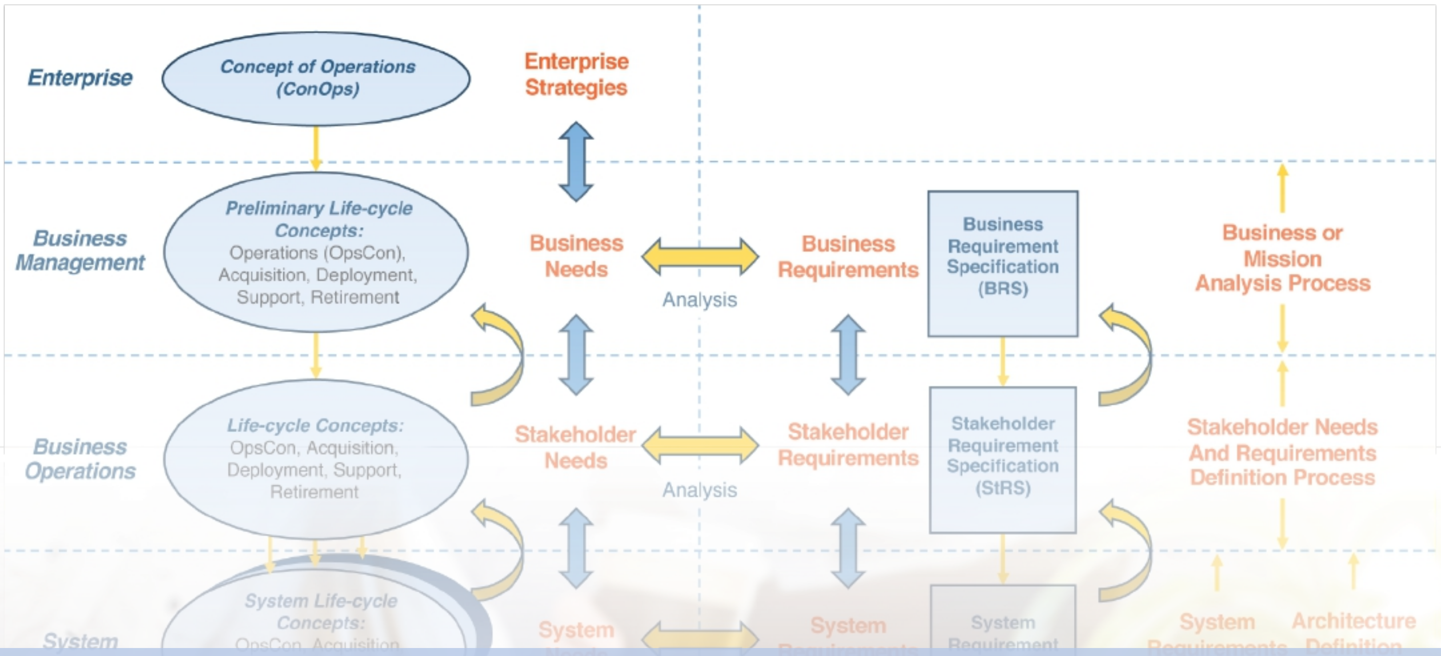
INCOSE



Enterprise
Concept of Operations (ConOps)
Enterprise Strategies
Business Management
Preliminary Life-cycle Concepts: Operations (OpsCon), Acquisition, Deployment, Support, Retirement
Business Needs
Analysis
Business Requirements
Business Requirement Specification (BRS)
Business or Mission Analysis Process
Business Operations
Life-cycle Concepts: OpsCon, Acquisition, Deployment, Support, Retirement
Stakeholder Needs
Analysis
Stakeholder Requirements
Stakeholder Requirement Specification (StRS)
Stakeholder Needs And Requirements Definition Process
System
System Life-cycle Concepts: OpsCon, Acquisition, Deployment, Support, Retirement
System Needs
Analysis
System Requirements
System Requirement Specification (SyRS)
System Requirements Process
Architecture Definition Process

