advanced linguistic analysis to automatically evaluate each of the nine essential characteristics —necessary, appropriate, unambiguous, complete, singular, feasible, verifiable, correct, and conforming — prescribed by ISO/IEC/IEEE 29148:2018 (E) to ensure the clarity, completeness, and consistency of individual requirements. The framework assesses requirements at various levels, including token, span, syntactic structure, and sentence levels, to determine compliance with the defined characteristics of well-formed requirements. Furthermore, this work introduces a quantitative metric, the well-formed requirement quality index, which amalgamates these nine characteristics into a scalar value within the range of [0, 1], thereby facilitating a comprehensive assessment of requirement quality. The efficacy of this framework is demonstrated through the evaluation of select requirements samples, offering a means to assess requirement quality and perform quantitative comparisons of individual requirements' qualities.*

**Keywords: System Requirements, Quality of requirements, ISO 29148, INCOSE, Well-Formed Requirement Quality Index**

*Corresponding author: csahu@clemson.edu

## 1. INTRODUCTION

Systems are developed to satisfy a set of requirements. The requirements are statements that translate or express the stakeholders' *needs* and their associated constraints and conditions [1]. The conditions are measurable attributes of a requirement indicating a specific circumstance under which the requirement applies to the system of interest. The requirements serve as a formal agreement between the stakeholders (*e.g.*, acquirers, suppliers, developers, customers, users and operators) for consistent development of the system. These requirements keep evolving from the ideation to the materialization of the system. Poor management of requirements risks system failure, inadequate performance, delays in schedule and overruns in cost [2]. A 2023 report from the Office of Inspector General concerning NASA's "Top Management and Performance Challenges" underscored the agency's difficulties in overseeing its contractors, attributing the challenges primarily to inadequately defined requirements [3]. This situation led to all of NASA's contractors failing to adhere to the planned schedules, notwithstanding the agency disbursing payments that exceeded the originally agreed-upon fixed prices. The cost of addressing errors over the system's lifecycle increases exponentially, with more than 50% of the errors originating in the requirements phase [4, 5]. Furthermore, the literature identifies requirements as a unifying factor linking various causes of project failure, including scope creep, inadequate communication, and insufficient stakeholder involvement, to a singular source of concern [6]. Consequently, the attainment of high-quality requirements is imperative for the successful development of a system. Systems engineered based on inadequately elicited requirements are at risk of aspiring to meet unnecessary, non-verifiable, incomplete, ambiguous, and solution-specific requirements, thereby compromising the developmental integrity and potential success of the system.

The requirements are *formally transformed* from the needs of stakeholders or higher-level parent requirements, constituting an *agreed-to-obligation* [7, 8]. This obligates the system to perform specific functions under defined conditions and constraints, aiming to fulfill stakeholders' needs. The system, upon materialization, is verified with the specified requirements to evaluate if the designed system will meet the stakeholder's needs. The *formal transformation* is essentially a requirement analysis process that ensures that the system is developed for the right purpose, to realize the need. The formal transformation of requirements necessitates that they be necessary, appropriate, singular, correct, and conforming. Concurrently, the *agreed-to-obligation* between stakeholders and the system ensures consensus on the requirements, stipulating that they must be unambiguous, complete, feasible, and verifiable [7, 8]. In the absence of these characteristics, a requirement fails to embody a fair agreement between all parties. Therefore, a requirement is considered well-formed when it embodies these essential attributes, which fundamentally constitute the "requirements of requirements".

The stakeholders elicit requirements in natural language (NL) due to its ergonomic nature. However, the freedom of expression in NL is also accompanied by inherent ambiguity and an unstructured nature. The presence of synonyms, homonyms, and context-dependent nature make NL a double-edged sword to elicit requirements [8, 9]. Thus, the ISO/IEC/IEEE 29148:2018 [1] states that the requirements shall be expressed in a very *specific, precise and unambiguous* manner to communicate the intent of the needs. Furthermore, consistent language across all the requirements eases the interpretation of requirements and deployment of formal methods on requirements. In light of the comprehensive array of characteristics that define the "requirements of requirements," it becomes imperative to delve deeper into understanding their implications and operationalization within the context of system development. This exploration naturally gives rise to pivotal inquiries, underscoring the essence of our investigation. The key underlying research questions (RQ) boil down to the following:

- **RQ1**: How to assess the qualities of requirements?
- **RQ2**: How do you compare the qualities of a pair of requirements?

To address these research questions, our paper makes the following key contributions:

- We create a requirement quality framework to evaluate the qualities of requirements. Our framework holistically evaluates the requirements using 42 NLP rules focusing on the token-level, span-level, syntactic structure-level and sentence-level attributes of the requirement.
- We propose a quantitative metric, *well-formed requirement quality* (wRQ) index. The metric incorporates the results of the 42 rules and the nine characteristics. The metric results in a score between 0 to 1 to compare the qualities of requirements. To the best of our knowledge, we are the first to offer a quantitative metric to evaluate the qualities of requirements in accordance with ISO/IEC/IEEE 29148:2018 (E).

Our research falls within the domain of requirement analysis, with a particular emphasis on verifying the quality of requirements through the application of Natural Language Processing (NLP) techniques. This approach is distinct from requirement validation, which assesses whether a requirement effectively communicates the intent of its need or parent requirement. In our work, we focus on verifying the well-formed nature of individual requirements by examining whether they exhibit the essential characteristics of a well-formed requirement, through a comprehensive evaluation based on a predefined set of rules.

This manuscript is structured as follows: Section 2 provides a comprehensive review of previous research in the field. The methodology adopted in this study is elaborated in Section 3, which outlines our approach to evaluating the quality of requirements through our requirement quality framework and details the computation of the well-formed requirement quality index. The findings of our research are presented and analyzed in Section 4. The paper concludes with Section 5, where we summarize our work and its implications.

## 2. RELATED WORK

### 2.1 NLP in RE

Requirements are an instance of NL. That drives the dominance of NLP in RE. The need for dedicated training and access tools for non-textual forms of requirements further advocates for NLP in RE [8, 11]. The systematic mapping study by Zhao et. al. is an excellent resource for exploring the full spectrum of applications of NLP in RE [12]. NLP has been used to detect, extract, model, classify, trace, search and retrieval of requirements. Part-of-Speech (PoS) tagging, tokenization, term extraction, stemming, stop-word removal, and parsing are some of the most commonly used NLP techniques in RE. Our work also uses some of them. The capability of NLP techniques for lexical, syntactic, semantic and pragmatic analysis of requirements was explored by Lash et. al. [13]. They proposed mapping requirements to a formal language to identify the subject, verb, object, and additional clauses in the requirement. The formal language gives rise to patterns for requirements. A formal syntax based on parts-of-speech, sentence structure and grammar was proposed by Lamar C. [14]. Structured requirements have been used to create ontologies as well [15]. The needs of customers were extracted from online product reviews using an NLP-based aspect-based sentiment analysis approach [16].

