Section 4: Rules for Individual Need and Requirement Statements and for Sets of Needs and Requirements

This section defines the rules for individual need statements and requirement statements, needs expressions and requirements expressions, and sets of needs and sets of requirements that help them to be formulated such that they will be well-formed helping to have the characteristics defined in this Guide.

Included with each rule is an explanation of the rule, any qualifications or exemptions to the rule, followed by selected examples of the application of the rule.

In addition to the rules outlined in this section, the reader is encouraged to follow the principles of good technical writing (as they apply to a need or requirement statement) such as those outlined in the Simplified Technical English (STE) specification (ASD-STE100).

Note: Because need statements are expressed at a higher level of abstraction, some of the rules in this section may not always apply as rigorously to need statements as they do to requirement statements. The key is that the need statements have the characteristics defined earlier that are appropriate to the level of abstraction at which the need statements are written. Appendix D provides a matrix that maps the applicability of each rule to need statements and requirement statements in order to provide readers with a clearer understanding of the applicability of the guidance provided by each rule.

Appendix E contains cross reference matrices mapping the rules to the characteristics discussed in this Guide as well as mapping concepts and activities and attributes discussed in the NRM to the characteristics which also contribute to the quality of the need statements and requirement statements.

4.1 Accuracy

4.1.1 R1 – Structured Statements

Definition:
Need statements and requirement statements must conform to one of the agreed patterns, thus resulting in a well-structured complete statement.

Elaboration:
Organizations should provide needs and requirements writers with a catalogue of agreed patterns to follow. Each different type of need or requirement should have, at least, one assigned pattern. Following a specific agreed pattern boosts consistency within the sets of needs and sets of requirements, where similar requirements (requirements of a same types) will always follow a similar structure. When one (and only one) pattern is followed for a given need statement or requirement statement, the quality characteristics defined in Section 2 will be leveraged. Ambiguity will be reduced (C3), completeness of each need statement and requirement statement is guaranteed (C4), each need statement or requirement statement will be singular (C5), verifiability and

validatability are enforced (C7), and each need statement and requirement statement will be conforming (C9).

If a condition that applies to a need or requirement is not stated explicitly within the need statement or requirement statement, the need or requirement will not be Unambiguous (C3), Complete (C4), Verifiable/Validatable (C7), nor Correct (C8).

Detailed information on the different blocks that comprise a need or requirement pattern and further information about patterns and examples can be found in Appendix C. See also R2, R3, R11, R18, and R27.

Examples:

See Appendix C, for examples of need statements and requirement statement patterns.

4.1.2 R2 – Active Voice

Definition:

Use the active voice in the need statement or requirement statement with the responsible entity clearly identified as the subject of the sentence.

Elaboration:

The active voice requires that the entity performing the action is the subject of the sentence.

The active voice is important in writing needs and requirements since the onus for satisfying the requirement is on the subject, not the object of the statement. If the entity responsible for the action is not identified explicitly, it is unclear who or what should perform the action making it difficult to determine how to verify/validate the statement since it is unclear as to which entity should be subject to the verification/validation activity.

Including the entity in the subject also helps ensure the need statement or requirement statement refers to the appropriate level requirement set consistent with the entity name (see R3).

If the entity is not clearly identified as the subject of the statement, the need or requirement is not Complete (C4).

Often when phrases such as "shall be" are used followed by a main verb, the statement is in the passive voice, which is why all the patterns into the agreed catalog of patterns (see R1 and Appendix C) shall make use of the active voice.

Examples:

Unacceptable: The Identity of the Customer shall be confirmed.

[This is unacceptable because it does not identify the entity that is responsible/accountable for confirming the identity.]

Improved by adding the subject: The Identity of the Customer shall be confirmed by the Accounting_System.

[Although this statement makes the subject clear, it is still unacceptable because the statement is in the passive voice in that the entity that is responsible/accountable for recording the audio is at the end of the statement rather than the beginning.]

Improved: The Accounting_System shall confirm the Customer_Identity.

[Note that "Accounting_System", "confirm", and "Customer_Identity" must be defined in the glossary since there are a number of possible interpretations of these terms—see R4. Also note that this statement is improved because it is in the active voice but, before it is acceptable, it needs to be further improved through the addition of appropriate condition and qualifying clauses]

Unacceptable: While in the Operations_Mode, the Audio having the characteristics defined in <ICD xxxx> shall be recorded.

[This is unacceptable because the entity that is responsible/accountable for recording the audio is not stated, which makes it difficult to identify which entity is responsible for the action, or how verification is to occur. In addition, the characteristics of the Audio are not referenced, nor are the conditions under which the action specified.]

Improved by adding the subject: While in the Operations_Mode, the Audio having the characteristics defined in <ICD xxxx> shall be recorded by the <SOI>.

[Although this statement makes the subject clear, it is still unacceptable because the statement is in the passive voice in that the entity that is responsible/accountable for recording the audio is at the end of the statement rather than the beginning.]

Improved: While in the Operations_Mode, the <SOI> shall record Audio having the characteristics defined in <ICD xxxx>.

[This is improved because it is in the active voice, but appropriate performance parameters should be added before the statement is acceptable. Also note that "Audio" must be defined in the glossary or data dictionary and required performance added for the requirement to be complete. The characteristics of the Audio received from a <xxxx> source is defined in the ICD.]

4.1.3 R3 – Appropriate Subject-Verb

Definition:
Ensure the subject and verb of the need or requirement statement are appropriate to the entity to which the statement refers.

Elaboration:

Subject

The subject of a need or requirement statement must be appropriate to the entity to which the statement refers, as discussed in Section 1.7.

Requirements referring to the:

- business management level have the form "The <business> shall …";
- business operations level have the form "The <business unit> shall …";
- system level have the form "The <SOI> shall …";
- subsystem level have the form "The <subsystem> shall ..."; and
- system element level have the form "The <system element> shall ...".

All these different structures (with the different subjects) shall be included in the catalog of agreed patterns (see R1 and Appendix C).

As a general rule, sets of need statements and sets of requirement statements for a specific entity should only contain needs or requirements for that entity—that is:

- a set of business management requirements would contain only business management needs or requirements,
- a set of business operations requirements would contain only business operations needs or requirements,
- a set of system requirements would contain only needs or requirements that apply to the integrated system,
- a set of subsystem requirements would contain only needs or requirements that apply to that subsystem, and
- a set of system element requirements would contain only needs or requirements that apply to that system element.

In some cases, however, a higher-level entity may wish to prescribe needs and requirements that apply to a lower-level entity. For example, it may be important for a business to mandate that the new aircraft under development must use a particular engine (perhaps for support reasons) in which case they may make a statement at the business level that refers to an entity at the subsystem level.

Therefore, any set of entity needs or requirements can state, for that entity, needs or requirements that refer to itself, as well a lower-level entity to which the need or requirement applies (when there is a valid reason to do so). When this is the case, the prescriptive allocated need or requirement is treated as a constraint on the lower-level entity. In these cases, the lower-level entity will then include an entity specific child requirement and trace that child requirement to its parent or source. The reason for doing so should be included in the rationale attribute for the need or requirement.

To continue our aircraft example above, although many requirements at the business management level of the ACME Aircraft Company will begin with "The ACME Aircraft Company shall …", the business may therefore wish to state at the business management level a requirement stating that all aircraft developed by the organization shall use an engine with specific characteristics. For a specific aircraft, child requirements will be written for the appropriate entity that implements the intent of the business management generic requirement. For the entity dealing with the engine specifically, the system level child requirement would begin "The Aircraft shall…."; and at the subsystem level the child requirement would begin "The Engine shall …" and trace back to the system level parent requirement, which, in turn, would trace back to the business management level constraint as the parent or source.

When talking about a quality or physical characteristic of an entity, some tend to write the requirements on the characteristic rather than the entity that has that characteristic which is not consistent with this rule nor rule and R3.

"The <entity characteristic> shall be <verb><value range>."

While this may be appropriate for a design output, it is not appropriate for a design input. In accordance with this rule, the subject of the requirement should be the entity to which the requirement applies, and the characteristic should be the object.

"The <entity> shall <verb><characteristic> <value range>."

Verb

As with the subject, the verb of a need statement or requirement statement must be appropriate to the subject of the need or requirement for the entity it is stated. For needs, the verbs such as "support", "process", "handle", "track", "manage", and "flag" may be appropriate. However, they are too vague for requirement statements which therefore may not be Unambiguous (C3) nor Verifiable/Validatable (C7).

For example, at the business management level the use of a verb such as "safe", may be acceptable if it is unambiguous at that level, decomposed at the lower levels, and is verifiable at those levels.

When transforming these need statements into requirement statements the functions referenced by these verbs would be decomposed into specific functions the <SOI> can be verified to perform at stated performance levels and conditions of operations.

Examples:

Subject examples*:*

Business management requirements have the form "The <business> shall …". For example, "ACME_Transport shall …".

Business operations requirements on business elements have the form "The <business element> shall …". For example, requirements on personnel roles have the form: "The <personnel role> shall …"—such as, "The Production_Manager shall …"; "The Marketing_Manager shall …".

System level needs have the form "The <stakeholders> need the system to ……"

System requirements have the form "The <SOI> shall ..."—for example, "The Aircraft shall …"

Subsystem level needs have the form "The <stakeholders> need the <subsystem> to ……"

Subsystem requirements have the form "The <subsystem> shall …" - for example, once the subsystems are defined: "The Engine shall …"; and "The Landing_Gear shall …".

Verb examples*:*

System level stakeholder need: "The <stakeholders> need the <SOI> to process data received from <other system>."

System level requirement: "The <SOI> shall [process] data received from <another system>. having the characteristics defined in <Interface Definition XYZ.>"

Through analysis, the verb/function "process" could be decomposed into sub functions such as "receive", "store", "calculate", "report", and "display".

Then a decision needs to be made regarding the specific subsystem or system element in which these sub functions are to be performed.

If more than one subsystem or system element has a role in performing any one of the sub functions, that requirement should be communicated at the system level and allocated to the applicable subsystems or system elements.

If a sub function is to be implemented by a single subsystem or system element, then the sub function requirement should be communicated at the subsystem or system element set of requirements and traced back to the higher-level requirement from which it was decomposed.

Unacceptable system requirement: "The User shall ..."

[As discussed in Section 1.8, this is unacceptable because the requirement should be on the system, not the user or operator of the system. This wording is often the result of writing requirements directly from user stories or resulting need statements, without doing the transformation of the use case or need into a requirement. Ask, what does the system have to do so that the need can be achieved?]

Improved: "The <SOI> shall …".

Unacceptable: The <SOI> shall display the Current_Time on the <Display Device> per <Display Standard xyz>.

[Again, "Current_Time" and "Display_Device" must be defined in the glossary or data dictionary. This statement is most likely unacceptable because the Display_Device is either a system element which is presumably at a level lower than the SOI (and is therefore not appropriate to this level), or the Display_Device is a system or system element outside the SOI (and therefore should be handled as an interface since requirements cannot be placed on it by this requirement set). The display standard would be developed by the organization that would define standards to be used when displaying all information by all applications the organization develops including fonts, font sizes, colors, spacing, brightness, human factors, etc.]

Improved: The <SOI> shall display the Current_Time per <Display Standard xyz>.

[This is improved because the action is appropriate to the level of the requirement set. Note that, before being an acceptable statement, appropriate condition and qualifying clauses must be added.]

4.1.4 R4 – Defined Terms

Definition:
Define all terms used within the need statement and requirement statement within an associated glossary and/or data dictionary.

Elaboration:

As systems become increasingly complex and software intensive, the ability to share data and information, including needs and requirements, across organizations both internal and external, is critical to project success. The definition and documentation of a common ontology by organizations and their projects is fundamental to the definition and documentation of a common ontology. This ontology includes the formal naming and definition of a set of terms, entities, data types, and properties as well as defining the relationships among these terms, entities, and data types that are fundamental to the project and organization of which the project is part.

Having a documented, common ontology and an associated glossary and/or data dictionary, helps ensure consistent use of this data and information across all system lifecycle process activities and work products as well as across various groups within and external to the project. This common ontology is key to tool interoperability and the ability to share sets of data and information, including needs and requirements.

Definitions of terms used within need statements and requirement statements must be agreed to, documented, and used consistently throughout the project and all SE artifacts developed during all lifecycle activities.

Most natural languages are rich with words having several synonyms, each with a subtly different meaning based on context.

For example, if a person states: "I need a cream for wrinkles." - what was the intended meaning of the word "for" – to "give", "remove", or "prevent"? If someone goes to a barber and states: "I need my hair shorter – I want it over my ears." – what was the intended meaning of the word "over" - "cover" or "above"?

In need statements and requirement statements, shades of meaning will most likely lead to ambiguity and to difficulty during verification and validation activities across the lifecycle. Define terms in some form of ontology, including a glossary, data dictionary, or similar artifact that allows the reader of a need statement or requirement statement to know exactly what the writer's intended meaning was when the word was chosen.

The meaning of a term should be the same every time the word is used no matter the work product, SE tool, or artifact being developed across all lifecycle stages.

A standard format should be agreed within the organization to make the use of glossary terms identifiable in the need statements and requirements statements; for example, glossary items may be capitalized and multiple words in single terms joined by an underscore (such as "Current_Time"). This is essential for consistency to avoid using the word with its general meaning without context. This is the convention used in the examples in this Guide. This standard should be implemented and enforced within all SE tools used by the project to help ensure consistency.

For cases where needs and requirements will be translated into a different language, it is helpful to develop a "translation matrix" where terms in the originating language are listed along with the acceptable term to be used in the target language such that the original intent is communicated. The use of this matrix will help ensure consistency in the translations when multiple people are involved in the translations over time.