Guide to Writing Requirements

1 Jul 2023

COPYRIGHT INFORMATION

Additional Copies/General Information

Copies of the *Guide to Writing Requirements*, as well as any other INCOSE document can be obtained from the INCOSE Store. General information on INCOSE, the Requirements Working Group, any other INCOSE working group, or membership may also be obtained from the INCOSE Central Office at:

International Council on Systems Engineering	Telephone: +1 858-541-1725
7670 Opportunity Road, Suite 220	Toll Free Phone (US): 800-366-1164
San Diego, California 92111-2222 | USA	Fax: +1 858-541-1728
E-mail: info@incose.org	Web Site: http://www.incose.org

GUIDE TO WRITING REQUIREMENTS

Document No.: INCOSE-TP-2010-006-04
ISBN: 978-1-93707-05-4
Version/Revision: Rev 4
Date: 1 July 2023
Prepared by:

Requirements Working Group
International Council on Systems Engineering
7670 Opportunity Road, Suite 220,
San Diego, California 92111-2222 USA

Preface

This Guide has been prepared and produced by the Requirements Working Group (RWG) of the International Council on Systems Engineering (INCOSE). The original document coalesced inputs from numerous INCOSE contributors, edited by Jeremy Dick, published in draft form in December 2009 for internal INCOSE review, and later published as an INCOSE Technical Product in July 2015. Subsequently, the document has been revised extensively following reviews by the RWG members. The latest revision represents a further evolution of the concepts communicated within the Guide based on comments and inputs received from members of the RWG and larger body of INCOSE members. This latest revision also brings the Guide in harmony with the latest versions of ISO/IEC/IEEE 15288, the INCOSE Systems Engineering Handbook version 5, and the other major RWG products: the Needs and Requirements Manual (NRM), the Guide to Needs and Requirements (GtNR), and the Guide to Verification and Validation (GtVV).

Authors

The principal authors of this Guide are:

Michael Ryan, Capability Associates Pty Ltd, Australia
Lou Wheatcraft, Wheatland Consulting, LLC, USA

Major Contributors

Those who made a significant contribution to the generation of the Guide are:

Kathy Baksa, Pratt & Whitney, USA
Ronald S. Carson, Retired, USA
Jeremy Dick, Retired, UK
José Fuentes, The Reuse Company, Spain
Juan Llorens, The Reuse Company, Spain
José Pereira, The Reuse Company, Spain
Ilyes Yousfi, The Reuse Company, Spain
Rick Zinni, L3Harris Corporation, USA

Reviewers

The following reviewers submitted comments during the development of this version of the Guide:

Angel Agrawal, Northrop Grumman, USA
Wale Akinwale, Flex Ltd,
Keith Collyer, Retired
Quinn Fatherley, Lawrence Livermore National Laboratory, USA
Bruno Favoreto, DE
Tami Katz, Ball Aerospace, USA
Thomas Konerth, Bechtel NSE, USA
Carlo Leardi, Tetra Pak, IT
Seth Sherman, L3Harris Corporation, USA
Ernest Tavares, KBR, USA
Raymond Wolfgang, Sandia National Labs, USA