The use of structured text has been widely endorsed to overcome the shortfalls of eliciting requirements in NL. Chahadi and Herbert recommended using consistent terms while creating a semantic network from product requirements [17]. They mentioned that homonyms and synonyms should be properly incorporated into the NLP system. The work of Ref [11] is one of the widely used formalizations of RE using patterns to mitigate ambiguity, vagueness, complexity, omission, duplication, wordiness, untestability and inappropriate implementation. They proposed simple syntactic rules of thumb for using 'while' for state-driven, 'when' for event-driven, 'where' for optional features and 'if-then' statements for unwanted behaviors. Usage of their patterns reduced vagueness, wordiness and lexical ambiguity. A small segment of our work, focusing on sentence-level patterns, is in-

**TABLE 1: Characteristics of a well formed requirement [1, 10]**

| Characteristics | ID | Description |
|---|---|---|
| Necessary | C1 | Shall define an essential capability, constraint or quality factor. Its exclusion shall deem the system deficient. |
| Appropriate | C2 | The details of the requirement shall be appropriate to the level of abstraction of the requirement. |
| Unambiguous | C3 | The requirement shall be interpretable in only one way. |
| Complete | C4 | Shall define the necessary capability/constraint to meet the need without needing any additional information. |
| Singular | C5 | The requirement shall state a single capability, characteristic, constraint or quality factor. |
| Feasible | C6 | The requirement shall be realizable within the system's constraints (e.g., technology, cost, schedule). |
| Verifiable | C7 | The requirement's realization by the system shall be verifiable. The requirement shall be measurable. |
| Correct | C8 | The requirement shall accurately represent the need from which it was transformed. |
| Conforming | C9 | The requirement shall conform to an approved style of writing or a standard requirement template. |

spired by their patterns. The work of Fuentes et. al. is one of two works that are closest to our work [18]. Their work is based on the 2013 version of the INCOSE rules. In contrast, our work is based on the 2023 version, which has significant revisions. They did not offer a single metric to assess the overall individual quality of requirements. The other work is by the QVscribe tool from QRA Corp [19]. However, their implementation leave about half of the rules to be assessed by other tools and RE professionals.

### 2.2 Quality of Requirements

The quality of individual requirements has been frequently described by a set of characteristics. Numerous researchers used one or more characteristics such as necessary, verifiable, implementation-free, complete, attainable, unambiguous, verifiable, minimal, feasible, valid, traceable, consistent, allocated, unique identifier, positively stated, non-redundant [20–22]. Most of the methods measure the quality of one of the aspects of requirements, such as similarity, ambiguity [23], inconsistency [24], conflict [25]. Aceituna et. al. used a model-based approach to verify requirements for ambiguity and incompleteness [23]. Mokammel et. al. combined diverse requirement quality measures based on lexical, syntactic and semantic approaches in a single software tool [26]. Their quality grade metrics focused on consistency, completeness and expressiveness. Multiple tools were created and endorsed by NASA to craft requirements of good quality [27]. Soeken et. al. deployed ten rules acquired from literature to evaluate requirements [28]. They highlighted that the sensibility of the rules are debatable. In most of the work, the focus is on classifying if a requirement has characteristics or not. The results are often binary with specific details of the requirement. A holistic view in evaluating the quality of requirements was proposed by Parra et. al. [29]. They used an array of quality metrics such as the number of ambiguous expressions, dependencies, hierarchical levels, verbs in passive voice, words and others. A quantitative measure of the quality of requirements is proposed in [22]. They measured features such as the number of words (for atomicity), the number of ambiguous terms, the number of verbs, and the number of dependencies. Nonetheless, their measures fail to capture high-level characteristics of requirements.

## 3. METHODOLOGY

### 3.1 ISO 29148 Guidelines

We establish a formal framework to evaluate the quality of the *requirement construct* following the guidelines of ISO/IEEE/IEC 29148:2018 (E) [1]. It states that a requirement is well-formed if the requirement (1) is qualified by measurable conditions, (2) is bounded by constraints, (3) can be verified, (5) defines the performance of the system when used under a specific condition by a specific user; and finally, (5) meets a need of the stakeholder. For instance, the requirement 'the diameter of the shaft shall be 2 mm' fails to be a well-formed requirement as a dimension of exactly 2 mm is hard to achieve. The inclusion of a tolerance, such as $2 \pm 0.1$ mm, makes the requirement achievable. Some of the guidelines for the requirements to be well-formed are very specific and clear. Examples include (1) the use of 'shall'; (2) avoiding the use of 'is', 'are', 'will', 'must', or 'should'; (3) avoiding the use of negation, such as 'shall not'; (4) avoiding 'shall be able to'. They can be directly implemented by natural language processing (NLP) techniques. Nonetheless, evaluating requirements only for these factors does not ensure the requirement to be well-formed. Each individual requirement shall have the following nine characteristics to be well-formed: Necessary (C1), Appropriate (C2), Unambiguous (C3), Complete (C4), Singular (C5), Feasible (C6), Verifiable (C7), Correct (C8), and Conforming (C9) [1]. The characteristics are briefly explained in Table 1. The characteristics C1, C2, C5, C8, and C9 pertain to the *formal transformation* of the sources (stakeholders' needs or parent requirements) to acquire the requirements. The characteristics C3, C4, C6, and C7 pertain to the *agreed-to-obligation* between the stakeholders and the system.