Examples:

Unacceptable: The <SOI> shall display the current time as defined in <Display Standard xyz>.

[This is unacceptable because it is ambiguous. What is "current"? In which time zone? To what degree of accuracy? In which format?]

Improved: The <SOI> shall display the Current_Time as defined in <Display Standard xyz>.

[Note that "Current_Time" must then be defined in the glossary or data dictionary in terms of accuracy, format, time zone, and units. Further, appropriate condition and qualifying clauses must be included before the statement is acceptable. In addition, the organization can define a display standard that applies to all products developed by the organizations concerning how information is to be displayed (fonts, colors, size, spacing, human factors, etc.]

4.1.5 R5 – Definite Articles

Definition:

Use the definite article "the" rather than the indefinite article "a".

Elaboration:

The definite article is "*the*"; the indefinite article is "*a*."

When referring to entities, use of the indefinite article can lead to ambiguity. For example, if the need or requirement refers to "a user" it is unclear whether it means any user or one of the defined users for which the system has been designed.

This causes further confusion during verification and validation activities across the lifecycle—for example, babies are arguably users of baby food, but the system would fail if the test agency sought to verify or validate that a baby could order, receive, open, and serve (or even independently consume) baby food.

From a medical device perspective who is the user – nurse, doctor, or patient? Although the patient is the one receiving the medical device, it is the nurse or doctor that will order, receive, open, install, and monitor the device after installation. From a risk perspective it is the patient that is of the main concern.

On the other hand, if the requirement refers to "the User", the reference is explicitly to the nature of the user defined in the glossary—in the baby food example, the "User" is presumably the adult responsible for feeding the baby. In the medical device example, the specific intended user would be defined in the glossary to remove ambiguity as to the intended user reference in the need statement or requirement statement.

Examples:

Unacceptable: The <SOI> shall provide a time display.

[This is unacceptable for a requirement because it is ambiguous—is the intent to provide **a** device to display the time or to display **the** time? If a device to display the time, then this is not appropriate to level. If to display **a** time - will any time displayed do? Is a one-off display of the time satisfactory? The writer's intention was most likely that they wanted the system to display continuously the current time in a format that is readable, yet if the developer provided **a** constant display of "10:00 am" (or even a one-off display of any time), they could argue (albeit unreasonably) that they have met the requirement; yet they would have clearly failed to meet the customer's need and intent.

Note that, as a need statement: "The <stakeholders> need the <SOI> to display the time.", the statement is arguably acceptable. However, as part of the transformation process, the requirement writers must remove the ambiguity - resulting in the following requirement statement (amongst others).

Improved*:* The <SOI> shall display the Current_Time as defined in <display standard xyz>.

[Note that "Current_Time" must be defined in the glossary or data dictionary since there are a number of possible meanings and formats of the more-general term "current time." For completeness, condition and qualifying clauses must also be added before the requirement is acceptable. In addition, the organization can define a standard that applies to all products developed by the organizations concerning how information is to be displayed (fonts, colors, size, spacing, human factors, etc.]

Exceptions and relationships:

The purpose of this rule is to avoid the ambiguity that arises because "a" or "an" is tantamount to saying, "any one of". In some cases, however, the use of an indefinite article is not misleading. For instance, "… with an accuracy of less than 1 second" allows the phrase to read more naturally and there is no ambiguity because of the accuracy quoted.

4.1.6 R6 – Common Units of Measure

Definition:

When stating quantities, all numbers should have appropriate and consistent units of measure explicitly stated using a common measurement system in terms of the thing the number refers.

Elaboration:
When stating quantities, all numbers should have appropriate and consistent units of measure explicitly stated using a common measurement system in terms of the thing the number refers. There are three primary measurement systems: British Imperial, US, Metric. For temperatures, the following are most often used: Celsius, Fahrenheit, or Kelvin. Within a project, a common measurement system must be used consistently. For example, do not mix both US and metric units of measure within any of the project's artifacts. Define precisely and use consistently the same expression for the measurement unit—for example, use consistently either 'liter' or the abbreviation 'l'. Once the measurement system has been chosen for each system or system element, it is fundamental to continue preserving consistency by using the same measurement unit for each property-element pair. For instance, the length of the screw should always be in mm and the length of the chair always in cm; never mix them within the same property-element pair. A word of caution: When converting from one measurement system to another, state the resulting value with a number of significant digits that appropriately preserves the precision of the original number. See also R40. Appropriate use of units can sometimes be more difficult that it may seem at first glance. For example, it should be noted that "liter" is the US spelling of the unit "litre", so a choice has to be made as to which spelling to use. Further, "liter/litre" is not an SI unit although it is one of the units (such as hours and days) that is accepted for use in the context of SI units—the correct SI unit for volume is *cubic metres (m^3)*, noting again that a choice has to be made since "meter" is the US spelling of "metre". It is clear, even from this tiny portion of the measurement domain, that care must be taken to avoid ambiguity by defining the unit in the project glossary and then using the unit consistently throughout the set of statements.

Exceptions and relationships:
In rare cases, projects may require integration between systems (or system elements) using different units of measure (for example when integrating COTS from one measurement system into an SOI based on another). If this is unavoidable, artifacts must clearly indicate which units of measure are used and for which elements.
Examples:
Unacceptable: With power applied, The Circuit_Board shall <…> a temperature of less than 30 degrees. *[This is unacceptable because the units used are stated without indicating the specific measurement system used.]* Improved: With power applied, The Circuit_Board shall <…> a temperature of less than 30 degrees Celsius. Unacceptable: The <SOI> shall establish communications as defined in <ICD xyz> with at least 4 in less than or equal to 10 seconds. [This is unacceptable because the units used are incomplete. "4" of what?] Improved: The <SOI> shall <establish communications> as defined in <ICD xyz> with at least 4 satellites in less than or equal to 10 seconds

[Note that the phrase "establish communications" is probably acceptable at the business level when used in a need statement, but the phrase would need further elaboration at lower levels so that the need is decomposed into verifiable requirements relating to frequency, type of communications (voice? data?), quality, bandwidth, etc. which would typically be defined in an ICD that defines the characteristics of the communication signal to be established.]

Unacceptable:

"The Fuel_System shall have a maximum fuel volume of 60 l".

"When the Fuel_System detects the Fuel_Tank volume is less than 500 ml, the Fuel_System shall notify the operator within 1 second."

[This is unacceptable because the two requirements use different units of measure ("l" and "ml").

Improved:

"The Fuel_System shall have a maximum Fuel_Tank volume of 60 l ".

"When the Fuel_System detects the Fuel_Tank volume is less than 0.5 l, the Fuel_System shall notify the operator within 1 second."

[Now both requirements are written using the same units of measure(liters).]

Additional Acceptable: "The Fuel_System shall have a maximum Fuel_Tank volume of 60 l." "When the Signal_System detects the Fuel_Tank volume is less than 50 ml, the Signal_System shall light the Low-Fuel_Iindicator within 1 second."

[In this particular case, while the property ("Fuel_Tank volume") remains the same, the component/element is different ("Fuel_System" vs "Signal_System"). The measurement unit can vary because we are dealing with different component–property pairs.]

[Note that "Fuel_System" and "Fuel_Tank" must be defined in the glossary or data dictionary.]

4.1.7 R7 – Vague Terms

Definition:
Avoid the use of vague terms.

Elaboration:

Vague terms can lead to ambiguous, unverifiable needs and requirements where the true intent is not being communicated.

Avoid words that provide vague quantification, such as "some", "any", "allowable", "several", "many", "a lot of", "a few", "almost always", "very nearly", "nearly", "about", "close to", "almost", and "approximate",

Avoid vague adjectives such as "ancillary", "relevant", "routine", "common", "generic", "significant", "flexible", "expandable", "typical", "sufficient", "adequate", "appropriate", "efficient", "effective", "proficient", "reasonable" and "customary."

Adverbs qualify actions in some way and are particularly troublesome if vague. Avoid vague adverbs, such as "usually", "approximately", "sufficiently", and "typically", which can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations.

As a general rule, words that end in "-ly" often result in ambiguity.

Examples:
Unacceptable: The <SOI> shall usually be online. [This is unacceptable because "usually" is ambiguous - is availability what is meant?] Improved: The <SOI> shall have an Availability of greater than xx% over a period of greater than yyyy hours. [Note that "Availability" must be defined in the glossary or data dictionary since there are a number of possible ways of calculating that measure. Alternately, the availability could be expressed as a need. Then the organization responsible for transforming the need into a requirement could develop feasible concepts for meeting the needed availability and derived one or more well-formed requirements that result in the need being met.] Unacceptable: The Flight_Information_System shall display per <Display Standard xyz> the Tracking_Information for relevant aircraft within <xxxx seconds> of detection. [This is unacceptable because it does not make explicit which aircraft are relevant. Additionally, the statement allows the developer to decide what is relevant; such decisions are in the province of the customer, who should make the requirement explicit.] Improved: The Flight_Information_System shall display per <Display Standard xyz> the Tracking_Information of each Aircraft located less than or equal to 20 kilometers from the Airfield when in the Operations_Mode within <xxxx seconds> of detection. [Now it is clear for which aircraft the information needs to be displayed. Note that "Aircraft", "Tracking_Information", "Airfield", and "Operations_Mode" must be defined in the glossary or data dictionary.]

Exceptions and relationships:
R3 points out that the use of a verb such as "safe", may be acceptable at the business management or operations level as long as it is unambiguous at that level, decomposed at the lower levels, and is verifiable at the level stated. Similarly, some vague adjectives may be allowable at the business management or operations level, providing they are not ambiguous at that level. NLP/AI tools providing automatic assessment of this rule shall be flexible enough and tailorable in order not to identify this issue as an error at a business level, while enforcing the absence of vague terms in other lower-level documents.

4.1.8 R8 – Escape Clauses

Definition:
Avoid the inclusion of escape clauses that state vague conditions or possibilities, such as "so far as is possible", "as little as possible", "where possible", "as much as possible", "if it should prove necessary", "if necessary", "to the extent necessary", "as appropriate", "as required", "to the extent practical", and "if practicable".

Elaboration:
Escape clauses give an excuse to the developer of the system at lower levels not to implement a need or requirement. From a contracting standpoint, needs or requirements with these phrases could therefore be interpreted as being optional even if communicated in a "shall" requirement statement.

Escape clauses can lead to ambiguous needs that the SOI cannot be validated to meet and are open to interpretation and that do not reflect accurately lifecycle concepts, or other sources, from which they were transformed.

Escape clauses can lead to ambiguous, unverifiable requirements that are open to interpretation and that do not reflect accurately the needs, source, or higher-level requirements from which they were transformed.

Examples:

Unacceptable: The GPS shall, *where there is sufficient space*, display the User_Location in accordance with <Display Standard xyz>.

[This is unacceptable because whether there is sufficient space is vague, ambiguous, and unverifiable. The requirement is clearer without the escape clause.]

Improved: The GPS shall display the User_Location in accordance with <Display Standard XYZ>.

[Note that appropriate performance measures must also be specified such as within what time, format, and accuracy. Also note that "GPS" and "User_Location" must be defined in the glossary or data dictionary.]

4.1.9 R9 – Open-Ended Clauses

Definition:

Avoid open-ended, non-specific clauses such as "including but not limited to", "etc." and "and so on".

Elaboration:

Open-ended clauses imply there is more required without stating exactly what.

Open-ended clauses can lead to ambiguous, unverifiable needs and requirements that do not reflect accurately the stakeholder's expectations and needs and can create ambiguity in the mind of the reader.

Needs or requirements with open-ended clauses are not Complete (C4).

Use of open-ended clauses also violates the one-thought rule (R18) that leads to the singular characteristic. If more cases are required, then include additional needs and requirements that explicitly state those cases.

Depending on the contract type (fixed price versus level of effort or cost plus) open-ended requirements can lead to serious interpretation problems concerning what is in or out of scope of the contract; possibly resulting in expensive contract changes. For level of effort or cost-plus contracts, open-ended requirements can be used by the supplier to do and bill the customer for additional work not intended by the customer leading to budget overruns and expensive contract changes.

Examples:

Unacceptable: The ATM shall display the Customer Account_Number, Account_Balance, and so on per <Display Standard xyz>.

[This is unacceptable because it contains an opened list of what is to be displayed.]

Improved: *[Split into as many requirements as necessary to be complete. Note that the types of customer information to be displayed needs to be defined in the glossary.]:*

The ATM shall display the Customer Account_Number in accordance with <Display Standard xyz>.
The ATM shall display the Customer Account_Balance in accordance with <Display Standard xyz>.
The ATM shall display the Customer Account_Type in accordance with <Display Standard xyz>.
The ATM shall display the Customer Account_Overdraft_Limit in accordance with <Display Standard xyz>.

[Note: Some may feel that a bulleted list or table is acceptable. While, from a readability perspective, this may be true, from a requirement management perspective, each item in the bulleted list is still a requirement. Because of this requirement statements as in the example above should be written as individual statements especially when allocation, traceability, verification, and validation activities are specific to the single item. See also R17, R18, R22, and R28.]

[Note the above examples are incomplete in that expected performance and under what conditions has not been included in the requirement statements.]

4.2 Concision

4.2.1 R10 – Superfluous Infinitives

Definition:

Avoid the use of superfluous infinitives such as “to be designed to”, “to be able to”, “to be capable of”, “to enable”, “to allow”.

Elaboration:

An **infinitive** is a verbal consisting of the word “to” + “a verb”.

Use of infinitives within requirement statements:

When writing a requirement statement, using more verbs than necessary to describe a single action (verb), such as “The <SOI> shall be designed to be able to <do an action> ...” or “The <SOI> shall be designed to be capable of <action>...” rather than simply “The <SOI> shall <action>...”.