REVISION HISTORY

Revision	Revision Date	Change Description & Rationale
0	12 Jul 2009	Original
0.1	1 Jan 2011	Revision to use standard INCOSE formatting
0.2	30 Jan 2011	Revision to export from DOORS into standard INCOSE formatting
0.3	30 Jan 2012	Revision to accommodate comments from review by INCOSE Technical Operations
0.4	2 March 2012	Revision based on comments from INCOSE RWG leadership
1	17 April 2012	Revision based on comments from INCOSE RWG general membership
1.1	26 Jan 2013	Update at IW2013 by RWG to remove typographical errors
1.2	6 Apr 2013	Update to revise attributes and to amend rules after training packages developed
1.3	10 Feb 2015	Update at IW2015 by RWG at definitions, update characteristics and attributes.
1.4	10 Mar 2015	Updates resulting from the first round of reviews by attendees at INCOSE IW 2015
1.5	22 Mar 2015	Updates resulting from review by RWG members
2	1 Jul 2015	Release by INCOSE Tech Ops
2.1	30 Jun 2017	Updates resulting from review by RWG members at INCOSE IW2016 & IW2017
2.2	10 Feb 2019	Updates resulting from review by RWG members at INCOSE IW2019
2.3	20 Mar 2019	Revision based on comments from INCOSE RWG general membership
3	19 Jul 2019	Release by INCOSE Tech Ops
3.1	1 May 2022	Alignment with other RWG products, editorial, and traceability updates
4.0	1 Jul 2023	Revision based on comments from INCOSE RWG general membership

TABLE OF CONTENTS

LIST OF FIGURES

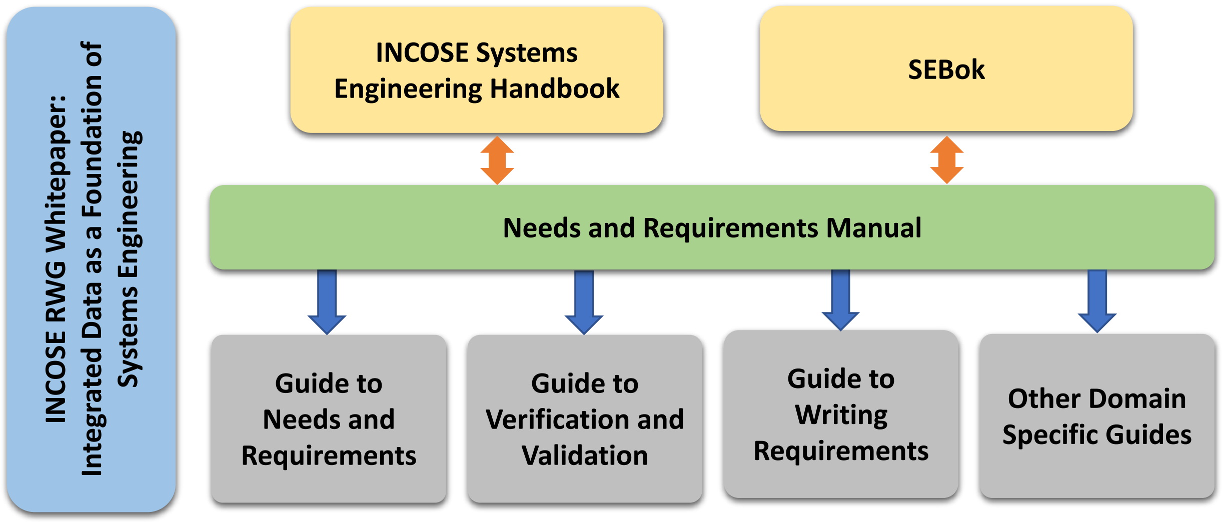
Section 1: Introduction

1.1 Purpose

The purpose of this Guide is to describe how to express need statements (needs) and requirement statements (requirements) clearly and precisely in textual form to support further analysis and implementation, independent of any systems engineering (SE) tool that may be used to capture and manage those needs and requirements throughout the system lifecycle. Clear, concise needs and requirements will be less taxing to interpret as well as simpler to verify, validate, and to identify defects. This, in turn, helps avoid costly rework, schedule slips, and needs and requirements not being realized.

The aim is to draw together advice from existing standards such as ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 29148 along with best practices from the authors, contributors, and reviewers into a single, comprehensive set of characteristics and rules for well-formed need and requirement statements using a structured, natural language.

As shown in Figure 1, this Guide complements and is aligned with the INCOSE Needs and Requirements Manual (NRM) in support of the INCOSE Systems Engineering Handbook (INCOSE SE HB).