### 3.2 INCOSE Guidelines

While ISO/IEEE/IEC 29148:2018 (E) [1] specifies the characteristics of a well-formed requirement, it does not specify how to achieve each of those characteristics, nor does it establish objective rules [30]. For instance, the questions 'how to ensure that the requirement shall be interpretable in only one way (unambiguous)' or 'how to ensure that the requirement statement does not need any additional information (completeness)' still remain unanswered. The guidelines of the Requirements Working Group of the International Council on Systems Engineering (INCOSE) go a level deeper and specify how to achieve each of those characteristics without being specific to any implementation details

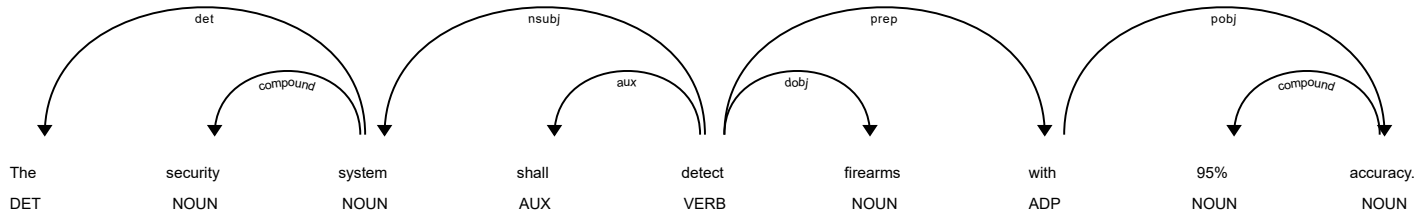


**FIGURE 1: The part-of-speech tags and the structure of a sentence in natural language applied to the requirement 'The security system shall detect firearms with 95% accuracy.'**

[8]. We do not discuss the specifics of the rules for brevity. We discuss our implementation of the INCOSE rules in §3.3. Table 4 in Appendix A summarizes the mapping between the characteristics of well-formed requirements with the INCOSE rules and offers a bird's-eye-view of our implementation.

### 3.3 Requirement Quality Framework

Our requirement quality framework is based on the premise that every requirement is an instance of natural language. All the rules of NL are applicable to requirements in addition to the requirements of requirements. Thus, we analyze the requirements at the atomic level of NL, the tokens. Tokens refer to words, symbols, punctuation, or numbers in the context of our work. Then, we evaluate the requirements at the span level. *Span* refers to an ordered set of tokens. Often, tokens depend on each other even though they are not in order. So, such dependencies can be analyzed using the syntactic structure of requirements. Lastly, we analyze the requirements at the sentence level. A few things to note are as follows: First, we consider each requirement to consist of a single sentence. In case a requirement consists of multiple sentences, we consider only the first sentence for analysis. Second, two rules (**R29** and **R42**) are not applicable to the characteristics of individual requirements.

**3.3.1 Token Based.** We implemented many of the rules using tokens or the attributes of the tokens. We attribute each token with both coarse-grained[1] PoS tags as well as their fine-grained PoS tags [31]. There are 20 coarse-grained PoS tags and 56 fine-grained tags in our implementation. The coarse PoS tags of an example requirement are shown in Fig. 1. Rule **R5** recommends the usage of the definite article 'the' over the usage of indefinite articles 'a' and 'an'. We check for the coarse-grained tag 'DET' to detect determiners and then verify whether the token is the same as 'the'. See the coarse PoS tag of the first token 'The' in Fig. 1. Similarly, **R7** prescribes to avoid vague terms. Vague terms compromise the verifiability of requirements. For example, it is hard to verify whether the requirement 'the mouse shall move smoothly' or 'the system shall display the relevant information' is correct. How much is smooth and what is relevant is hard to verify. We complemented two methods to implement **R7**. First, we used a pattern to detect adverbs. The coarse PoS tag of adverbs is 'adverb'. Then, we use a token library to detect words such as 'relevant'. It includes a list of tokens such as 'several', 'allowable', 'several', 'common', 'flexible', 'typical' and 'sufficient'. We detected pronouns (**R24**) using a mixed approach like **R7**.

Rule **R14** advises to use correct punctuation. Moreover, the presence of more punctuation in a requirement makes it more ambiguous. Thus, we use two token-based rules to assess the requirements. We inspect the coarse PoS tags to check if the token is a punctuation. Then, we check for the finer PoS tags to see if the token is a period, a comma, a colon or a hyphen. If there are more than two punctuations in a sentence, we deem the requirement to violate **R14**. The terminal period and a comma within the requirement are permissible without complicating the requirement.

**3.3.2 Span Based.** This class of implementation focuses on the spans to assess the requirements for the rules. E.g., **R8**, which recommends avoiding the inclusion of escape clauses. We match the span of the requirement with escape clauses such as 'as little as possible', 'if necessary' and 'as applicable'. Rule **R9** prescribes avoiding open-ended clauses such as 'and so on' and 'including but not limited to.' Rule **R20** prohibits the inclusion of phrases indicating reason/intent/purpose in requirements. We include example phrases such as '... to ...', 'so that', 'thus allowing' and 'in order to'. We supplemented the list of phrases with additional phrases during our implementation. **R26** recommends against the use of unachievable absolutes such as '100%', 'every', 'all', 'never' and 'always'. They make the requirements non-verifiable. We detect such phrases along with additional phrases that play a similar role, such as 'without exception', 'at all times', 'without interruption'. We use both token-based and span-based features to implement **R26**.

**3.3.3 Syntactic Structure-based.** If the order of the tokens is altered, the span-based patterns become ineffective. Patterns based on the syntactic structure of the requirements come in handy in such cases where we exploit the connection between various tokens irrespective of their order. We represent the syntactic structure of the requirements using a tree structure. Every token in the requirement becomes a node in the tree. The syntactic relations between pairs of tokens become the links in the tree. The tree is rooted at the verb token. Other tokens (such as subject, object, period, and preposition) and subtrees (from phrases, such as conditions, constraint and capabilities of requirements) are connected to the main tree or other subtrees. The nodes are attributed with their coarse-grained and fine-grained PoS tags from the Penn Treebank [31]. The tree structure for an example requirement is depicted in Fig. 2. Consider **R6**, which recommends the use of common units of measure. The units of measure are nouns, and