By forcing conformance with one of the agreed structured statements (See R1), the use of such superfluous infinitives is avoided.

Note that at the enterprise and business levels, requirements for an entity to “provide a capability” are acceptable. Where capability is made up of people, processes, and products; these requirements will be decomposed to address the people aspects (skill set, training, roles, etc.), processes (procedures, work instructions, etc.); and products (hardware and software systems). The enterprise and business level higher-level requirements will be allocated appropriately to the people, processes, and products (SOI), as appropriate.

When the resulting requirement sets are defined for all three areas, the capability will be communicated by a set of requirements for each (people, process, product) that will result in the needed capability to be provided.

The use of "be able to" or "have the capability to" is also ambiguous in terms of defining the success criteria for system verification. For example, if the SOI is tested 100 times, and is only successful once, the case could be made that it is "able to" or is "capable of", but that isolated performance is probably not acceptable to the customer.

As another example, the SOI may be capable of a given function provided that the customer buys an additional capability. In this case the SOI could be designed to be expandable to allow additional capabilities to be added. Again, although the SOI is capable of a given feature if additional functionality is acquired, that is probably not what the customer desires.

If the intent of wording such as "be able to" or "be capable of" is to communicate the system will only need to do the required action based on some condition, trigger, or state, then the state the condition, trigger, or state must be included as part of the requirement (see R1 and R11).

Use of infinitives within need statements:

The use of "to be able to", "to have the capability to", "to enable the user to", or "to allow the user to" may be acceptable within a need statement. For example: "The <stakeholders > need the system "to provide the capability for users to" <do some action>."

Those transforming the need statements into requirements will define a feasible concept that will result in the system being able to provide a capability or allow or enable an action. As part of the transformation process from need to one or more requirements, the requirement writer will determine what the SOI is required to do to provide that capability or allow or enable the action.

When using "allow" or "enable' in a need statement it is important to understand how the terms are interpreted and the intent of the resulting actions. "Allow" can be interpreted as "to not prohibit something, to let it happen, to remove any constraint that would prevent something from happening" and "enable" can be interpreted as "to make something possible". Understanding this distinction is important when deriving the requirements that will meet the intent of the need.

Examples:

Unacceptable: The Weapon_System shall *be able to* store the location of each Ordnance.

[This is unacceptable because it contains the superfluous infinitive "be able to" which implies the requirement is ubiquitous in that it can be applied at any time, but it therefore begs the question as to the condition under which the feature can be invoked. However, this is acceptable for the stakeholder need: "The stakeholders need the Weapon_System to be able to store the location of each Ordnance."]

Improved: When <condition>he Weapon_System shall store the Location of each Ordnance.

[This is better because the requirement applies only in a particular condition. Note that the terms "Weapon_System", "Location", and "Ordnance" must be defined in the glossary or data dictionary. To be complete the appropriate performance characteristics would also need to be defined and included within the requirement statement.]

4.2.2 R11 – Separate Clauses

Definition:

Use a separate clause for each condition or qualification.

Elaboration:

Each need or requirement should have a main verb describing a basic function or need. If appropriate, the main sentence may then be supplemented by clauses that provide conditions or qualifications (performance values, trigger, or constraints). A single, clearly identifiable clause should be used for each condition or qualification expressed.

As mentioned in R1 and Appendix C, a need and requirement should match one, and only one, pattern from the catalog of agreed patterns.

If an applicable qualifying clause or condition is not stated explicitly within the need or requirement statement, the need or requirement statement is not Complete (C4), Verifiable/Validatable (C7), nor Correct (C8).

If a qualifying clause is not stated explicitly within the need or requirement statement, the need or requirement statement is not Complete (C4)), Verifiable/Validatable (C7), nor Correct (C8)—for example, performance associated with the action verb or for an interface requirement where a pointer to where the specific interaction is defined (such as in the ICD).

When using clauses, make sure the clause does not separate the object of the sentence from the verb. See also R1, R18, R27 and Appendix C.

Examples:

Unacceptable: The Navigation_Beacon shall provide Augmentation_Data having the characteristics defined in <ICD xyz> at an accuracy of less than or equal to 20 meters to each Maritime_User during Harbor_Harbor_Approach_Maneuvering (HHAM).

[This is unacceptable because it inserts a phrase in such a way that the object of the sentence is separated from the verb.]

Improved: The Navigation_Beacon shall provide Augmentation_Data having the characteristics defined in <ICD xyz> to each Maritime_User engaged in Harbor_Harbor_Approach_Maneuvering (HHAM), at an accuracy of less than or equal to 20 meters.

[This rewrite places the basic function in an unbroken clause followed by the sub-clause describing performance. Note that: “Navigation_Beacon”, “Maritime_User”, and “Harbor_Harbor_Approach_Maneuvering (HHAM)” must be defined in the glossary or data dictionary. There is also an issue in that when discussing accuracy, precision needs to also be addressed.]

4.3 Non-ambiguity

4.3.1 R12 – Correct Grammar

Definition:

Use correct grammar.

Elaboration:

We interpret language based on the rules of grammar. Incorrect grammar leads to ambiguity and clouds understanding. This is especially true when the recipient of the need statement or requirement statement is working in a second language relying on specific rules of grammar. If these rules are not followed, that person may misinterpret the meaning of the need statement or requirement statement.

Incorrect use of grammar may make the true intent unclear resulting in an incorrect requirement and thus make it difficult to verify the SOI meets the intent of the requirement.

Care must be taken when translating need statements and requirement statements from one language to another and when sentence structure differs depending on the language in which the original need or requirement statement was written. Punctuation varies from language to language and even between dialects of a given language.

Be cautious when need statements and requirement statements must be translated. An interesting exercise is to translate a requirement statement from one language to another and translate the result back to the original language.

Many word processing applications are capable of applying a set of grammar rules based on a set of built-in rules that are selectable. When using an application specific set of grammar rules, project-specific grammar items may be flagged as incorrect as they may not be applicable to the project's use of grammar. Therefore, the application grammar checking can be regarded as a project-specific aspect. Many applications allow the user to "turn off" or "ignore" grammar rules that do not apply to the project.

See also R4 - Define terms.

Examples:

Unacceptable: The Weapon_System shall *storing* the location of all ordnance.

[This is unacceptable because the grammatical error leads to uncertainty about the meaning.]

Improved: The Weapon_System shall store the location of all Ordnance.

[Note that "Ordnance" must be defined in the glossary to be explicit about the types of weapons and ammunition. Also, where the location is to be stored and the format of the data, must be addressed. The action "store" needs to be further evaluated to determine if that is an appropriate action (verb) for the Weapon_System vs a user interacting with the Weapon_System as discussed in R3. Additionally, to be complete any specific performance measures and conditions should be included within the requirement statement.]

Unacceptable: When in the Active_State, the Record_Subsystem shall display each of the Names of the Line_Items, without obscuring the User_ID per <Display Standard xyz>.

[This is unacceptable because the grammatical error involving the inappropriate placement of "each of"—it is most likely that a Line_Item has only one name.]

Improved: When in the Active_State, the Record_Subsystem shall display on <Display Device> the Name of each Line_Item, without obscuring the User_ID per <Display Standard XYZ.>.

[This is acceptable the ambiguity has been addressed. The requirement is now more complete in that it references where the information is to be display and the standard to be used concerning the display of the information. If there are any performance measures that apply, they would also need to be addressed.]

Unacceptable: The <corporate website> shall only use Approved_Fonts.

[This is unacceptable because it mandates that the website shall *only* use the designated fonts—that is, it is not expected to perform any other function except to use those fonts. This is clearly not what is meant but becomes ambiguous by the inappropriate grammar and the incorrect placement of the word "only". What is most likely meant is that the only fonts to be used are the approved fonts defined in the organization's display standard.]

Improved: The <corporate website> shall display information using Approved_Fonts defined in <Display Standard xyz>.

[This is better because the ambiguity has been addressed. To be acceptable, any conditions and qualifying clauses would also need to be added. If the organization has a standard for displaying information that addresses acceptable fonts, font sizes, colors, spacing, human factors, etc., the requirement should refer to that standard.]

4.3.2 R13 – Correct Spelling

Definition:

Use correct spelling.

Elaboration:

Incorrect spelling can lead to ambiguity and confusion. Some words may sound the same but, depending on the spelling, will have entirely different meaning. For example, “red” versus “read”, “ordinance” versus “ordnance”, or “brake” versus “break”.

In other cases, the word could be spelled the same, but have a different meaning or the meaning changes depending on the context of which it is used. For example, "clear windscreen" and "clear the screen” contain 2 different meanings for "clear", one is an adjective, and the other a verb. *In addition, the word “sound” could be ambiguous as it can be a noun, a verb, adverb, or adjective.* In these cases, a spell checker cannot distinguish the meaning nor context not finding these kinds of errors.

A requirement that has spelling errors is not correct (C8) and may not be Verifiable/Validatable (C7).

In addition to misspelling, this rule also refers to the proper use of:

- Capital letters in acronyms: avoid “SYRD” and “SyRD” in the same set of needs and requirements.
- Capital letters in other non-acronyms concepts: avoid “Requirements Working Group” and “Requirements working group” in the same set of needs and requirements.
- Proper use of hyphenation: “non-functional” versus “nonfunctional.” Often hyphenation is used when two related words are used as adjectives but is not used when used as a noun.
- Proper form for compound words. Compound words are when two or more words combine to form a new single word or a phrase that acts like a single word. There are three different types of compound words in grammar: open compound words with spaces between the words (ice cream), closed compound words with no spaces (firefighter), and hyphenated compound words (up to date). The project team needs to agree on how specific compound words are to be formed, define them in the project glossary, and be consistent in their use.

Many word processing applications are capable of doing a spelling check as text is entered based on a built-in dictionary and a user defined dictionary. When using an application specific spelling dictionary, project-specific vocabulary items or terms may be flagged as misspelled words as they may not be included in the application’s internal core dictionary. Therefore, the spellchecking can be regarded as a project-specific aspect. If practical, these terms should be added to a user defined dictionary within the application, if allowed.

Examples:

Unacceptable: The Weapon_System shall store the location of each *ordinance*.

[This is unacceptable because the word “ordinance” means regulation or law. It is unlikely that the Weapon_System is interested in the location of ordinance (regulations). In the context of a weapon system, what the authors meant to use is "ordnance" as in weapons and ammunition, not “ordinance”.]

Improved: The Weapon_System shall store the Location of each Ordnance.

[Note that “Location” and “Ordnance” must be defined in the glossary or data dictionary to be explicit about the types of weapons and ammunition. The action “store” needs to be further evaluated to determine if that is an appropriate action (verb) for the Weapon_System vs a user interacting with the Weapon_System as discussed in R3. Additionally, to be complete any specific performance measures and conditions should be included within the requirement statement.]

4.3.3 R14 – Correct Punctuation

Definition:
Use correct punctuation.

Elaboration:

Incorrect punctuation can cause confusion between sub-clauses in a need or requirement statement.

A need or requirement statement with incorrect punctuation is not Correct (C8)

Note also that the more punctuation in a need or requirement statement, the greater the opportunity for ambiguity.

Many word processing applications are capable of applying a set of punctuation rules as text is entered based on a set of built-in rules that are selectable. When using an application specific set of punctuation rules, project-specific punctuation items may be flagged as incorrect as they may not be applicable to the project’s use of punctuation. Therefore, the application punctuation checking can be regarded as a project-specific aspect. Many applications allow the user to “turn off” or “ignore” punctuation rules that do not apply to the project.

Examples:

Unacceptable: The Navigation_Beacon shall provide Augmentation_Data having the characteristics defined in <ICD xyz> to each Maritime_User, engaged in Harbor_Harbor_Approach_Maneuvering (HHA) at an accuracy of less than 20 meters.

[This is unacceptable because the incorrectly placed comma in this sentence confuses the meaning, leading the reader to believe that the accuracy is related to the maneuver rather than to the augmentation data.]

Improved: The Navigation_Beacon shall provide Navigation_Data having the characteristics defined in <ICD xyz> to each Maritime_User engaged in Harbor_Harbor_Approach_Maneuvering (HHA), at an accuracy of less than 20 meters.

[The positioning of the comma now makes it clear that the accuracy and availability relate to the data. Also note that, while performance measures are included, the appropriate conditions are not addressed. Further “Navigation_Beacon”, “Navigation_Data”, “Maritime_User”, and “Harbor_Harbor_Approach_Maneuvering (HHA)” must be defined in the glossary or data dictionary.]

4.3.4 R15 – Logical Expressions

Definition:
Use a defined convention to express logical expressions such as "[X AND Y]", "[X OR Y]", [X XOR Y]", "NOT [X OR Y]".

Elaboration:

As with the other rules and characteristics, we want to keep requirement statements as one thought with singular statements. Thus, we avoid using "and" when it involves tying two thoughts together. However, it is acceptable to use "AND", "OR", "XOR", and "NOT" in a logical sense when talking about conditions to which the verb applies. All logical expressions decompose to either "true" or "false", resulting in a singular statement.

Examples of conventions:

1. Place conjunctions in italics or in all capitals (AND, OR, XOR, NOT) to indicate that the author intends the conjunction to play a role in a condition.
2. Place conditions within square brackets, also using the brackets to control their scope. For example, "[X AND Y]."

Further, use of "and/or" is non-specific and therefore ambiguous. The most common interpretation of the expression "and/or" is as an inclusive OR: either X OR Y OR both.

- If an inclusive OR is intended, that should be written as "at least one of <the two or more requirements>".
- If an Exclusive OR is intended, that should be written as "Either <Requirement 1> OR <Requirement 2> but NOT both", and similar wording.