INCOSE RWG Whitepaper: Integrated Data as a Foundation of Systems Engineering
INCOSE Systems Engineering Handbook
SEBok
Needs and Requirements Manual
Guide to Needs and Requirements
Guide to Verification and Validation
Guide to Writing Requirements
Other Domain Specific Guides

Original figure created by L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 1: Relationships Among RWG Products

To better understand the context of the material presented in this Guide, the reader is encouraged to review the underlining concepts and activities within the NRM as well as the related guides: Guide to Needs and Requirements (GtNR) and the Guide to Verification and Validation (GtVV). The quality of need and requirement expressions, and the quality of their subsequent verification and validation, are highly dependent on the understanding of the underlying concepts and successful implementation of activities described in the Needs and Requirements Definition and Management (NRDM) the NRM, GtNR and GtVV.

1.2 Scope

This Guide:

- Provides practical, cross domain guidance, with examples, which will enable project teams to define well-formed need and requirement statements, need and requirement expressions, and sets of needs and requirements.
- Provides a definitive source that organizations can reference when developing and documenting a set of rules, guidelines, processes, methods, and templates that are tailored to their specific organizational needs, unique culture, domain, and product line.
- Defines a set of characteristics that should be possessed by well-formed needs, requirements, sets of needs, and sets of requirements to give design teams the best basis to create an architecture and resulting design that will satisfy, as fully as possible, the baselined sets of needs and associated requirements.
- Provides a set of rules for writing needs and requirements that will result in the relevant need and requirement statements having the desired characteristics.
- Addresses the concept of a boilerplate, template, or pattern for need and requirement statements.

This Guide is not about the discovery, capture, or elicitation of requirements; nor is it about needs and requirements analysis, needs and requirements management, and the development of models or design outputs. Those key activities are discussed in the NRM and the GtNR.

Furthermore, although this Guide focuses on textual statements, it does not presume to preclude the use of other means of addressing needs and requirements and their attributes, such as the use of visual models. Approaches such as those contained in SysML diagrams or MBSE tools can be powerful; however, the details of graphical requirements (in or outside a given tool) are outside the scope of this Guide.

The characteristics and rules for needs and requirements discussed in this Guide apply to needs and requirements for any entity no matter the level within an organization or system architecture. As such, they also apply to the business requirements and stakeholder needs and requirements resulting from the ISO/IEC/IEEE 15288 and INCOSE SE HB technical processes: *Business and Mission Analysis*, *Stakeholder Needs and Requirements Definition*, and *System Requirements Definition*.

1.3 Audience

This Guide is intended for those whose role is to *solicit*, *write, review, implement*, and *manage* textual needs and requirements throughout the system development lifecycle process activities, as well as those who *verify* that the realized System of Interest (SOI) meets the requirements and those who *validate* that the realized SOI meets the needs in the intended operational environment when used by the intended users.

Major user groups who will benefit from the use of this Guide include systems engineers, requirements engineers, business analysts, product developers, system architects, configuration managers, designers, testers, verifiers, validators, manufacturers, coders, operators, users, disposers, course developers, trainers, tool vendors, project managers, acquisition personnel, lawyers, regulators, and standards organizations.

This Guide is addressed to practitioners of all levels of experience. Someone new to systems engineering should find specific guidance through the characteristics and rules and associated examples provided herein to be useful, and those more experienced should be able to find new insights through the characteristics and rules discussed in this Guide, often absent from other texts, guides, or standards.

This Guide may also assist tool vendors who are applying Artificial Intelligence (AI) and Natural Language Processing (NLP) to the evaluation of needs and requirement statements. Given the number of characteristics and rules, it may be overwhelming for systems engineers to apply them all simultaneously—that is not the intent of this Guide. The goal of these NLP/AI based tools is to be invaluable as a “digital assistant”, helpful in the crafting of higher quality, well-formed needs and requirements—which is the ultimate goal of this Guide.