[1] https://universaldependencies.org/u/pos/

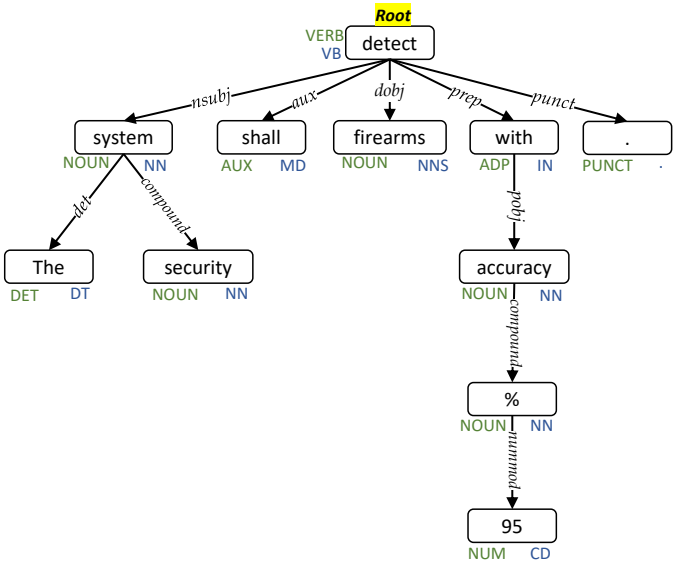


**FIGURE 2: The tree structure of the requirement 'The security system shall detect firearms with 95% accuracy.'. The coarse PoS tags and the fine PoS tags of the tokens and the dependencies among the tokens are also depicted.**

they succeed numeric values. Units such as 'm/s' and 'V/mm' are a composition of units of measure. We use the dependency of the token to classify it as a unit. If a token's dependency is 'numeric modifier', we label it as a unit. Then, we use a bank of units for each measuring system (such as Metric or Imperial) to check if the unit is consistent with one of the measuring systems. While **R6** needs the link between a pair of tokens, some rules call for scrutinizing the dependencies of multiple tokens. We consider the case of **R2**, use of active voice, to illustrate this. The onus of satisfying a requirement is with the subject of the sentence. The active voice clearly identifies the entity responsible for making the action. A span-based implementation to detect the presence of 'shall be' by generalizing from requirements like 'The firearms shall be detected by the security system.' labels requirements such as 'The length of the firearm shall be within 1m.' as passive voice. The latter is in active voice. In our implementation, we check for the dependencies of the tokens connected to the root verb. If the tokens with 'nominal subject (passive)', 'auxiliary (passive)' dependencies are connected to the root node, we classify the requirement to be in passive voice.

Furthermore, we combine token-based and tree-based rules as per necessity and effectiveness. We implement **R10**, which prescribes avoiding superfluous infinitives. We inspect the tokens and check if their fine-grained PoS tag is 'infinitival to'. If yes, we check for the coarse-grained PoS tag on its head to verify if it is a verb. The requirement contains superfluous infinitives if both conditions are evaluated to be true. We inspect the subtrees in the tree structure of the requirement to verify if a separate clause is used for each condition or qualification (for **R11**).

**3.3.4 Sentence Based.** We implement a pattern-based approach to evaluate the structure of requirements. These patterns are essentially a series of building blocks (pattern slots) to construct a well-formed requirement. A requirement shall consist of a *capability*, a *condition* and a *constraint*. The basic pattern for a requirement is 'The <entity> shall <do what>, <how well>, <under what conditions>.' For instance, 'The autonomous driving system shall safely navigate urban areas (capability) at a minimum speed of 40 mph (condition) while adhering to traffic rules' (constraint). This pattern ensures that the requirement is complete and the system is verifiable with the requirement. We implement five patterns in our framework to assess if the requirement is a structured statement (**R1**). The patterns are listed in Table 2. We focus on multiple aspects of the requirements to evaluate if a requirement follows one of the patterns. First, we construct the tree structure of the requirements. Adherence to the basic pattern is evaluated by identifying the dependencies of the root verb. The primary subject and primary object of a sentence have the 'nominal subject' and 'direct object' dependencies, respectively. The 'measurable response' is a phrase that becomes a sub-tree in the sentence tree. See Fig. 2 and 3 for the phrase 'with 95% accuracy'. The subtree is attached to one of the primary components of the sentence, the subject, the verb or the object. The subtree is attached to the primary tree with a 'prepositional modifier' or a 'clausal modifier of noun (adjectival clause)' or an 'adverbial clause modifier'. For every requirement, we conduct a minimum check for the presence of the primary subject, verb and object. Then, we check for the presence of the sub-tree. We conclude that a requirement follows the basic pattern if both conditions are satisfied. We want to emphasize that the presence of only ['subject', 'verb', 'object'] still makes it a requirement. It is a functional requirement such as 'The security system shall detect firearms.' Such a requirement does not include a measurable outcome as in how often the security system shall detect or miss detecting firearms so that the system becomes verifiable. Inclusion of a measurable outcome, as in 'The security system shall detect firearms with 95% accuracy' with the preposition 'with' makes the system verifiable. Furthermore, as per the standard INCOSE [8] and EARS [11] guidelines, we classify the requirement as 'event-driven' or 'state-driven' or 'unwanted-behavior' or 'optional feature' based on the presence of the keywords 'when' or 'while' or 'if' or 'where' respectively (See Table 2). See Fig. 3 illustrating that the condition in a state-driven requirement is connected to the root verb using an 'adverbial clause modifier.' These patterns place the identifying keywords at the beginning of the sentences without explicitly restricting the locations of these keywords. We do not restrict the location of the keywords in our implementation. Lastly, we classify the requirement as a complex requirement if the requirement satisfies more than one criterion.

**3.3.5 Others.** We have implemented rules for which we need information beyond just the requirement itself. For instance, **R4** advises to define all the terms in the requirements with a glossary. This glossary is specific to a set of requirements, as in a requirement specification document or the common terminology of a project. Our framework accepts this glossary and verifies if the requirement contains one or more terms from the glossary. Similarly, **R12** advises for using correct grammar, and **R13** ad-

**TABLE 2: Patterns for requirements**

| Type | Pattern | Example |
|---|---|---|
| Basic | The <SoI> shall <action> <measurable response>. | The security system shall detect firearms with 95% accuracy. |
| Event-driven | When < trigger>, the <SoI> shall <system response>. | When the sensors detect unauthorized access, the security system shall detect firearms with 95% accuracy. |
| State-driven | While <precondition)>, the <SoI> shall <system response>. | While the security system is in 'alert' mode, it shall detect firearms with 95% accuracy. |
| Unwanted behavior | If <trigger>, then the <SoI> shall <system response>. | If a person carries a concealed firearm, the security system shall detect firearms with 95% accuracy. |
| Optional Feature | Where <feature is included>, the <SoI> shall <system response>. | Where the advanced threat detection module is enabled, the security system shall detect firearms with 95% accuracy. |
| Complex | More than one of the above cases are applicable. | While the security system is in 'alert' mode, if motion sensors detect unauthorized access into the premises, the security system shall detect firearms with 95% accuracy. |