Note that caution should be used when including logical expressions in requirement statements. In many cases the use of logical expressions is more appropriate to design output specifications/requirements. Design Input Requirements should focus on why the logical actions are needed - prevent something bad from happening, for example,

The use of logical expressions within requirement statements is appropriate when stating "under what conditions" apply to a need or requirement. For example, an action that must take place based on whether at logical condition is true or false.

Also see R19 and R20.

Examples:

Unacceptable: The Engine_Management_System shall disengage the Speed_Control_Subsystem within <TBD seconds> when the Cruise_Control is engaged, and the Driver applies the Accelerator.

[This is unacceptable because of the ambiguity of "and" could be confused with combining two separate thoughts. Instead use the form of a logical expression [X AND Y].]

Improved: When [the Cruise_Control is Engaged] AND [the Accelerator is Applied], the Engine_Management_System shall Disengage the Speed_Control_Subsystem within <TBD seconds>.

["Engine_Management_System", "Speed_Control_Subsystem", "Disengage", "Engaged", "Accelerator", "Applied", and "Cruise_Control" must be defined in the glossary or data dictionary.]

Exceptions and relationships:

While R21: Avoid Parentheses states that parentheses or brackets are to be avoided within general sentence structure, this rule suggests that brackets may be used as part of a convention to avoid ambiguity when expressing a logical expression.

4.3.5 R16 – Use of "Not"

Definition:

Avoid the use of the word "not".

Elaboration:

The presence of the word "not" in a need statement or requirement statement implies "not ever", which is impossible to verify in a finite time, in which case, the need statement or requirement statement is not correct (C8).

In theory, there is a large number of actions the system should not do. Such statements should be re-written in the positive—that is, referring to what the entity is to do, rather than what it is not to do.

Rewriting the need statement or requirement statement to avoid the use of "not" results in a need o statement requirement statement that is clearer and is verifiable/validatable (C7).

Examples:

Unacceptable: The <SOI> shall not fail.

[This is unacceptable because verification of the requirement would require infinite time. The requirement is also infeasible in that, as written, the implication is to never fail, under any conditions.]

Improved:

The <SOI> shall have an Availability of greater than or equal to 95%. or

The <SOI> shall have a Mean Time Between Failures (MTBF) of xx operating hours.

[For quality requirements, it would be more precise in the above were written as need statements. Then those transforming the quality need statements into requirement statements would define feasible concepts that would result in the needed quality attributes and derive well-formed requirements on the <SOI> that would result in those needs to be met.]

Unacceptable: The <SOI> shall not contain mercury.

[This is unacceptable because verification of the requirement would require the ability to measure the amount of mercury with infinite accuracy and precision. In addition, the real requirement may not be stated, for example, the real concern may be the use of toxic materials, not just mercury. If that is the case, it may be best to reference a standard from a governmental agency concerning allowable exposures to a list of common toxic materials.]

Improved: The <SOI> shall limit metallic mercury exposure to those coming in contact with the <SOI> to less than or equal 0.025 mg/m^3 over a period of 8 hours.

Exceptions and relationships:

It may be reasonable to include "not" in a requirement when the logical "NOT" is implied—for example when using not [X or Y]. In that case, however, in accordance with R15, it may be better to capitalize the "NOT" to make the logical condition explicit: NOT [X or Y].

There may be other cases such as “The <SOI> shall not be red in color.”, which is stating a constraint and is verifiable, as long as the range of shades of red is stated (RBG rr,bb,gg range or a “name” of red in some standard).

The key consideration is verification. If the “not” can be unambiguously verified, then its use is acceptable.

4.3.6 R17 – Use of Oblique Symbol

Definition:

Avoid the use of the oblique (“/”) symbol.

Elaboration:

The oblique symbol (“/”), or “slash”, has so many possible meanings that it should be avoided.

The slash symbol (such as in “user/operator”, “budget/schedule” or the construct “and/or” (discussed in R15) can lead to ambiguous statements that do not reflect accurately the true stakeholder needs or lifecycle concepts from which the needs in the Integrated Set of Needs were derived. Also see R19.

Exceptions and relationships:

Exceptions to this rule include where the oblique symbol is used in units (for example “km/h”) or when communicating a symmetrical range of a value (for example +/- 5 degrees F).

The oblique symbol may also be used when expressing ratios or fractions (such as 1/16)—see R40.

Examples:

Unacceptable: The User_Management_System shall Open/Close the User_Account in less than 1 second.

[This is unacceptable because it is unclear as to what is meant by open/close: open, close, or both?]

Improved: (Split into two requirements with an appropriate condition)

When <condition>, the User_Management_System shall Open the User_Account in less than 1 second.

When <condition>, the User_Management_System shall Close the User_Account in less than 1 second.

Unacceptable: When the Clutch is Disengaged and/or the Brake is Applied, the Engine_Management_System shall disengage the Speed_Control_Subsystem within <XYZ ms>.

[This is unacceptable because of the use of “and/or.” If simultaneity is intended—that is the dual conditions must be met at the same time—write the requirements as a logical AND. If “and” is meant in the sense that the action is to be completed under each of the conditions, split the two thoughts into separate requirements, one for each condition. If “or” is meant, write the requirement as a logical OR.]

Improved: (As one requirement if simultaneity is intended):

When [the Clutch is Disengaged] AND [the Brake is Applied], the Engine_Management_System shall disengage the Speed_Control_Subsystem within <XYZ ms>.

Improved: (As two requirements if two separate conditions are intended):

> When the Clutch is Disengaged, the Engine_Management_System shall disengage the Speed_Control_Subsystem within <XYZ ms>.
>
> When the Brake is Applied, the Engine_Management_System shall disengage the Speed_Control_Subsystem within <XYZ ms>.

Improved: (As one requirement if inclusive OR is intended)

> When EITHER [the Clutch is Disengaged] OR [the Brake is Applied], the Engine_Management_System shall disengage the Speed_Control_Subsystem.

4.4 Singularity

4.4.1 R18 – Single Thought Sentence

Definition:

Write a single sentence that contains a single thought conditioned and qualified by relevant sub-clauses.

Elaboration:

Need statements and requirement statements (based on the concepts of allocation, traceability, validation, and verification) must contain a single thought allowing:

- needs to be traced to their source;
- the single thought within a requirement statement to be allocated;
- the resulting single-thought child requirements to trace to their allocated parent,
- requirements to trace to a single-thought source;
- design and system validation and design and system verification against the single-thought need or requirement.

Sometimes a need statement or requirement statement is only applicable under a specific trigger, condition, or multiple conditions as discussed in Section 1.11.

If multiple actions are needed for a single condition, each action should be repeated in the text of a separate need statement or requirement statement along with the triggering condition, rather than stating the condition and then listing the multiple actions to be taken. Using this convention, the system can be verified to perform each action, and each action can be separately allocated to the entities at the next level of the architecture.

Also avoid stating the condition or trigger for an action in a separate sentence. Instead write a simple affirmative declarative sentence with a single subject, a single main action verb and a single object, framed and qualified by one or more sub-clauses.

Avoid compound sentences containing more than one subject/verb/object sequence. This constraint is enforced in the catalog of agreed patterns (see R1 and Appendix C).

Often when there are multiple sentences for one requirement, the writer is using the second sentence to communicate the conditions for use or rationale for the requirement for the first sentence. This practice is not acceptable—rather include rationale in the attribute A1 - *Rationale* as part of the requirement expression and include the condition of use within the need statement or requirement statement or an attribute within the need or requirement expression.

See also R1, R11, R27, and R28.

Examples:

Unacceptable: When in the Active_State, the Record_Subsystem shall display the Name of each Line_Item and shall record the Location of each Line_Item, without obscuring the User_ID.

[This is unacceptable because the sentence contains two requirements and the qualification only applies to the first requirement, not the second.]

Improved: (Split into two separate requirements)

When in the Active_State, the Record_Subsystem shall display per <Display Standard XYZ> the Name of each Line_Item, without obscuring the User_ID.

When in the Active_ State, the Record_Subsystem shall record the Location of each Line_Item.

[Note use of the glossary or data dictionary to define terms. Any applicable performance measures must be addressed for the requirement to be complete.]

Unacceptable: The Control_Subsystem will close the Inlet_Valve until the temperature has reduced to 85 °C, when it will then reopen it in less than 1 second.

[This is unacceptable because the sentence contains two event driven requirements. Additionally, the sentence contains two occurrences of the pronoun "it" ambiguously referring to different things (see R26), the term "has reduced" is ambiguous, and the action verb must be "shall", not "will". Also, a time constraint is included but it is not clear whether it applies to both actions or only the second action.]

Improved: (Split into two requirements each with its own time constraint):

If the Water_Temperature in the Boiler increases to greater than 85 °C, the Control_Subsystem shall close the Inlet_Valve in less than 1 second.

When the Water_Temperature in the Boiler reduces to less than or equal to 85 °C, the Control_Subsystem shall open the Inlet_Valve in less than 1 second.

[Note use of the glossary or data dictionary to define terms.]

Unacceptable:

In the event of a fire detection, within <xxx ms>:

- Security entrances shall be set to Free_Access_Mode.
- Fire escape doors shall be unlocked.

[This is unacceptable because the condition "in the event of a fire detection" is stated following a list of actions to be taken; each of which the system must be verified against. Also, note the actions are written in passive voice which violates R2. Each action needs to be communicated in a separate active voice requirement statement. For multiple conditions for a single action see R28.]

Improved: (Split into two requirements)

In the event of Fire_Detection, the Fire_Control_Subsystem shall set the Free_Access_Mode to ON within <xxx ms>.

In the event of Fire_Detection, the Fire_Control_Subsystem shall send the Fire_Door_Unlock unlock command having the characteristics defined in <ICD xyz> to each Fire_Door within <xxx ms>.

[There is a Fire Detection subsystem responsible for detecting fires and setting the Fire_Detection parameter to "TRUE" which triggers the two actions. All other subsystems that are monitoring when the Fire_Control_Subsystem has set the Free_Access_Mode to "TRUE" will take the required action as stated in their set of requirements. All Fire Doors will be monitoring for the Fire_Door_Unlock command and will unlock when the command is received.]

Unacceptable: The <corporate website> shall use only approved fonts. The approved fonts are: <Font1>, <Font2> and <Font3>.

[This is unacceptable because it is non-singular. Rather than splitting into three requirements (which is quite difficult to so whilst conforming to the "only" constraint, the term "Approved_Fonts" can be included in the Glossary, which then states: "*Approved_Fonts: <Font1>, <Font2> and <Font3>.*"]

Improved: (using the glossary term "Approved_Fonts")

The <corporate website> shall use only Approved_Fonts.

[Alternately, the organization can define a display standard which includes a definition of approved fonts, and the requirement could reference that standard.]

Improved: (referring to a standard.)

The <corporate website> shall display information in accordance with <Display Standard xyz>.

[Note that this also allows better configuration management of the requirements database, particularly if the approved fonts are referred to in a number of requirements—that is, should the approved fonts change or characteristics of the fonts change, the list of approved fonts and characteristics only needs to be updated in the Glossary, rather than in multiple requirements (in which case there is a risk that one or more of the requirements is missed).]

[In reality, requirements involving the actions such as "display" are interface requirements given that the information will be provided by something to some display device for display.]

[Having the information in the display standard then can be referred to in the set of requirements for that display device. Note that referring to a standard also allows for later specialization, for example different parts of the standard may apply to different display devices.]

Exceptions and relationships:

Every requirement should have a main clause with a main verb (R1). However, additional sub-clauses with auxiliary verbs or adverbs may be used to qualify the requirement with performance attributes.

Such sub-clauses cannot be verified in isolation since they are incomprehensible without the main clause. Sub-clauses that need to be verified separately from others should be expressed as separate requirements.

For example, "The Ambulance_Control_System shall communicate Incident_Details to the Driver" is a complete, comprehensible statement with a single main verb. An auxiliary clause may be added to provide a qualifying constraint "The Ambulance_Control_System shall communicate Incident_Details to the Driver, *while simultaneously maintaining communication with the Caller."*

[Note: the verb "communicate" is more appropriate for a need statement. The resulting Design Input Requirements would address the specific means of communication.]

Similarly, if the requirement is to extinguish and dispose of a match as a single combined action, the requirement must ensure that both are verified the same time and allocated the same.

Note that, if performance attributes need to be verified separately, they should be expressed as sub-clauses in separate requirements.

4.4.2 R19 – Combinators

Definition:

Avoid words that join or combine clauses, such as “and”, “or”, “then”, “unless”, “but”, “as well as” “but also”, “however”, “whether”, “meanwhile”, “whereas”, “on the other hand”, or “otherwise”.

Elaboration:

The presence of such combinators in a statement usually indicates that multiple thoughts are contained within the statement violating rule R18.

To address these instances, either a logical expression is warranted (rule R15) or the statement must be broken into separate need or requirement statements, each containing a single thought.

Examples:

Unacceptable: The user shall either be trusted or not trusted.

[This is unacceptable for several reasons. The intention is that a user should be classified in one of two ways, but it is also a passive requirement written on the user rather than on the system (R2) and it is ambiguous: the requirement would still be met if the system took the option of treating all users as trusted.]

Improved: The Security_System shall categorize each User as EITHER Trusted OR Not_Trusted.

Unacceptable: The <SOI> shall display the Location and Identity of the Lead_Vehicle.

[This is unacceptable because is in fact stating two requirements which may well need to be verified by different verification actions. In addition, the format of the displayed information is not referenced.]

Improved: (Express as two requirements)

The <SOI> shall display the Location of the Lead_Vehicle per <Display Standard xyx>.