1.4 Approach

This Guide presents the underlying characteristics of individual need and requirement statements as well as the characteristics of sets of needs and requirements. Practical rules and patterns are then presented that can be followed when writing need and requirement statements. An understanding of the underlying characteristics and reasons for those characteristics then informs the adaptation of practical rules and patterns which, when followed, will result in quality sets of needs and requirements.

Effective systems engineering transforms the design inputs into design outputs. Figure 2 illustrates the logic flow, leading to the definition of well-formed needs and requirements. In this figure, the needs are contained within an Integrated Set of Needs and the requirements are contained within a set of Design Input Requirements. Together, the Integrated Set of Needs and resulting set of Design Input Requirements are considered inputs into the ISO/IEC/IEEE 15288 *Architecture Definition* and *Design Definition* processes which transform the Design Input Requirements into sets of design output specifications (outputs of the design activity) to which the System Element is realized.

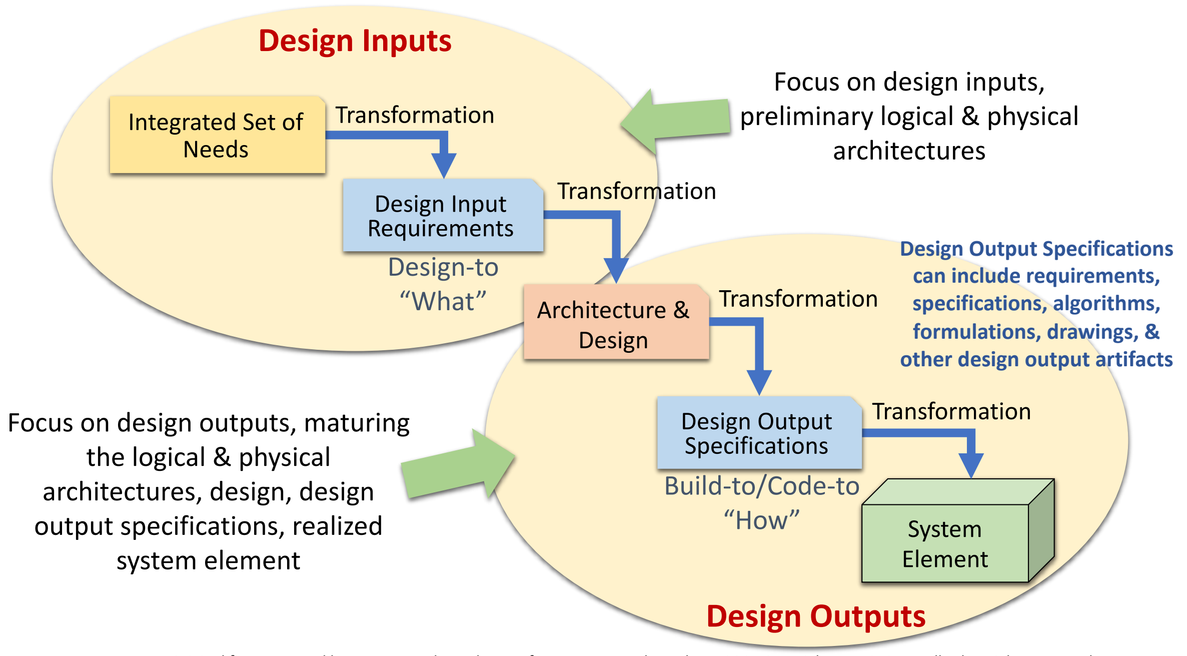

Design Inputs
Integrated Set of Needs
Transformation
Design Input Requirements
Design-to “What”
Transformation
Focus on design inputs, preliminary logical & physical architectures
Architecture & Design
Transformation
Design Output Specifications can include requirements, specifications, algorithms, formulations, drawings, & other design output artifacts
Design Output Specifications
Build-to/Code-to “How”
Transformation
System Element
Focus on design outputs, maturing the logical & physical architectures, design, design output specifications, realized system element
Design Outputs

Figure 2: Needs and Requirements in Context

1.5 Why use the Textual Form of Communication?

For many ideas and concepts that need to be communicated, well-formed, textual need and requirement statements are an effective form of communication that covers the wide variety of concepts that must be communicated throughout a system lifecycle. Text is the enduring method to capture the conversation that is the negotiation between stakeholders in an engineering project.