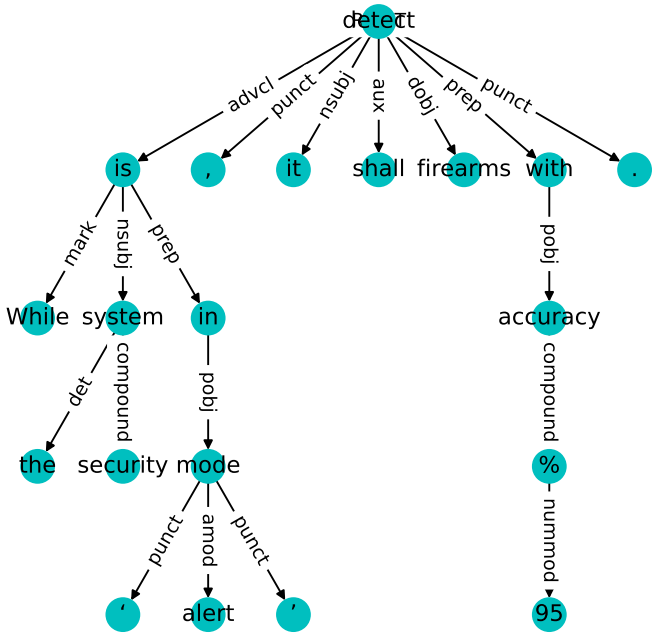


**FIGURE 3: The tree structure of a state-driven requirement from Table 2 showing that the condition is connected to the root verb with an 'adverbial clause modifier'.**

vises for using correct spelling. We use the LanguageTool[2] for checking for grammar and spelling. It uses 6000+ rules to check for English grammar and spelling [32].

### 3.4 Requirement Quality Index

We evaluate every requirement for each of the 42 rules. During implementation, we produce binary results: 1 if the requirement satisfies a rule and 0 otherwise. Evaluation of each requirement produces a binary vector of length 42, one dimension for each rule. There are $[2, 3, 30, 16, 8, 2, 25, 11, 10]$ rules applicable for the characteristics [C1, C2, C3, C4, C5, C6, C7, C8, C9] respectively. See the last nine columns (grey color) in Table 4 where 'O' indicates that a rule is applicable to a characteristic. An empty cell indicates that the rule is not applicable for specific characteristics. The *rule-to-characteristics* ($RC$) mapping matrix for individual requirements is a $42 \times 9$ matrix. In case of a list of $l$ requirements, the *normalized characteristics matrix* $C^N$ can be computed as

$$C^N_{l\times 9} = \frac{RR_{l\times 42} \times RC_{42\times 9}}{\mathbf{c}^w_{1\times 9}} \tag{1}$$

where $RR$ is the *rule evaluation matrix* for $l$ requirements, and $C^N$ is the *normalized characteristics matrix* for $l$ requirements. The subscripts specify the size of the matrices and vectors. $\mathbf{c}^w$ normalizes the characteristics matrix. $\mathbf{c}^w$ is the characteristic vector of a well-formed requirement, a requirement that satisfies all the rules, *i.e.*, $\mathbf{c}^w = [2, 3, 30, 16, 8, 2, 25, 11, 10]$. The normalization scales the value corresponding to each characteristic in $C^N$ to 1. A requirement that does not satisfy all the rules will have its elements in its characteristic vector less than 1. Substituting $l = 1$ in Eq. 1 produces the equation to compute the normalized characteristic vector for a single requirement. The normalized characteristic vector of a well-formed individual requirement is equal to the unit vector of length 9, one for each characteristic. We quantify the well-formed nature of a requirement by the *well-formed requirement quality* ($wRQ$) index, which is the cosine similarity between the normalized characteristic vector of an individual requirement and that of a well-formed requirement.

[2] https://github.com/jxmorris12/language_tool_python

**TABLE 3: Well-formed requirement quality of a few sample requirements including their imperfect characteristics and the rules violated by the requirements**

| ID | Requirement | $wRQ$ | Imperfect Characteristics | Violated Rules |
|---|---|---|---|---|
| $Req1$ | The autonomous driving system shall safely navigate urban areas at a minimum speed of 40 mph while adhering to traffic rules. | 0.985353 | C3, C4, C6, C7, C8 | R5, R7, R33, R34 |
| $Req2$ | The autonomous driving system shall navigate urban areas at the speed of 40 ± 5 mph while adhering to traffic rules. | 1 | None | None |
| $Req3$ | The firearms shall always be detected by the security system for the safety of the employees. | 0.756098 | C1, C2, C3, C4, C5, C6, C7, C8, C9 | R1, R2, R20, R26 |
| $Req4$ | The vehicle shall be able to operate in a variety of temperatures. | 0.999425 | C3, C4, C7 | R5, R7 |
| $Req5$ | The ASD shall process all radio messages at a minimum rate of 10 Hz. | 0.985360 | C3, C4, C6, C7, C8 | R5, R32, R33, R34 |

The well-formed requirement quality index for $l$ requirements is given by

$$wRQ^{i}_{l\times 1} = C^{N}_{l\times 9} \odot \mathbf{I}_{1\times 9} \tag{2}$$

where the $\odot$ represents the cosine similarity between the two vectors. $\mathbf{I}$ is the normalized characteristics vector of a well-formed requirement. Since it satisfies all the rules, its normalized characteristic vector becomes a unit vector upon normalization. $wRQ^{i}$ is the well-formed requirement quality index of $l$ individual requirements. The superscript $i$ represents individual requirements. Finally, we define the individual well-formed requirement quality of a list of requirements ($wRQ^{l}$) as the average of the well-formed requirement quality index of each individual requirement. It is defined as

$$wRQ^{l} = \frac{1}{l}\sum_{i=1}^{l} wRQ^{i} \tag{3}$$

The well-formed requirement quality index is useful for comparing the quality of lists of requirements, such as requirement specification documents. $wRQ \in [0, 1]$, i.e., the well-formed requirement quality index, lies between 0 and 1. A well-formed individual requirement has $wRQ^{i} = 1$, whereas a requirement that does not satisfy any rule will have $wRQ^{i} = 0$.