The <SOI> shall display the Identity of the Lead_Vehicle per <Display Standard xyx>.

Unacceptable: The <SOI> shall provide an audible or visual alarm having the characteristics defined in <alarm standard xyz>.

[This is unacceptable because it is a moot point as to whether the designer can choose which alarm to implement, or (which is probably the intent of the writer) whether both alarms are required, and the operator is alerted by at least one.]

Improved: (if the choice is available to the designer)

The <SOI> shall provide EITHER an Audible_Alarm OR Visual_Alarm having the characteristics defined in <Alarm Standard xyz>.

Improved: (if both alarms are required)

The <SOI> shall provide an Audible_Alarm having the characteristics defined in <Alarm Standard xyz>.

The <SOI> shall provide a Visual_Alarm having the characteristics defined in <Alarm Standard xyz>.

[As separate requirements, each can be allocated differently, and the SOI will be verified to meet each.

Exceptions and relationships:

AND, OR, NOT can be used in need and requirement statements as logical conditions and qualifiers as stated in R15. See also R16 and R17.

Combinators such as in "Command_and_Control" can be used in the project data dictionary, ontology or glossary to cover one single function or quality of the system.

4.4.3 R20 – PURPOSE PHRASES

Definition:

Avoid phrases that indicate the "purpose of ", "intent of", or "reason for" the need statement or requirement statement.

Elaboration:

Need statements and requirement statements should be as concise as possible and be Complete (C4). In an attempt to avoid ambiguity or communicate why the need or requirement is Necessary (C1), some writers include additional information to make the purpose, intent, and reason clearer by including additional text within the need statement or requirement statement or include an additional sentence of explanation immediately following the need statement or requirement statement.

The text of a need statement or requirement statement should not include this extra text, nor should an additional sentence be included as part of the need or requirement statements.

Expressions of purpose or intent are often indicated by the presence of phrases such as "……to", "in order to", "so that", and "thus allowing."

This rule does not mean to imply that the additional information with regard to purpose is superfluous and should not be included at all. On the contrary, this extra information is useful to readers of a statement, but it should not be included in the statement; rather it should be included separately in the need expression or requirement expression using the Rationale attribute (A1) as discussed in the NRM.

Examples:

Unacceptable: The Record_Subsystem shall display the Name of each Line_Item so that the Operator can confirm that it is the correct Item.

[This is unacceptable because the text "so that the Operator can confirm that it is the correct Item" is rationale.]

Improved: The Record_Subsystem shall display the Name of each Line_Item per <Display Standard XYZ>.

[The text "so that the Operator can confirm that it is the correct Item" should be included in the rationale attribute.]

Unacceptable: The Operation_Logger shall record each Warning_Message produced by the system.

[This is unacceptable because of the superfluous words "produced by the system".]

Improved: The Operation_Logger shall record each Warning_Message within <performance measure>.

4.4.4 R21 – Parentheses

Definition:

Avoid parentheses and brackets containing subordinate text.

Elaboration:

If the text of a need statement or requirement statement contains parentheses or brackets, it usually indicates the presence of superfluous information that can simply be removed or can be communicated in the rationale. Other times, brackets introduce ambiguity.

If the information in the parentheses or brackets aids in the understanding of the intent of the requirement, then that information should be included in the Rationale Attribute (A1) defined in the NRM.

Conventions for the use of parentheses or brackets may be agreed upon for specific purposes. Such conventions should be documented in the project's requirements and work instructions concerning defining needs and requirements.

Examples:

Unacceptable: When the Water_Temperature exceeds 85 °C (usually towards the end of the boiling cycle), the Control_Unit shall disconnect power from the Boiler within <xxx ms>

[This is unacceptable because of the parenthetical phrase as well as the ambiguous word "usually". Note that the parameter Water_Temperature needs to be defined in the glossary in order to be specific]

Improved: If the Water_Temperature increases to greater than 85 °C, the Control_Unit shall disconnect power from the Boiler within <xxx ms>.

[Note: for the above example, if this behavior is the result of a design decision, then the requirement needs to be communicated as a design output. The design input requirement should focus on why this behavior is needed - for example, prevent the boiler from over pressurizing and exploding. This is a good example of the benefit of asking "Why" for requirements of this type. The answer is the real design input requirement. If included in the set of Design Input Requirements, then the "why" should be included in the rationale.]

Improved: If the Water_Temperature increases to greater than 85 °C, the Control_Unit shall prevent the Boiler from over pressurization.

[How this is done is a design decision.}

Exceptions and relationships:

While this rule states brackets are to be avoided, R15 indicates that brackets may be used as part of a convention for eliminating ambiguous conditions such as logical expressions. In that case, it is useful to settle on a combination to be used—that is, for example, the organization template may avoid round brackets "( )" in any statement and may use square brackets "[...]" in logical expressions.

4.4.5 R22 – Enumeration

Definition:

Enumerate sets explicitly instead of using a group noun to name the set.

Elaboration:

If a number of functions are implied, a need statement or requirement statement should be written for each.

The use of a group noun to combine functions or entities is often ambiguous because it leaves membership of that group in doubt.

Other issues include allocation, traceability, verification, and validation. As with the singularity characteristic C5, each member of the set could be allocated differently, have different child requirements, each of which must be individually verified and validated.

It is almost always best to list all the members of the set as separate needs or requirements. See exceptions and relationship discussion below.

See also R18.

Examples:

Unacceptable: The thermal control system shall manage temperature-related functions.

[This is unacceptable because there is ambiguity regarding which functions are to be managed as well as what is meant by "managed". The specific functions should be enumerated explicitly in separate requirement statements along with performance expectations and any applicable conditions, triggers, or constraints.]

Unacceptable: The thermal control system shall monitor system temperature.

This is unacceptable because there is ambiguity regarding the function "monitor". Similar to "managed" What are the expectations of the stakeholders – update the display of the system temperature, maintain the temperature within some range, or store the history of the system temperature? The specific subfunctions should be enumerated explicitly in separate requirement statements along with performance expectations and any applicable conditions, triggers, or constraints.]

Improved: (three separate requirements*):*

The Thermal_Control_System shall update the display of System_Temperature every 10 +/- 1 seconds.

The Thermal_Control_System shall maintain the System_Temperature between 95°C and 98°C.

The Thermal_Control_System shall store the history of System_Temperature.

[Note use of the glossary to define terms.]

Exceptions and relationships:

It is almost always better to enumerate members in a set, but the rule may be softened at the higher levels of abstraction if the resulting requirement is sufficiently unambiguous. For example, at the business management level the business may state "ACME Consulting shall manage

Human Relations (HR) functions centrally." or a system-level need statement may state: "The <stakeholders> need the <SOI> to manage HR functions centrally."

Although this raises the question of which HR functions are being referred to, it is not useful to include a long list at these levels of abstractions and it is often not necessary since the statement is sufficiently clear at the level at which it is stated. The detailed enumeration of functions will be undertaken at the business operations level, and the business management stakeholders should be comfortable that no function will be omitted as a result of not listing it at the business management level.

At the system level, need statements may include group nouns to name a set of entities or include a higher-level function that is appropriate to the level of abstraction being communicated within the need statement. During the transformation of the need statement into requirements, the project team would decompose the need into individual functional/performance requirements as well as define requirements concerning the word "centrally" and would then validate with the stakeholders that the resulting requirements, when realized, would meet the intent of the need statement.

Another exemption would be to use the glossary or data dictionary to explicitly elaborate the set. For example. "Account_Information" may be defined in the Glossary to include "Account_Name", "Account_Number", and "Account_Balance".

4.4.6 R23 – SUPPORTING DIAGRAM, MODEL, OR ICD

Definition:
When a need or requirement is related to complex behavior, refer to a supporting diagram, model, or ICD.

Elaboration:

Sometimes it can be difficult to express a complicated need or requirement in words, and it is better simply to refer to a supporting diagram or model.

An example is a requirement on voltage at turn on which involves a magnitude, rise time, overshoot, and dampening time along with tolerances. Stating all these values in the same requirement could be seen to violate our combiner and multiple thought rules.

Having a requirement that states: "Upon turn on, the <SOI> shall supply the initial voltage with the characteristics shown in Drawing xxxxx." It is much simpler for the drawing to show the magnitude, rise time, allowable overshoot, and dampening time. Stating each as a separate requirement is not applicable because all four conditions are part of the one action.

This approach may seem to violate the singularity rule in that the SOI will need to be verified against each value in the drawing or figure. Because of this, an organization may choose to address each value in a separate requirement with the requirement or rationale referring to the diagram or drawing for context.

When multiple characteristics for an entity involved in an interaction between two systems, these characteristics can be defined in an ICD which is referred to within the requirement statement for each entity involved in the interactions, for example electrical power passing across an interface boundary can have multiple characteristics that must be define. (See example below for interfaces).

When using figures and models, care must be taken to not make matters worse but using disparate modelling languages and different terms in figures from those used in the statements.

R36 applies here to ensure consistency in terms, not just across the sets of need and sets of requirements but also across all associated models, figures, and diagrams.

Examples:

Unacceptable: The Control_System shall close Valves A AND B within 5 seconds of the temperature exceeding 95 °C AND within 2 seconds of each other.

[This is unacceptable because of the confusing set of conditions. In this case what is meant will be clear if the requirement referred to a diagram for context.]

Improved: [Within 5 seconds of the temperature exceeding 95 °C] AND [within 2 seconds of each other as shown in Timing_Diagram_6], the Control_System shall close [Valve A AND Valve B].

[Note that this assumes, of course, that the timing diagram itself is not ambiguous. If it is not possible to remove the ambiguity by using a diagram, it may be better to split the requirement into two.]

[Note: for the above example, if this complex behavior is the result of a design decision, then the requirement needs to be communicated as a design output. The design input requirement should focus on why this behavior is needed.]

Interface Example:

An interface requirement may refer to the characteristics of the electrical power (current/voltage) being supplied by a system.

There can be multiple characteristics that need to be defined such that the systems receiving the power have a clear understanding of the characteristics of the power they are receiving. In this case it is common for the characteristics to be defined in an ICD which both the provider and receiver of the power can refer to in their Design Input Requirements. From the provider perspective the provider system will have to be verified that the power provided has all of these characteristics and the receiver will have to verify it can receive and operate on power having these same characteristics.

<System 1> shall provide power to <System 2> having the characteristics defined in <ICD xyz>. Table 123>.

<System 2> shall receive power from <System 1> having the characteristics defined in <ICD xyz>. Table 123>.

<System 2> shall operate on power having the characteristics defined in <ICD xyz>. Table 123>.

4.5 Completeness

4.5.1 R24 – Pronouns

Definition:

Avoid the use of personal and indefinite pronouns.

Elaboration:

Pronouns are words such as “it”, “this”, “that”, “he”, “she”, “they”, and “them.”

Repeat nouns in full instead of using pronouns to refer to nouns included in the need statement or requirement statement or other need statements or requirement statements.

When authoring stories, pronouns are a useful device for avoiding the repetition of nouns; but when writing need statements and requirement statements, pronouns are effectively cross-references to nouns included within the need statement or requirement statement or other need statements or requirement statements and, as such, are ambiguous and should be avoided.

When originally written, the noun that defines the pronoun may have preceded the pronoun previously used in the need statement or requirement statement or in a previous need statement or requirement statement.

However, as the set of needs or set of requirements mature, individual needs or requirements may be added, deleted, reordered, or regrouped, such that the defining need or requirement no longer precedes the need or requirement containing the pronoun. This is especially true when requirements are stored in a requirement management tool where they exist as single statements in a database that may not be in order. To avoid these problems, repeat the proper nouns rather than using a pronoun.

Indefinite pronouns are words such as "all", "another", "any", "anybody", "anything", "both", "each", "either", "every", "everybody", "everyone", "everything", "few", "many", "most", "much", "neither", "no one", "nobody", "none", "one", "several", "some", "somebody", "someone", "something", and "such."

Indefinite pronouns stand in for unnamed people or things, which makes their meaning subject to interpretation, ambiguous, and unverifiable.

See also R32 and R36.

Examples:

Unacceptable: The controller shall send the driver his itinerary for the day. It shall be delivered at least 8 hours prior to his Shift.

[This is unacceptable because the requirement is expressed as two sentences and the second sentence uses the pronouns "it" and "his."]

Improved: At least 8 hours prior to the Driver_Shift, the Controller shall send the Driver_Itinerary for the day to the Driver.

[Note use of the glossary to define terms and to be explicit about the relationship between the driver, shift, and the itinerary for that particular driver.]

4.5.2 R25 – Headings

Definition:

Avoid relying on headings to support explanation or understanding of the need or requirement.

Elaboration:

It is a common mistake in document-centric documentation of needs and requirements to use the heading for a specific topic or subject under which the need or requirement applies to contribute to the explanation of the need or requirement. The need expression or requirement expression should be complete in and of itself and not require the heading for the intent of the expression to be clearly understood.

Use of a heading can be avoided when using a data-centric approach to SE where sets of needs and requirements are developed and managed using a modern RMT or other SE tool rather than a legacy tool that is document-centric which organizes needs and requirements within sections and paragraphs of a document.

Examples:

Example heading: "4.0 Alert Buzzer Requirements."

Unacceptable: 4.1 The <SOI> shall sound it for greater than 20 minutes.

[This is unacceptable because the requirement uses the pronoun "it" (R24), which requires the heading to understand what "it" means. In addition, the word "sound" could be ambiguous as it can be a noun, a verb, adverb, or adjective. We do not know whether Alert_Buzzer is a subsystem that controls the noise or the thing that makes the noise.]

Improved: When <triggering event>, the <SOI> shall sound an alert having the characteristics defined in <Alert Standard xyz> for greater than 20 minutes.