However, a potential flaw in the use of textual needs and requirement statements is the inherent ambiguity in the use of an unstructured, natural language. English and other natural languages include many synonyms and words with slightly different shades of meaning dependent upon context and culture. Because of this, it can often be difficult to be clear and precise, which is a major concern when the statements are the basis of communication, particularly in a contractual context.

To help avoid the ambiguity of an unstructured natural language, this Guide imposes a rule-based structure—a “structured, natural language” to develop “well-formed” need and requirement statements, which have the characteristics, employ the templates, and follow the rules discussed in this Guide.

Of course, text is not the only medium by which needs and requirements can be expressed. Alternative methods used to discover and express needs and requirements include:

- operational scenarios, use cases, and user stories (as used as part of Agile development methodologies), or epics, features and stories in a Scaled Agile Framework (SAFe®));
- prototypes, such as used in production-driven and rapid application development methodologies;
- language-based modeling approaches and tools; and
- models and diagrams as part of a modeling approach with well-defined semantics, such as UML for software and Systems Modeling Language (SysML) for generic systems.

Refer to the NRM for a detailed discussion concerning the complementary use of these methods to discover, analyze, and express needs and requirements. However, just as there are issues with any form of technical communication, these other approaches can also be imperfect as they have their own presentational, understandability, traceability, and management challenges. No single tool will, by itself, fix ill-formed needs or requirements.

The key is that the textual form of needs and requirements in conjunction with operational scenarios, use cases, diagrams, and other types of models can support the analysis and development of an integrated data and information model of the SOI. This underlying data and information model captures not only the needs and requirements but also their relationships to each other and to other SE artifacts developed across the SOI lifecycle. This enables all stakeholders to view the data and information in whichever form is best for what they are trying to observe, analyze, communicate, or achieve.

1.5.1 Advantages Of Text-Based Needs And Requirements

For many ideas and concepts that need to be communicated, well-formed, text-based needs and requirements expressed in a structured, natural language have been proven to be the most effective approach (particularly in formal contracting) with a number of advantages.

1.5.1.1 Communication

Despite the increased emphasis on the use of language-based models in systems engineering, there remains a sizable audience who cannot interpret, who do not understand, or who are not willing to work with, diagrammatic or other non-textual representations of need or requirement statements, especially when the formal technical aspects of such representations are not intuitively obvious to the reader.

In particular, some managers, customers, regulators, and users of the system or other non-technical stakeholders, may not have been trained in language-based models or find the terminology used in some diagrams and models confusing and non-intuitive. Even engineers who create visual models (SysML or other) need to be trained on the tool and technique, which may not be a trivial expense.

Forcing stakeholders to learn a specific detailed technical language to describe their needs and requirements may well have them lose interest in the critical definition activities (for additional insight, see NRM, Section 2.10). Consequently, diagrammatic or model representation must be supported by well-formed, textual statements and descriptions for the representations to be understood unambiguously by all stakeholders.

The combination of well-formed need and requirement statements, with visual models can be extremely powerful during analysis and in communicating abstract or hard-to-describe concepts and features. The use of both is encouraged; however, the use of textual statements at some point in the design input definition activities is mandatory.

1.5.1.2 Power of Expression

There is a wide variety of types of needs and requirements that must be expressed. Use cases, user stories, scenarios, diagrams, and models tend to focus on the functional architecture and behaviors expressing functional, performance, and interactions. However, these forms of expression are not presently well suited to expressing non-functional needs and requirements that deal with the physical system elements associated with quality (-ilities), regulations, standards, and physical characteristics. Textual forms using a structured, natural language carry the universal power of expression for all types of needs and requirements outside of physical "build-to-print" drawings and other design representations included in the set of design output specifications.