## 4. RESULTS AND DISCUSSION

### 4.1 Quality of an Individual Requirement

We illustrate our requirement quality framework by evaluating the individual requirement: 'The autonomous driving system shall safely navigate urban areas at a minimum speed of 40 mph while adhering to traffic rules.' which is listed as *Req1* in Table 3. It follows the ISO/IEC/IEEE 29148:2018 definition of a requirement to include a condition and a constraint. Nonetheless, evaluating this requirement with the fourty-two INCOSE rules (See Table 4 in Appendix A) highlights the violation of four rules: **R5** (includes an indefinite article 'a'), **R7** (includes the vague terms 'minimum' and 'safely'), **R33** (did not include a range of values for the speed), **R34** (includes the keyword 'minimum' for a measurable performance). Evaluating this requirement with the rules produce the 42-dimensional rule evaluation vector: $RR_{1\times 42} = [1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1]^{T}$. It has zeros at the $5^{th}, 7^{th}, 33^{rd}$ and the $34^{th}$ positions indicating that the requirement has violated the rules: **R5**, **R7**, **R33** and **R34**. Multiplying this with the $RC_{42\times 9}$ matrix produces the characteristics vector $C_{1\times 9} = [2, 3, 26, 13, 8, 1, 21, 10, 10]$. The indices of the red-colored values correspond to the imperfect characteristics: unambiguous (C3), complete (C4), feasible (C6), verifiable (C7) and correct (C8). Upon dividing the characteristics vector $C$ by the well-formed characteristic vector $\mathbf{c}^{w}$, we acquire the normalized characteristic vector $C^{N} = [1, 1, 0.8667, 0.8125, 1, 0.5000, 0.8400, 0.9091, 1]$. The values in the normalized characteristic vector which are not equal to 1 represent the imperfect characteristics, and vice-versa. Now, the $wRQ$ index of *Req1* is 0.985353 which is its cosine similarity with the normalized characteristic vector of a well-formed requirement (=$\mathbf{I}_{1\times 9}$, a nine-dimensional unit vector).

### 4.2 Comparison of Requirements

We show the effectiveness of our requirement quality framework and $wRQ$ by comparatively evaluating few more individual requirements. See Table 3. Upon revisiting *Req1*, we notice that the benchmark for the target system is not clear. The specified minimum speed of '40 mph' can be achieved by an autonomous vehicle with a top speed of 60 mph or 80 mph or a vehicle with an average speed below 40 mph which can reach a top speed beyond 40 mph. Moreover, the term 'safely' is hard to define and makes the requirement impossible to validate without the proper definition of 'safe'. Since the requirements are elicited from the perspective of the system, it is not clear (1) if the keyword 'safe' in the requirement refers only to the safety of the vehicle or also includes the safety of its occupants; (2) if the vehicle shall be deemed safe if the vehicle experienced an accident without injuring any of its occupants; (3) if an autonomous vehicle that faced an accident without injuring any of its occupants is equally safe as a vehicle which avoided all accidents. *Req1* can be rewritten as 'The autonomous driving system shall navigate urban areas at the speed of 40 ± 5 mph while adhering to traffic rules.'. The rewritten form does not violate any rules, satisfies all the characteristics, and has a $wRQ$ index equal to one. The rewritten form is a well-formed requirement. It is shown as *Req2* in Table 3. Additionally, *Req1* needs the definition of 'traffic rules' to

elicit the expected behaviour of the autonomous vehicle when specific traffic rules are applicable. *Req1* shall be decomposed into child requirements such as 'when a red signal is detected, the autonomous driving system shall stop within 30 ± 5 seconds'.

The requirement $Req3$ in Table 3 violates four rules: **R1** as it does not match any of the specified patterns due to lack of a measurable performance, **R2** as it is in passive voice, **R20** for including the purpose phrase 'for the safety of the employees', and **R26** for including the unachievable keyword 'always'. As these four rules altogether affect all the nine characteristics, *Req3* fails to achieve any of the nine characteristics even though its $wRQ = 0.756098$. Nonetheless since only four rules are violated, it is relatively easier to rewrite *Req3* (for instance, 'The security system shall detect firearms with 95 ± 2% accuracy.') to make it a well-formed requirement.

Now, we consider a requirement $Req4$ from the Deep Orange 13[3] project developed at Clemson University. The project developed a non-combat autonomy-enabled off-road vehicle. $Req4$ violated **R5** (used the indefinite article 'a') and **R7** (has a vague term 'variety'). Specifying the operating conditions as 'a variety of temperatures' makes the requirement ambiguous, incomplete and non-verifiable. Thus, the characteristics C3, C4, C7 are imperfect. Specifying a range of temperatures as '0-40 F' or '80-100 F' would have demanded the vehicle to operate in cold or hot deserts. Our last example, $Req5$, is from the system requirement specification of New York City's connected vehicle pilot deployment program [33]. It failed to satisfy **R5** (presence of the indefinite article 'a'), **R32** (includes 'all' for universal qualification), **R33** (did not use a range of values for '10 Hz') and **R34** (includes the keyword 'minimum' for measurable performance). Consequently, it did not meet the unambiguous (C3), complete (C4), feasible (C6), verifiable (C7) and correct (C8) characteristics.

We want to emphasize the differences between the requirements listed in Table 3. The requirements $Req1$ and $Req3$, being random samples, do not have an accompanying glossary (**R4**), a hierarchy (for **R3** to use the subject or verb appropriate to the entity), or a heading (**R25**). These rules are not applicable to $Req1$ and $Req2$. Such rules are evaluated to be true. On the other hand, requirements drawn from system requirement specifications, as $Req4$ from [33], have an accompanying glossary, acronyms, hierarchy and other additional details. More rules are applicable. The additional rules are evaluated accordingly. E.g., $Req4$ has an acronym ASD, an acronym for aftermarket safety device. Thus, **R37**, using consistent acronyms, is applicable and evaluate to be true. Furthermore, **R23** recommends a supporting diagram or model. This rule fails if the requirement refers to an object and the object does not accompany the requirement. Since none of the example requirements refer to a supporting object, we evaluate them to satisfy the rules.

## 5. CONCLUSION

We introduce an innovative framework for quantitatively assessing the quality of system requirements. Leveraging advanced rule-based NLP techniques in accordance with ISO/IEC/IEEE 29148:2018 (E) and INCOSE guidelines, we propose the well-formed requirement quality (wRQ) index as a metric for objectively evaluating the quality of individual system requirements. The comprehensive framework outlined herein underscores the critical characteristics that requirements must embody to be deemed well-formed. Those characteristics are necessary, appropriate, unambiguous, complete, singular, feasible, verifiable, correct, and conforming. We include practical examples and a detailed discussion on the implementation of the wRQ. The wRQ emerges as a pivotal tool for professionals in requirements engineering, offering a precise measure to gauge and enhance the quality of system requirement specifications. In the future, we will extend our framework to accommodate broader and more complex requirement scenarios, such as recursion in sentence-based patterns. Furthermore, we will upgrade our framework from assessing the rules to correcting requirements to satisfy the rules. Our work contributes significantly to the field of requirements engineering and sets a new benchmark for the systematic improvement of requirement quality, ultimately facilitating the development of more reliable, efficient, and successful systems.