[In this improved rewrite, the existence of the header is not needed to interpret the requirement. Even though the above heading referenced an "Alert Buzzer"—an implementation, the improved requirement avoids implementation to a specific device, and rather indicates that when the triggering event happens, an alert is required to be sounded having the characteristics defined in the alert standard.

Exceptions and relationships:

The use of headings is useful and acceptable when producing an output of an RMT, grouping like types or categories of needs or requirements. However, care must be taken that the headings are used solely for management of the subsets of statements—the heading must not be necessary to understand any of the grouped statements. See R41 and R42.

4.6 Realism

4.6.1 R26 – Absolutes

Definition:

Avoid using unachievable absolutes.

Elaboration:

An absolute, such as "100% availability", is not achievable. Think ahead to design and system verification and design and system validation: how 100% availability be proven? Even if such a system could be built, could the project afford to validate, or it meets the need or requirement?

Other examples to avoid are "all", "every", "always", and "never" since such absolutes are impossible to verify without an infinite number of verification or validation activities.

A need or requirement that contains absolutes is neither Feasible (C6), Verifiable/Validatable (C7), nor Correct (C8).

Examples:

Unacceptable: The <SOI> shall have 100% availability.

[This is unacceptable because 100% is an absolute that is impossible to achieve and verify. Also, available over what time period?]

Improved: The <SOI> shall have an Availability of greater than or equal to 98% during Operating_Hours.

[Note use of the glossary to define Operating_Hours. Alternatively, this requirement could be stated as a need. Then those transforming the need into requirements would develop a feasible concept for doing so and then derive specific Design Input Requirements that would result in the need being met.]

Unacceptable: The Pumping_Station shall maintain the flow of water at 100 liters per second for 30 minutes.

[This is unacceptable because the requirement is impossibly precise to maintain in terms of the flow rate (and impossibly precise to measure in verification) as well as being impossibly precise to deliver for 30 minutes exactly. Also, it is not clear under what conditions this applies—when in operations, when powered on, when commanded?]

Improved: When in operations, the Pumping_Station shall maintain Water_Flow at 100 ±10 liters per second for greater than 30 minutes.

[Now the range of acceptable flow performance and time frame is clear and feasible. Given that the condition "when in operations" this requirement applies whenever the Pumping_Station is active—no matter how long. Also, what is the intent for how long—31 minutes satisfies the requirement, but what if the real intent is for however long the Pumping Station is in operations which could be hours or days—again depending on the operating conditions and need. Further, is the 30 minutes of continuous flow or a total of 30 minutes within a set time period?]

4.7 Conditions

4.7.1 R27 – Explicit Conditions

Definition:

State conditions' applicability explicitly instead of leaving applicability to be inferred from the context.

Elaboration:

Sometimes needs or requirements are only applicable under certain conditions and multiple actions may need to be taken when a given condition exists. In these cases, the condition should be repeated in the text of each need statement or requirement statement, rather than stating the condition and then listing the actions to be taken. This will enable verification of each action occurring when the condition exists and identifying any specific action that fails to occur and resolving that issue specifically.

As presented in R1, and described in more detail in Appendix C, some common applicability conditions include *event-driven requirements*, *state-driven requirements*, *optional features requirements*, or *unwanted behavior requirements*. Furthermore, some of those applicability conditions can be mixed together ending up into a more complex yet complete structure (mixing a *while* statement to refer to a given state, and *when* statement to refer to a trigger is common in many system requirements).

If a condition that applies to a need or requirement is not stated explicitly within the need or requirement statement, the need or requirement statement is not Complete (C4), Verifiable/Validatable (C7), nor Correct (C8) unless the condition clause is included.

Needs are communicated at a higher level of abstraction, often concerning an overall capability of the system, as such they can be ubiquitous at this level of abstraction and are often stated without any specific conditions. However, if a specific condition is applicable, it should be included within the need statement to ensure the intent clear.

See also R1, R11, R18 and Appendix C.

Examples:

Unacceptable:

The Free_Access_Mode shall be set to ON within <xxx ms>.

The Fire_Door_Unlock command having the characteristics defined in <ICD xyz> shall be sent to each Fire_Door within <xxx ms>.

[This is unacceptable because the condition or trigger for these actions is not stated. Also, note the actions are written in passive voice which violates R2. Each action needs to be communicated in a separate active voice requirement statement with the condition or trigger for which the action is to be initiated. For multiple conditions for a single action see R28.]

Improved: (Split into two requirements)

In the event of Fire_Detection, the Fire_Control_Subsystem shall set the Free_Access_Mode to ON within <xxx ms>.

In the event of Fire_Detection, the Fire_Control_Subsystem shall send the Fire_Door_Unlock unlock command having the characteristics defined in <ICD xyz> to each Fire_Door within <xxx ms>.

[Note that there is a Fire Detection subsystem responsible for detecting fires and setting the Fire_Detection parameter to “TRUE” which triggers the two actions. All other subsystems that are monitoring when the Fire_Control_Subsystem has set the Free_Access_Mode to “TRUE” will have their own requirements to monitor this parameter and take the required action(s) as stated in their set of requirements. All Fire Doors will have their own requirements to monitor for the Fire_Door_Unlock command and to unlock when the command is received.]

4.7.2 R28 – Multiple Conditions

Definition:

Express the propositional nature of a condition explicitly for a single action instead of giving lists of actions for a specific condition.

Elaboration:

Sometimes a given action is to be performed based on the existence of multiple conditions or a combination of one or more conditions. When multiple conditions are listed for a single action in a single requirement statement, it may not be clear whether all the conditions must hold (a conjunction) or any one of them (a disjunction) for the action to take place.

The wording of the requirement should make this clear in order to avoid ambiguity but also to facilitate verification the system performs each of the individual actions as specified.

When the combination of multiple conditions that trigger an action is complex, consideration should be given to the use of a diagram or table (see R23).

Needs are communicated at a higher level of abstraction often concerning an overall capability of the system, as such they are ubiquitous at this level of abstraction and are often stated without any specific conditions.

Examples:

Unacceptable:

The Audit _Clerk shall be able to change the status of the action item when:
- the Audit _Clerk originated the item;
- the Audit _Clerk is the actionee; and
- the Audit _Clerk is the reviewer.

[This is unacceptable because it is not clear whether all the conditions must hold (a conjunction) or any one of them (a disjunction). Also, the requirement contains the phrase "be able to" which violates R11.]

Better: If the requirement is interpreted as a disjunction:

The Audit_System shall change the Action_Item status requested by the Audit_Clerk when one or more of the following conditions are true:
- the Audit_Clerk originated the Action_Item;
- the Audit_Clerk is the Actionee; or
- the Audit_Clerk is the Reviewer.

[Although the requirement is now improved, there is still the issue of verification. If the status does not change with one of the listed conditions, then the entire requirement fails verification, even if the other two conditions are met as specified.]

Better*:* In this case, it would be best to state three distinct requirements as this makes each atomic and retains the same overall intent. From an allocation, traceability, and verifiability perspective this form is preferable.

The Audit_System shall change the Action_Item status requested by the Audit_Clerk when the Audit_Clerk originated the Action_Item.
The Audit_System shall change the Action_Item status requested by the Audit_Clerk when the Audit_Clerk is the Actionee.
The Audit_System shall change the Action_Item status requested by the Audit_Clerk when the Audit_Clerk is the Reviewer.

Better if interpreted as a conjunction:

The Audit_System shall change the Action_Item status requested by the Audit_Clerk when the following conditions are true:
- the Audit_Clerk originated the Action_Item AND
- the Audit_Clerk is the Actionee AND
- the Audit_Clerk is the Reviewer.

[In this form, the three conditions are expressed as a logical condition such that all three must all be true for the logical condition to be true.]

4.8 Uniqueness

4.8.1 R29 – Classification

Definition:
Classify needs and requirements according to the aspects of the problem or system it addresses.

Elaboration:
By classifying needs or requirements by type or category, it is possible to group or sort requirements to help identify potential duplications and conflicts within a given type or category.

The ability to view specific groups of needs or requirements can also assist in identifying what needs or requirements may be missing helping to address the characteristic of completeness (C10) as associated with the set of needs or set of requirements.

It is useful to assign to each need statement and requirement statement an attribute associated with its type or category. (See NRM for more guidance on organizing needs and requirements and the use of Attribute A 40 - type/category).

Examples of types or categories of needs and requirements include form, fit, function/performance, quality (-ilities), and compliance (standards and regulations). These can be further grouped in to sub types or categories. Refer to the NRM Sections 4 and 6 for a detailed discussion concerning organizing needs and requirements.

The type/category attribute is most useful because it allows the needs and requirements database to be viewed by a number of designers and stakeholders for a wide range of users. For example, maintainers could review the database filtering out all but the maintenance requirements, engineers developing test plans could filter and extract all verification requirements, and so on. To that end then, the organization would choose types or categories that are useful for the management of their need or requirement set tailored to their specific product line and processes. As discussed in the NRM, the use of attributes allows reports to be generated for all needs or requirements of a certain type or category, allowing the identification of dependencies, duplication, conflicts, or absence of requirements.

Each different type might be related to one or a few different patterns in the catalog of agreed patterns.

The classification used for needs and requirements is defined at organization or project level. The classification schema should be defined as part of the project data dictionary.

How an organization organizes needs and requirements should be clearly defined in the organization's process documents and associated templates for organizing sets of needs and requirements.

Examples:

Example classifications: Functional, Performance, Operational, Reliability, Availability, Maintainability, Safety, Security, Design and Construction Standards, Physical Characteristics.

See the NRM for more guidance on organizing needs and requirements.

See Also R41 – Related Requirements

4.8.2 R30 – Unique Expression

Definition:

Express each need and requirement once and only once.

Elaboration:

Avoid including the same or equivalent need and requirement more than once, either as a duplicate or in similar form. Exact duplicates are relatively straightforward to identify; finding similar need or requirement statements with slightly different wording is much more difficult but is aided by the consistent use of defined terms (R4) and by classification (R29), as well as the use of a properly defined project data dictionary, ontology or glossary in which synonymies and equivalences between terms and acronyms can be defined.

NLP/AI tools can help in the identification of duplicates or similar needs or requirements. In any case, when following the agreed patterns for statement (R1) and a data dictionary, the detection of duplicates and similar results far easier.

Avoidance of duplication can be aided by classification (R29) so a subset of needs or requirements can be compared.

Examples:

Exact duplicates can be found by matching of text strings. The main problem is to identify similarities with different expressions, but which are equivalent.

For example, the following statements are overlapping in that the first is a subset of the second:

The <SOI > shall generate a report of financial transactions containing the information defined in <some standard or contract deliverable listing>.

The <SOI> shall generate a financial report.

4.9 Abstraction

4.9.1 R31 – Solution Free

Definition:

Avoid stating implementation in a need statement or requirement statement unless there is rationale for constraining the design.

Elaboration:

As design inputs, every system endeavor should have a level of needs and requirements for an entity that captures inputs to the architectural and design activities (what - design inputs) without including implementation (how - design outputs) as shown in Figure 2.

System level needs and requirements should provide a system-level requirement for the overall concern to be addressed by design. The first level of architecture may be laid out, but subsystems are considered as black boxes to be elaborated as the project team moves down levels of the architecture. See the NRM concerning levels and moving between levels.

When reviewing a requirement that states implementation (how), ask "for what purpose? The answer will reveal the real requirement. (Notice that this question is subtly different from simply asking "Why?" and encourages a teleological rather than causal response.)

Understanding the concepts of design inputs versus design outputs allows engineers to ask the question: "Is this requirement addressing "what" the system needs to do versus "how" the system needs to do it"? The answer to these questions has the potential to help do three things:

1. Rephrase the requirement in terms of the concern to be addressed by the design.
2. Determine if the requirement is at the right level.
3. Identify which design input requirement(s) or need(s) the design output requirement is addressing or whether the design input requirement is missing.

Often when rationale is provided with requirements that state implementation (design output), the rationale often answers the "for what purpose?" question, and thus the real (and often missing) design input requirement may be extracted from the rationale.

Examples:

General: For a medical diagnostic system

- Stakeholders need statement: "The <stakeholders> need the <diagnostic system> to measure [something] with an accuracy as good as or better than similar devices in the market."

  [This is an appropriate level of abstraction for a stakeholder need statement, clearly stating the expectation the stakeholders have concerning accuracy, however this would not be a good design input requirement.]

- Requirement transformed from the stakeholder need statement: "The <diagnostic system> shall measure [something] with an accuracy of [xxx]."

  [The developers have explored various concepts for meeting the need for accuracy, have examined candidate technologies, have accessed their maturity (technology readiness level (TRL), and have decided that the value [xxx] is feasible with acceptable risk for this lifecycle stage. As stated, this is an appropriate level of detail for a design input requirement.]

Note that when dealing with measures, both accuracy and precision must be addressed, either as separate requirements or including both in a measurement requirement defining characteristics of the measured parameter.

These system-level design input accuracy and precision requirements are then allocated to the parts of the system architecture that have a role in meeting the overall system accuracy and precision requirements. For a medical diagnostic system this could include allocations to the hardware (instrument), assay (biological sample), and software. These allocations could then be further sub-allocated within the hardware, assay, and software lower-level entities.

As long as the requirements are written on the accuracy and precision allocations and not how that accuracy or precision will be obtained by one of the architectural entities, the requirements are Design Input Requirements. As soon as specific hardware components are named (laser, LEDs emitting light at a specific wavelength, optical system components, specific magnifications, algorithms, formulations, etc.,) then the requirements are reflecting design outputs and need to be communicated within design output artifacts (specifications).

Unacceptable: Traffic lights shall be used to control traffic at the intersection.