Two other points illustrate the enduring importance of textual-based need and requirement statements:

- Problem statements, operational scenarios, use cases, user stories should be written from the perspective of the user’s (actor’s) interaction with other actors in the context of the system under development. In contrast, many models describe the system from the perspective of what the system under development must do in order for the users to interact with the SOI in the way they expect. While those forms of expression are excellent conceptual tools for stakeholder expectation elicitation and management, helping to understand the features and associated functionality and performance expected by the stakeholders, they do not always effectively replace well-formed, text-based needs and requirements for the various ideas and concepts that must be communicated, especially what is sometimes referred to as “non-functional” needs and requirements.

- Use cases, diagrams, models, and other alternate forms may not be able to communicate stakeholder needs and stakeholder requirements as effectively as can be done using a text-based, structured language that can be clearly understood by all stakeholders *over* *time*. While these alternate forms are useful tools, they are not sufficient by themselves, without also defining a set of textual needs and requirement statements.

1.5.1.3 Managing Sets of Needs and Requirements

Textual needs and requirements lend themselves to the presentation of large numbers of different types of needs and requirements in an easily digestible form. In contrast, SysML requirement diagrams can present individual requirement expressions but are not well suited to representing multiple or large sets of requirements associated with all the parts of the system architecture.

In addition, the requirement statements defined within modeling tools, often do not have the characteristics of well-formed requirements discussed in this Guide. This limitation of the SysML to enable the visualization of the sets of requirements contained in the system model requires alternate, textual visualizations of requirements expressions as well as sets of requirements. *Note: currently SysML does not include an entity type "need", nor corresponding needs diagrams.*

Note: Tool vendors are now providing tools that allow the linking of requirements contained within language-based models with the same requirements defined and managed within a Requirement Management Tool (RMT). This enables the requirements to remain consistent between tools and allows practitioners to view needs and requirements in whichever form is needed for what they are doing. This also enables the capability for the requirements contained within the language-based models to be well-formed.

1.5.1.4 Accessibility

Even when stakeholders are willing to spend the time to learn modeling languages such as UML and SysML or other language-based modeling tools, these SE tools may not be readily available and assessable to all stakeholders due to a limited number of "licenses" or "seats" or "modeling skills"; while this is a solvable problem, it comes with an increased cost. Obtaining and maintaining the necessary skills takes time and is at an additional cost separate from the tool.

In contrast, most stakeholders are already well versed in the use of common office applications and the natural language used within these applications. Being able to provide textual needs and requirements in an electronic document format (pdf, or common office application formats) allows the stakeholders to view the needs and requirements in common office applications that have been installed on their computers without further training or expense. In addition, there are still stakeholders who will still prefer and demand, printed, text-based documents, and will continue to do so for the foreseeable future.

1.5.1.5 Attributes

Both the need and requirement expressions include a set of attributes that can be used to manage them as well as the system under development across all lifecycle activities. While modeling languages allow users to define an entity having the name "attribute" and link that entity to a need or requirement statement, few practitioners do so, especially when there are multiple attributes that the project team has decided to use and define. Attributes can be a powerful tool to produce meaningful reports and dashboards for use by management stakeholders.

While several MBSE tools are making progress on having a number of attributes for each need or requirement statement, and in linking graphical content (use cases, etc.) to attributes, text-based approaches are often far simpler. Appendix E includes a listing of attributes mapping those attributes to the characteristics defined within this Guide to which they contribute. Refer to the NRM, Section 15 for a more detailed discussion concerning the use of attributes.

1.5.1.6 FORMAL, BINDING AGREEMENT

The legal profession represents perhaps the greatest barrier to adoption of models and non-textual tools for expressing needs and requirements. Textual need and requirement statements are more easily understood in a formal agreement or contract-based system development effort by a wider, and often, non-technical group of stakeholders including business management, project management (PM), configuration management (CM), contract administrators, and legal practitioners.