[3] https://cuicardeeporange.com/project/do13/

## REFERENCES


[1] ISO 29148:2018(E). "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering." *ISO/IEC/IEEE 29148:2018(E)* : pp. 1–104DOI 10.1109/IEEESTD.2018.8559686.

[2] Wheatcraft, Louis S. "5.1. 2 Triple Your Chances of Project Success Risk and Requirements." *INCOSE International Symposium*, Vol. 21. 1: pp. 543–558. 2011. Wiley Online Library.

[3] Office of Inspector General. "2023 Report On Nasa's Top Management And Performance Challenges." (2023). URL https://oig.nasa.gov/docs/MC-2023.pdf.

[4] Stecklein, Jonette M, Dabney, Jim, Dick, Brandon, Haskins, Bill, Lovell, Randy and Moroney, Gregory. "Error cost escalation through the project life cycle." *14th Annual International Symposium*. JSC-CN-8435. 2004.

[5] Boehm, Barry W. *Software engineering economics*. Springer (2002).

[6] PMI. *Requirements Management: A Core Competency For Project And Program Success*. Project Management Institute (PMI) (2014).

[7] Ryan, Michael J and Wheatcraft, Louis S. "On a cohesive set of requirements engineering terms." *Systems Engineering* Vol. 20 No. 2 (2017): pp. 118–130.

[8] Ryan, Michael, Wheatcraft, Ryan and Requirements Working Group, INCOSE. "INCOSE Guide to Writing Requirements, Jul-2023." *International Council on Systems Engineering* (2023): pp. 1–144.

[9] Sen, Chiradeep, Mukhopadhyay, Alolika, Fields, John and Ameri, Farhad. "An Approach for Measuring Information Content of Textual Engineering Requirements Using a Form-Neutral Representation." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 46292:

p. V01BT02A006. 2014. American Society of Mechanical Engineers.

[10] International Council on Systems Engineering, INCOSE-TP-2003-002-04.1 *Systems Engineering Handbook, A Guide for System Life Cycle Processes and Activities*. John Wiley & Sons Ltd. (2023).

[11] Mavin, Alistair, Wilkinson, Philip, Harwood, Adrian and Novak, Mark. "Easy Approach to Requirements Syntax (EARS)." *2009 17th IEEE International Requirements Engineering Conference*: pp. 317–322. 2009. DOI 10.1109/RE.2009.9.

[12] Zhao, Liping, Alhoshan, Waad, Ferrari, Alessio, Letsholo, Keletso J, Ajagbe, Muideen A, Chioasca, Erol-Valeriu and Batista-Navarro, Riza T. "Natural language processing for requirements engineering: A systematic mapping study." *ACM Computing Surveys (CSUR)* Vol. 54 No. 3 (2021): pp. 1–41.

[13] Lash, Alex, Murray, Kevin and Mocko, Gregory. "Natural language processing applications in requirements engineering." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 45011: pp. 541–549. 2012. American Society of Mechanical Engineers.

[14] Lamar, Carl. "Linguistic analysis of natural language engineering requirements." Ph.D. Thesis, Clemson University. 2009.

[15] Lemke, Mark T, Stone, Robert B and Arlitt, Ryan A. "Ontologies to support customer requirement formulation in aerospace design." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 58219: p. V007T06A018. 2017. American Society of Mechanical Engineers.

[16] Mokadam, Aashay, Shivakumar, Shrikrishna, Viswanathan, Vimal and Agumbe Suresh, Mahima. "Online Product Review Analysis to Automate the Extraction of Customer Requirements." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 85420: p. V006T06A049. 2021. American Society of Mechanical Engineers.

[17] Chahadi, Youssef and Birkhofer, Herbert. "Semantic Product Requirement Network as Approaches for a Requirement Terminology and Tool for Linguistic Analysis of Requirements in Continuous Text." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 43277: pp. 223–230. 2008.

[18] Fuentes, Jose, Fraga, Anabel, Génova, Gonzalo, Parra, Eugenio, Alvarez, Jose María and Llorens, Juan. "Applying INCOSE Rules for writing high-quality requirements in Industry." *INCOSE International Symposium*, Vol. 26. 1: pp. 1875–1889. 2016. Wiley Online Library.

[19] QRA Corp. "Automating the INCOSE Guide for Writing Requirements." URL https://qvscribe.com/hubfs/Automating-The-INCOSE-Guide-For-Writing-Requirements.pdf.

[20] Boehm, Barry W. "Verifying and validating software requirements and design specifications." *IEEE software* Vol. 1 No. 1 (1984): p. 75.

[21] Hirshorn, Steven R, Voss, Linda D and Bromley, Linda K. "Nasa systems engineering handbook." Technical report no. 2017.

[22] Génova, Gonzalo, Fuentes, José M, Llorens, Juan, Hurtado, Omar and Moreno, Valentín. "A framework to measure and improve the quality of textual requirements." *Requirements engineering* Vol. 18 (2013): pp. 25–41.

[23] Aceituna, Daniel, Walia, Gursimran, Do, Hyunsook and Lee, Seok-Won. "Model-based requirements verification method: Conclusions from two controlled experiments." *Information and Software Technology* Vol. 56 No. 3 (2014): pp. 321–334.

[24] de Sousa, Thiago C, Almeida Jr, Jorge R, Viana, Sidney and Pavón, Judith. "Automatic analysis of requirements consistency with the B method." *ACM SIGSOFT Software Engineering Notes* Vol. 35 No. 2 (2010): pp. 1–4.

[25] Sardinha, Alberto, Chitchyan, Ruzanna, Weston, Nathan, Greenwood, Phil and Rashid, Awais. "EA-Analyzer: automating conflict detection in a large set of textual aspect-oriented requirements." *Automated Software Engineering* Vol. 20 (2013): pp. 111–135.

[26] Mokammel, Faisal, Coatanea, Eric, Christophe, Francois, Ba Khouya, Mohamed and Medyna, Galina. "Towards an approach for evaluating the quality of requirements." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 55867: p. V02BT02A024. 2013. American Society of Mechanical Engineers.

[27] McCoy, James R. "NASA software tools for high-quality requirements engineering." *Proceedings 26th Annual NASA Goddard Software Engineering Workshop*: pp. 69–69. 2001. IEEE Computer Society.

[28] Soeken, Mathias, Abdessaied, Nabila, Allahyari-Abhari, Arman, Buzo, Andi, Musat, Liana, Pelz, Georg and Drechsler, Rolf. "Quality assessment for requirements based on natural language processing." *Forum on Specification and Design Languages. Proceedings*. 2014. Citeseer.