[This is unacceptable because "Traffic lights" are a solution (design output). Why are traffic lights needed? This requirement is also written in passive voice - see R2]

Improved: (several requirements):

When a Pedestrian signals an intent to cross the street at the Intersection, the Traffic_Control_System shall provide [the Pedestrian a "Walk" signal AND provide the traffic a "Stop" signal].

The Traffic_Control_System shall limit the Wait_Time of Vehicles traversing the Intersection to less than Daylight_Wait_Time during Normal_Daytime traffic conditions.

[Note that glossary definitions should be used for Traffic_Control_System, Daylight_Wait_Time_Value, Normal_Daytime, Vehicles, and Intersection.]

Unacceptable: When a pedestrian signals his presence by pressing a button on the traffic-light pillar, the traffic light shall turn red for the traffic to stop.

[This is unacceptable because this requirement contains solution-biased detail - design output. In addition, the requirement is unacceptable because it has additional information concerning the action traffic is to take and does not specify the duration of the red light. The use of a pronoun is also not recommended].

Improved: When the presence of a Pedestrian that needs to cross the street at the Intersection during Day_Light_Hours is detected, the Traffic_Control_System shall issue a Traffic_Stop_Signal for Average_Pedestrian_Crossing_Time.

[This design input requirement allows freedom in determining the best solution (design output), which may be a means of automatic detection rather than button pushing along with the type and characteristics of the signal issued. Note that pedestrians may be in proximity of the intersection that do not wish to cross the street.]

[An interesting consideration from a consistency or conflict perspective, what if the Average_Pedestrian_Crossing_Time" is greater than the "Daylight_Wait_Time_Value" defined for the traffic in the previous example. Each time is from a different entity perspective.]

[Note that glossary definitions should be used for "Pedestrian", "Traffic_Control_System", "Traffic_Stop_Signal", "Day_Light_Hours", "Average_Pedestrian_Crossing_Time", and "Intersection".]

Exceptions and relationships:

Sometimes solutions have to be described in requirements, even if it is very detailed for a given level, for example if the airworthiness authorities require the use of a specific template for a certain report; or if a naval customer requires that the new naval vessel be equipped with a specific weapon system from a specific supplier; or if all vehicles in a fleet are required to use the same fuel or the next model make use of a particular engine. In these cases, it is not a premature solution, but a real stakeholder or customer need concerning a constraint. As such this is acceptable as a design input.

However, as a rule, if a detailed, specific design solution is expressed as a design input without proper justification, it may be a premature solution and should be communicated as a design output and an appropriate set of design input needs and associated requirements developed that communicate "For what purpose?" which the design output requirements can be traced to.

Examples of issues concerning design outputs expressed as design inputs include:

1. The project is developing an upgrade to an existing system (brownfield SE). Rather than documenting design input "what" requirements, the project team focuses on known solutions and implementations and documenting design output level requirements as design inputs. This is problematic in that the real "for what purpose?" question is not being addressed and the real Design Input Requirements are not communicated.
2. A similar case exists when procuring an off-the-shelf (OTS) solution and communicating the requirements for that solution as Design Input Requirements rather than in a design output specification. Again, this is problematic in that the real "for what purpose?" question is not being addressed and the real Design Input Requirements are not communicated.
3. A common issue arises when an existing system or OTS exists, yet the project develops and documents as design inputs a detailed design output level set of requirements (design output specification) for the system rather than a set of design input set of "what" requirements that would guide the selection of an appropriate OTS. This can result in a redundant set of requirements that are not necessary, will have to be verified - adding additional cost to the project, yet not addressing the "for what purpose?" design input requirement set that should drive the selection of the specific OTS. Refer to the NRM for a detailed discussion the use of OTS entities.

4. As part of the lifecycle concept analysis and maturation activities the project team developed a prototype to access feasibility. The resulting needs and resulting requirements are based on the prototype and an associated trade study that resulted a specific solution that meets the needs of the stakeholders. Rather than communicating the Design Input Requirements, the design output requirements for the prototype are included in the design input set of requirements while not addressing the "for what purpose?" design input requirement set that would drive the design for the prototype.

4.10 Quantifiers

4.10.1 R32 – Universal Qualification

Definition:
Use "each" instead of "all", "any", or "both" when universal quantification is intended.

Elaboration:
The use of "all", "both", or "any" is confusing because it is hard to distinguish whether the action happens to the whole set or to each element of the set. "All" can also be hard to verify unless "all" can be clearly defined as a closed set. In many cases, the word "all" is unnecessary and can be removed, resulting in a less ambiguous need or requirement statement.

Examples:

Unacceptable: The Operation_Logger shall record any (or all) warning messages.

[This is unacceptable because of the use of the word "any", which is then made worse by the addition of "(or all)".]

Improved: The Operation_Logger shall record each Warning_Message.

[Note that Warning_Message must be defined in the glossary so that it is clear that the SOI only will record each defined Warning Message. Also, any conditions or performance must also be included in the requirements statement for it to be complete.]

Unacceptable: The Record_Subsystem shall display per <Display Standard xyz> the Names of both of the Line_Items.

[This is unacceptable because of the use of the word "both".]

Improved: The Record_Subsystem shall display per <Display Standard xyz> the Name of each Line_Item.

Improved: The Record_Subsystem shall display per <Display Standard xyz> each Line_Item_Name.

4.11 Tolerance

4.11.1 R33 – Range of Values

Definition:
Define each quantity with a range of values appropriate to the entity to which the quantity applies and against which the entity will be verified or validated.

Elaboration:
When it comes to defining performance, single-point values are seldom sufficient and are difficult to test. Ask the question: “if the performance was a little less (or more) than this, would I still buy it?” or “Is it still fit for its intended use?” If the answer is yes, then change the need or requirement statement to reflect an acceptable range of values. It also helps to consider the underlying goal: are you trying to minimize, maximize or optimize something? The answer to this question will help determine whether there is an upper bound, lower bound, or both. State the quantities contained in a need or requirement statement with ranges or limits with a degree of accuracy and precision that is appropriate to the entity to which the need or requirement applies and against which the entity will be verified against. Care should be taken to avoid unspecified value ranges and ensure the quantities are expressed with tolerances or limits. There are two reasons for this: 1) Several requirements may have to be traded against each other, and providing tolerances or limits is a way of describing the trade-space. Seldom is a quantity absolute. A range of values is usually acceptable, providing different performance levels. 2) Verification against a single absolute value is usually not feasible or at best expensive and time consuming, whereas verification against a defined range of values with upper and lower limits makes verification more manageable. Care should also be taken to ensure the tolerances are no tighter than needed. Tighter tolerances can drive costs both in system design and manufacturing as well as the costs in verifying the system can perform within the tighter tolerances. Ranges and tolerances must be described in a consistent and unambiguous way. Examples of possible structures for such information are: 10 kg ± 2 kg 10 kg ± 5% 10 kg (+0.0 kg / -0.5 kg) … from 10 kg to 12 kg. … between 10 kg and 12 kg. Specific scenarios such as '8.0 oz (+0/-0,5 oz)' can be particularly dangerous: 1) They require uniformity in terms of decimal format (R40) between the value and the tolerance ranges: '8.0 oz (+0.0/-0.5 oz)'. 2) The measurement units must be appropriate and clearly defined for each numerical value (R6): '8.0 oz (+0.0 oz / -0.5 oz)' 3) For comprehension reasons, these structures can be simplified as: "from 7.5 oz to 8.0 oz" or "between 7.5 oz and 8.0 oz" (note that the ranges could also be expressed as "from 8.0 oz to 7.5 oz" or "between 8.0 oz and 7.5 oz", but comprehension is facilitated by nominating the lower value first). Also, be aware of tolerance stack up issues. It may be that individual tolerances seem reasonable, but when combined with other related component tolerances, there may be an issue. If one part is at the lower end of the tolerance range and the part it is to interface with is at the higher end of the tolerance range, the parts may not be able to be connected. Thus, the tolerance ranges must be consistent and not in conflict with other related parts of the system architecture in which a part is to interact. See also R40.

Examples:
Unacceptable: The Pumping_Station shall maintain the flow of water at 100 liters per second for 30 minutes. [This is unacceptable because we do not know whether a solution that addresses more or less than the specified quantities is acceptable Also, it is not clear under what conditions this applies – when in operations, when powered on, when commanded?] Improved: When in operations, the Pumping_Station shall maintain Water_Flow at 100 ±10 liters for a minimum of 30 minutes. [Now the range of acceptable flow performance is clear and that the 30 minutes is a minimum acceptable performance. Given that the condition "when in operations", this requirement applies whenever the Pumping_Station is active—no matter how long. Also, a question to be asked is what is the intent for how long—31 minutes satisfies the requirement, but what if the real intent is for however long the Pumping Station is in operations which could be hours or days – again depending on the operating conditions and need. What if there is not enough water at the inlet to the pumping station to establish the required flow rate?] Unacceptable: The Flight_Information_System shall display the current altitude to approximately 1 meter resolution. [This is unacceptable because it is imprecise. What is "approximately" in the context of a distance of 1 meter? Who has the option of deciding what is "approximately"? How will "approximately" be verified? What is the acceptable tolerance?] [Note that care must be taken to confirm that the units are appropriate in the context of the organizational and project templates. See also R6.] Improved: The Flight_Information_System shall display Current_Altitude with an accuracy of ±1 meter. [Note that "Current_Altitude" must be defined in the glossary since there are a number of possible interpretations of the term.] Unacceptable: The <SOI> shall limit arsenic contamination in the drinking water to allowable levels. Rationale: Arsenic contamination in drinking water can cause health problems. [While "allowable" is acceptable in a need statement, it is unacceptable in a requirement statement because allowable is ambiguous - allowable by whom? What specific concentration is allowable? In what market?] Unacceptable: The <SOI> shall limit arsenic contamination in the drinking water to 1 part per trillion. Rationale Attribute (A1): Arsenic contamination in drinking water can cause health problems. [This is unacceptable because the EPA contamination limit in drinking water is 10 parts per billion. Requiring a tighter limit may be beyond the ability of current technology to measure or if measuring concentrations of 1 part per trillion are possible, the cost to do so may be unacceptably high. Also, no range is specified. Using 'less than' is probably the real intent.] Improved: The <SOI> shall limit arsenic contamination in the drinking water to less than10 parts per billion. Rationale Attribute (A1): EPA set the arsenic standard for drinking water at 10 ppb (or 0.010 parts per million). The EPA has determined that concentrations of this level or less will protect consumers from the effects of long-term, chronic exposure to arsenic.
Exceptions and relationships:

The use of tolerances in need statements may not be as mandatory as their use in requirement statements—see Appendix D.

4.12 Quantification

4.12.1 R34 – Measurable Performance

Definition:

Provide specific measurable performance targets appropriate to the entity to which the need or requirement is stated and against which the entity will be verified to meet.

Elaboration:

Some words signal unmeasured quantification, such as "prompt", "fast", "routine", "maximum", "minimum", "optimum", "nominal", "easy to use", "close quickly", "high speed", "medium-sized", "best practices", and "user-friendly." These are ambiguous and need to be replaced by specific quantities within feasible ranges that can be measured.

Examples:

Unacceptable: The <SOI> shall use minimum power.

[This is unacceptable because both words "use" and "minimum" are ambiguous and unverifiable.]

Improved: The <SOI> shall consume less than or equal to 50W of mains power.

[This both considers the underlying goal—to minimize power consumption—and provides a measurable target.]

Unacceptable: The engine shall achieve an emissions level that is at least 5% less than the competition's emission levels 2 years from now.

[This is an actual requirement from marketing to an engineering department. The statement sets a completely unmeasurable end state.]

Improved: The Engine shall achieve an emissions level that is less than or equal to xxx.

[where xxx represents the required threshold value, including the appropriate units.]

Unacceptable: The <SOI> shall conform to best practices for spurious emissions.

[This statement is vague and unverifiable from number of specifics]

Improved: The <SOI> shall limit Spurious_Emissions in accordance with <Clause xab of Standard XYZ>.

Exceptions and relationships:

Some quantification terms such as "minimum", "maximum", "optimal" are almost always ambiguous. Other terms may be ambiguous at lower levels but sufficient at the higher levels and as need statements. For example, it may be appropriate that the business state that "The Aircraft shall provide class-leading comfort."—while such a requirement is not quantifiable and therefore not measurable, it may be sufficient for the business to communicate its intentions as a need statement to developers who can then turn comfort into such measurable quantities such as seat dimensions and leg length.

This exception also applies to needs stated at the system or system element levels.

4.12.2 R35 – Temporal Dependencies

Definition:

Define temporal dependencies explicitly instead of using indefinite temporal keywords such as “eventually”, “until”, “before”, “after”, “as”, “once”, “earliest”, “latest”, “instantaneous”, “simultaneous”, and “at last”.

Elaboration:

Using indefinite temporal keywords such as those express above signal non-specific timing, are ambiguous, and not verifiable. Indefinite temporal keywords can cause confusion or unintended meaning. These words should be replaced by specific timing constraints.

Examples:

Unacceptable: Continual operation of the pump shall eventually result in the tank being empty.

[This is unacceptable because “eventually” is ambiguous. Also, this statement is not written in the proper form. It is written on “operation” rather than on a requirement on the pump which violates R3.]

Improved: The Pump shall remove greater than 99% of the Fluid from the Tank in less than 3 days of continuous operation.

4.13 Uniformity of Language

4.13.1 R36 – Consistent Terms and Units

Definition:

Use each term and unit of measure consistently throughout need and requirement sets as well as associated models and other SE artefacts developed across the lifecycle.