[29] Parra, Eugenio, Dimou, Christos, Llorens, Juan, Moreno, Valentín and Fraga, Anabel. "A methodology for the classification of quality of requirements using machine learning techniques." *Information and Software Technology* Vol. 67 (2015): pp. 180–195.

[30] Gregory, Sarah. "Requirements engineering: The quest for meaningful metrics: Time for a change?" *IEEE Software* Vol. 36 No. 6 (2019): pp. 7–11.

[31] Petrov, Slav, Das, Dipanjan and McDonald, Ryan. "A universal part-of-speech tagset." *arXiv preprint arXiv:1104.2086* (2011).

[32] "LanguageTool Community: Error Rules for LanguageTool."

[33] New York City Department of Transportation. "Connected Vehicle Pilot Deployment Program Phase 1 Systems Requirements Specification (SyRS) – New York City." (2017).

## APPENDIX A. INCOSE RULES TO CHARACTERISTICS MATRIX [8]

**TABLE 4: Rules to characteristics (RC) mapping matrix [8]. The *characteristics* of individual requirements are necessary (C1), appropriate (C2), unambiguous (C3), complete (C4), singular (C5), feasible (C6), verifiable (C7), correct (C8), and conforming (C9). The *type* of implementation is split into token-based (T), span-based (s), syntactic structure-based (Sy), sentence-based (S) and others (O)**

| Rule | Title | Details | Type | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R1 | Use Structured Statements | Conform to one of the agreed patterns | S | | | O | O | | | O | O | O |
| R2 | Use Active Voice | With the responsible entity as the subject | Sy | | O | O | O | | | O | | |
| R3 | Appropriate Subject-Verb | Use subject and verb appropriate to the entity | T | | O | O | | | | O | | |
| R4 | Use Defined Terms | Define all the terms in an associated glossary | T | | | O | | | | O | | |
| R5 | Use Definite Articles | Use ‘the’ instead of ‘a’ or ‘an’ | T | | | O | | | | O | | |
| R6 | Common Units of Measure | Consistent units with common measurement system | T | | | O | O | | | O | O | |
| R7 | Avoid vague Terms | E.g., ‘some’, ‘relevant’, ‘efficient’, ‘typical’ | T | | | O | O | | | O | | |
| R8 | Avoid Escape Clauses | E.g.,‘if necessary’, ‘as appropriate’, ‘as required’ | s | | | O | | | | O | | |
| R9 | Avoid Open-ended Clauses | E.g., ‘including but not limited to’, ‘and so on’ | s | | | O | O | O | | O | | |
| R10 | Avoid Superfluous infinitives | E.g., ‘to be able to’, ‘to allow’, ‘to enable’ | s | | | O | | | | O | | |
| R11 | Separate Clauses | Use a separate clause for each condition/qualification | Sy | | | O | O | | | O | O | |
| R12 | Correct Grammar | Use grammatically correct sentences | O | | | O | | | | O | O | O |
| R13 | Correct Spelling | Use correct spelling | O | | | O | | | | O | | |
| R14 | Correct Punctuation | Use correct and fewer punctuations | T | | | O | | | | | O | |
| R15 | Logical Expressions | Use a defined convention for logical expressions | Sy | | | O | | | | O | | |
| R16 | Avoid the Use of ‘not’ | Avoid negation in requirements | T | | | O | | | | O | O | |
| R17 | Avoid the Oblique Symbol | Do not use the oblique (‘/’) symbol | T | | | O | | | | O | | |
| R18 | Single-thought Sentence | Single sentence qualified by relevant sub-clauses | Sy | | | O | | O | | O | | O |
| R19 | Avoid Combinators | E.g., ‘and’, ‘then’, ‘or’, ‘but’, ‘whether’, ‘whether’ | T | | | O | | O | | | | |
| R20 | Avoid Purpose Phrases | Indicates intent/reason, e.g., ‘... because ...’, ‘... to ...’ | s | O | | | | O | | | | |
| R21 | Avoid Parentheses and Brackets | They contain subordinate text. | T | | | | | O | | | | |
| R22 | Enumerate Sets Explicitly | Instead of using a group noun to name the set | T | | | O | | O | | | | |
| R23 | Supporting Diagram, Model | Refer to relevant figure/diagram/model | T | | | O | O | O | | | | |
| R24 | Avoid Pronouns | Avoid personal and indefinite pronouns | T | | | O | O | | | O | | |
| R25 | Avoid Reliance on Headings | Avoid relying on headings to get context | T | | | | O | | | | | |
| R26 | Avoid Unachievable Absolutes | E.g., ‘100%’, ‘all’, ‘always’, ‘never’ | T | | | | | | O | O | O | |
| R27 | Use Explicit Conditions | State conditions’ applicability explicitly | Sy | | | | O | | | O | O | |
| R28 | Use Multiple Conditions | Multiple conditions for single action is better | Sy | | | O | | | | O | | |
| R29 | Classify Requirements | According to the aspects of the problem or system | NA | | | | | | | | | |
| R30 | Unique Expression | Express each requirement once and only once | S | O | | | | | | | | O |
| R31 | Be Solution Free | Avoid stating implementation in a requirement | Sy | | O | | | | | | | |
| R32 | Universal Qualification | Use ‘each’ instead of ‘all’, ‘any’ or ‘both’ | T | | | O | | | | O | O | |
| R33 | Use a Range of Values | Define each quantity with a range of values | Sy | | | O | O | | O | O | O | |
| R34 | Measurable Performance | Provide specific measurable performance targets | T | | | O | O | | | O | | |
| R35 | Explicit Temporal Dependencies | Instead of indefinite ones: ‘until’, ‘simultaneous’ | T | | | O | O | | | O | | |
| R36 | Consistent Terms and Units | Consistent across the project’s lifecycle and ontology | T | | | O | | | | | O | O |
| R37 | Use Consistent Acronyms | Consistent throughout requirements and the lifecycle | T | | | O | | | | | | O |
| R38 | Avoid Abbreviations | Avoid in requirements, models and lifecycle artefacts | Sy | | | | | | | | | O |
| R39 | Use a Style Guide | Use a project-wide style guide for all requirements | T | | | | O | O | | | | O |
| R40 | Consistent Decimal Format | Use a consistent format and number of signification digits | T | | | O | O | | | | | O |
| R41 | Related Requirements | Group related requirements together | S | | | | O | | | | | O |
| R42 | Structured Sets | Conform to a template for organizing requirements | NA | | | | | | | | | |