Elaboration:

R4 requires the definition of terms in each requirement and R6 requires units of measure to be included with numbers. In addition to those rules, terms and units of measure must be used consistently throughout not only the sets of needs and requirements, but all artifacts developed across all lifecycle stages.

A common ontology therefore needs to be defined for each project defining terms and units of measure across all artifacts, including the sets of needs and requirements as well as all design output artifacts. Synonyms are not acceptable.

A glossary or data dictionary is extremely useful to define words precisely. Terms defined within the glossary or data dictionary are capitalized to signify that the word or term being used has a specific meaning in the context of the set of statements.

For a numeric value for a given variable that may appear in multiple needs or requirements, to help ensure consistency, it is a best practice to define the variable and its numeric value and units of measure in a glossary or data dictionary. An example would be the maximum and minimum environment temperatures specified in multiple places. For example, Maximum_Temperature_Value. This also resolves the units problem as the units can be defined at the same time in the glossary entry. Although it could be argued that the loss of the value in the text makes the requirement harder to understand, the ability to maintain consistency is of higher priority.

Ideally, the glossary or data dictionary used for the sets of needs and requirements is the same as the project glossary or data dictionary.

When describing an action performed by a system or system element, states and modes should be used consistently to make sure it is clear as to which state/mode the statement applies.

Additionally, whenever mode/state definitions or transitions are involved, these must come from a project dictionary (ontology, model, etc.) ensuring its consistent use throughout the specification.

It is also important to maintain consistency in how a group of related units is formatted for example for hours, minutes, or seconds (hh:mm:ss) or dates month day year (mm/dd/yyyy vs yyyy/mm/dd vs dd/mm/yyyy)

Examples:

Unacceptable: It would not be acceptable for one requirement to refer to an entity using one term and another to refer to the same entity using another term.

For example, a subsystem set of Design Input Requirements contains the following three requirement statements:

The Radio shall ....
The Receiver shall ....
The Terminal shall ....

Or:

The Bleed_Valve shall ....
The High-Pressure_Bleed_Valve shall ....
The HPBV shall ....

If each term refers to the same subject, the statements need to be modified to use the same word (or, if they are meant to be different, the words must be defined to be so).

Improved: Settle on only one term, define it in the glossary or data dictionary, and then use it consistently in each need, requirement, and design output artifact.

Unacceptable: When the Input_Valve is connected to the Water_Source, the Control_Subsystem shall open the Inlet_Valve.

[Two terms are used for the same thing “Input_Valve” and “Inlet_Valve”.]

Improved: When the Inlet_Valve is connected to the Water_Source, the Control_Subsystem shall open the Inlet_Valve.

Unacceptable: It would not be acceptable for one requirement to use one unit of measure (for example, US - feet) and another requirement to use another unit of measure (for example, Metric - meter).

Improved: Settle on which units of measure will be used and use that unit of measure consistency across all SE artifacts including needs, Design Input Requirements and design output specifications.

Also, see R40 for guidance regarding the conversion from one measurement system to another and the use of significant digits.

4.13.2 R37 – Acronyms

Definition:

If acronyms are used, they must be consistent throughout need and requirement sets as well as associated models and other SE artefacts developed across the lifecycle.

Elaboration:

The same acronym must be used in each need and requirement; various versions of the acronym are not acceptable. The use of different acronyms implies that the two items being referred to are different. Inconsistency in the use of acronyms can lead to ambiguity.

In a document-based practice of SE, a common rule is to use the full term and the abbreviation or acronym (in brackets) the first time and then use just the abbreviation or acronym from then on within a document (as is done in this Guide). In a data-centric practice, need and requirement sets that are generated and managed within an RMT, so there is no guarantee that the set will be extracted in any particular order, so the old practice is not useful. To prevent confusion, either avoid using acronyms or (more usefully) ensure that all acronyms are defined in the project glossary or data dictionary.

When used, acronyms must be used consistently throughout not only the sets of needs and requirements, but all artifacts developed across all lifecycle stages.

Acronyms must be written in a consistent way in terms of capitalization and periods. For example, always use "CMDP" and not "C.M.D.P." nor "CmdP".

Examples:

Unacceptable: It would not be acceptable for one requirement to use the acronym "CP" for command post and another acronym "CMDP" to also refer to the "command post." The use of two different acronyms implies that the two system elements being referred to are different.

Improved: Settle on only one acronym, define it in the list of acronyms, and then use it consistently throughout the requirement set.

Unacceptable: It would not be acceptable for one requirement to refer to the "Global Positioning System" and the remaining requirements refer just to "GPS."

Improved: Use the full term every time or, perhaps more usefully, define the acronym in the project acronym list or glossary and use it every time.

4.13.3 R38 – Abbreviations

Definition:

Avoid the use of abbreviations in needs and requirement statements as well as associated models and other SE lifecycle artefacts.

Elaboration:
The use of abbreviations adds ambiguity and is to be avoided unless the context is clear, and the abbreviation is clearly defined in the project glossary or acronym list. It is common to have an abbreviation with multiple meanings depending on context. In the case where there is a single meaning and that abbreviation is defined in the project glossary, it is acceptable to use the abbreviation within the text of the need or requirement. However, when there are abbreviations that have different meanings, it is a best practice to avoid ambiguity by spelling out the abbreviation.

Examples:
Unacceptable: It would not be acceptable for one requirement to refer to the abbreviation "op" meaning operation and another to refer to the "op" meaning the operator. Improved: Avoid the abbreviation and use the full term every time.

4.13.4 R39 – Style Guide

Definition:
Use a project-wide style guide for individual need statements and requirement statements.

Elaboration:
A style guide provides a template and patterns for developing requirements, defines the attributes that the organization chooses to document with each need statement and requirement statement, selects the requirement patterns to be used for specific types of needs and requirements, and defines what other information should be included (such as the glossary). The style guide should also list the rules the organization wants to use (based on this Guide). When managing needs and requirements electronically (versus in a printed document) within a RMT or other systems engineering tool, this information is the basis of the schema that defines the organization of the data within the tool database. To ensure need statements and requirement statements are consistently structured, the organization needs to include pre-defined patterns or templates in their development and management process. The patterns or templates can come from an international standard, or an organizational standard set of patterns or templates based on type of stakeholder need or requirement. If using an RMT, a standard schema should be defined as part of the needs and requirements development and management process. The patterns need to be tailored to the organization's domain and the distinct types of products within that domain. Patterns and templates for individual need statements and requirement statements help ensure consistency and completeness of the individual statements (see Appendix C). See also R1.

Examples:
For example, each organization should have a style guide or schema to address such issues as the selection and definition of what need patterns and requirement patterns should be used for writing need statements and requirement statements of a particular type, the selection and use of attributes, standard abbreviations and acronyms, layout and use of figures and tables, layout of documents or databases and, statement numbering.

Terms used throughout the sets of needs and sets of requirements are defined in an ontology or at least a project glossary.

4.13.5 R40 – Decimal Format

Definition:

Use a consistent format and number of signification digits for the specification of decimal numbers.

Elaboration:

A decimal separator is a symbol used to separate the integer part of a number from the fractional part of a number written in decimal form. For example, a period '.' is used as the decimal separator (called a decimal point) in '12.45'; and a comma ',' is used as the decimal separator (called a decimal comma) in '12,45'. The choice of decimal separator also affects the choice of symbol for the thousands separator.

There are three principal conventions to combine thousand separators and decimal separators (where 'x' represents the thousands and 'y' represents the hundreds, tens and units of the integer part, while 'z' defines the decimal part):

- A space is used as the thousands separator (recommended internationally) and a decimal point or a decimal comma as the decimal separator—for example 'xxx yyy.zzz' and 'xxx yyy,zzz'.
- A comma is used as a thousands separator (used in most English-speaking countries) and a decimal point is used as a decimal separator—for example 'xxx,yyy.zzz'.
- A period is used as a thousands separator (used in many non-English speaking countries) and a decimal point is used as a decimal separator—for example 'xxx.yyy,zzz'.

Any one of the above conventions may be used, providing the one convention is used consistently throughout.

When specifying numbers between 1 and –1, avoid using '.99' or '–,99'; use the zero in front of the decimal separator. For instance, use '0.99' or '–0,99'.

The use of the number of significant digits should also be consistent throughout (based on the need for precision in the need and requirement set). For example, avoid using '512', '10.15' and '5.1' in the same set—use instead '512.00', '10.15' and '5.10' (assuming two significant places is the agreed convention throughout).

In respect to numerical values represented as factions (1/16). Some tools may provide decimal equivalents in terms of decimal numbers with results of .06 or 0.063 instead of (0.0625) based on number of decimal places allowed in the requested output. Fractions imply a level of accuracy and tolerancing which may be lost when the design drawings are created and transferred to another entity.

Caution concerning significant digits should also be observed when converting from one measurement system to another. For example, kilograms to pounds. 1 kg = 2.2046226218 lbs 1 lbs. = 0.45359237 kg Example: convert 15 kg to lbs.: 15 kg = 15 × 2.2046226218 lbs. = 33.0693393277 lbs. What number of significant digits is appropriate? In this example is it ok to use 33 lbs., 33.1 lbs., 33.07 lbs., or 33069 lbs.? Notice the 15 kg that was being converted had no decimal places. What precision is needed when 15 kg a "ballpark" estimation of someone says "about 1000kg"? If so, using 1, 2, or 3 decimal places may not be warranted.

Again, organizations need to be consistent concerning these conversions and the number of signification digits included in a need or requirement statement. Where a conversion has been made, it may be useful to record in the Rationale Attribute A1 that the figure has been converted from another measurement system and rounded to x decimal places.

Care should be taken with regard to the selection of the number of significant digits on the cost of design, verification, and validation—inappropriately high precision can cause significantly higher costs.

Examples:

Unacceptable: The <SOI> shall have an MTBF of less than or equal to 9.499,99 h. &

The <SOI> shall consume less than or equal to 0.99 W.

[These two statements are unacceptable when included in the same set because a different format is used in the two statements.]

Improved:

The <SOI> shall have an MTBF of less than or equal to 9,499.99 h. &

The <SOI> shall consume less than or equal to 0.99 W.

OR

The <SOI> shall have an MTBF of less than or equal to 9.499,99 h. &

The <SOI> shall consume less than or equal to 0,99 W.

Unacceptable: The <SOI> shall have an MTBF of less than or equal to 0.99 h. &

The <SOI> shall consume less than or equal to .99 W.

[These two statements are unacceptable when included in the same set because a different format is used in the two statements.]

Improved:

The <SOI> shall have an MTBF of less than or equal to 9,499.99 h. &

The <SOI> shall consume less than or equal to 0.99 W.

4.14 Modularity

4.14.1 R41 – Related Needs and Requirements

Definition:

Group related needs and requirements together.

Elaboration:

Need and requirements statements that belong together should be grouped together. This is a good principle for organizing sets of needs and requirements within an RMT or another SE tool.

One approach is to put requirements into groups based on their type or category. See also R29 and R42 and the NRM for more guidance on organizing needs and requirements.

An example grouping could be form, fit, function, quality, and compliance. See R42. This grouping forces the team to look at the system from each of these perspectives helping to ensure completeness of the sets. Methodologies that only focus on function/performance and fit will result in incomplete sets of needs and requirements.

In functional decomposition, functional/performance requirements are grouped by the function that originates them.

Grouping by type/category helps to ensure completeness of the sets, consistency within sets, and makes it easier to identify missing, overlapping, inconsistent or redundant requirements. See also R29.

Examples:

Requirements may be related by:

- type (for example, critical functions, enabling functions, safety, or security requirements);
- scenario (for example, requirements arising from a single scenario).
- interactions with other systems (for example interface requirements).
- function (for example, multiple requirements defined for a function, each addressing a different performance characteristics, mode, state, condition, or trigger.)
- capability (for example a needed capability may be released by a set of requirements that together result in the needed capability to be provided.
- compliance (for example requirements whose purpose it is to implement a requirement in a standard or regulation.

See R29 and the NRM for more guidance on organizing needs and requirements.

4.14.2 R42 – Structured Sets

Definition:

Conform to a defined structure or template for organizing sets of needs and requirements.

Elaboration:

A well-organized set of needs and requirements allows an understanding of the whole set to be assimilated without undue cognitive loading on the reader.

Despite what has been stated about the completeness of individual need statements and requirement statements, it is often easier to understand when they are placed in context with other related requirements.

Whether managed within a document or within an RMT, the ability to link together, through traceability, related needs or requirements and groups of needs or requirements is a vital tool in identifying repetition and conflict between need statements or requirement statements.

Templates for organizing needs and requirements will help ensure consistency and completeness of your requirement set.

Refer to the NRM for a detailed discussion on organizing needs and requirements.

See also R29, R39, and R41

Examples:

For sets of needs or requirements, an outline can be defined that organizes them in categories or by type.

Examples of Type/Category of needs and requirements include:

- Function: Functional/Performance.
- Fit: Operational: interactions with external systems - input, output, external interfaces, operational environmental, facility, ergonomic, compatibility with existing systems, logistics, users, training, installation, transportation, storage.
- Quality (-ilities): reliability, availability, maintainability, accessibility, transportability, quality provisions, growth capacity.
- Form: physical characteristics.
- Compliance:
    - Standards and regulations—policy and regulatory.
    - Constraints—imposed on the project and the project must show compliance.
    - Business rules—a rule imposed by the enterprise or business unit.
    - Business requirements—a requirement imposed by the enterprise or business unit.

Refer to the NRM Sections 4 and 6 for a more detailed discussion on this approach to organizing needs and requirements.

Outlines for organizing needs and requirements can come from international standards or an organizational standard tailored to the organization's domain and different types of products within that domain. (The organization for system level needs and requirements may be different than the organization of a set of software needs and requirements.)