To be part of a binding agreement, especially in a legal contract, the sets of needs and requirements must be expressed formally, and configuration managed in a form that 1) makes it clear the statements are binding and 2) have the characteristics of well-formed need and requirement statements and sets of needs and requirements as defined in standards and guides such as this Guide.

Use of “shall” in requirements statements, or another term defined to have the same meaning, makes it clear that what is being communicated is formal, the requirement statement is binding, and the system must be verified to meet the requirements.

While drawings and figures can certainly be part of Statements-of-Work (SOWs), it is currently difficult to include a model file as a contractual obligation. Text statements are far easier to enforce legally. Perhaps one day this will evolve; however, until then, the reader is recommended to continue defining well-formed, text-based needs and requirements, particularly in support of contract-based developments.

1.5.1.7 SYSTEM VERIFICATION AND SYSTEM VALIDATION

Most formal contract-based product development and management processes, as well as the highly regulated products themselves, require evidence of successful system verification and system validation. These formal processes must occur prior to product acceptance, qualification, certification, and approval for use.

In highly regulated, safety-critical industries such as the medical device industry, ground transportation, aviation, and consumer products, formal evidence that the design outputs (including the product) meet the design inputs (Integrated Set of Needs and set of Design Input Requirements) is required prior to the product being approved for its intended use in the intended operational environment by the intended users and released into the market.

Currently, no form other than textual needs and requirements has been able to meet these requirements. Even if these textual requirements are entered into a modeling tool, verification is performed against the statement in the requirement-block. For a detailed discussion on system verification and system validation refer to the NRM and GtVV.

1.5.2 SUMMARY

More than anything else, textual needs and requirements are a form of communication. As such, it is vital that the intended message is clearly, and unambiguously communicated to those for whom the message is intended, over time.

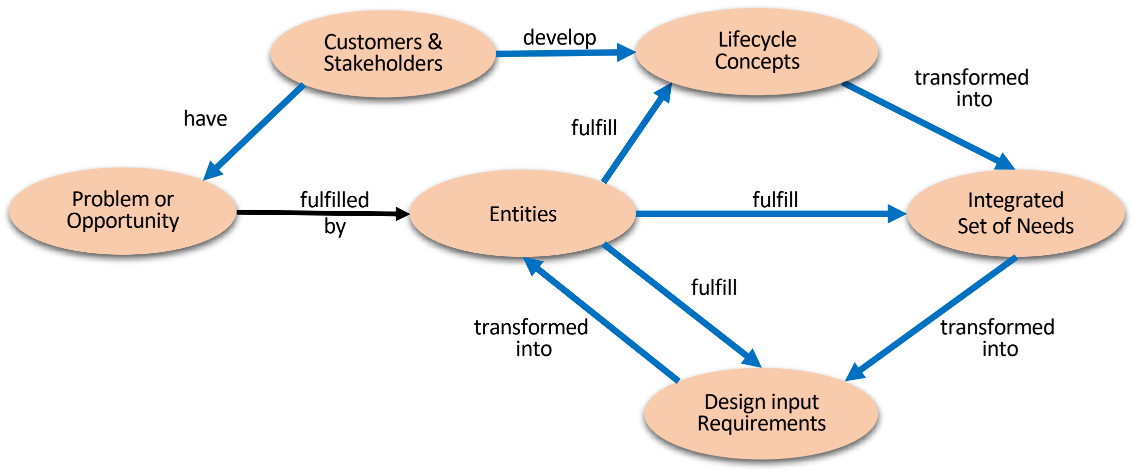
This Guide refers solely to the expression of textual need and requirement statements. If an organization chooses to use alternate forms to define needs and requirements, the characteristics defined in this guide are still applicable. It is important to understand that, from a legal perspective, the responsibility of communicating the message is the responsibility of the sender, so the onus is on the writer for unambiguous communication of needs and requirements. If the alternate form does not have these characteristics, there is a risk that the intent of what is being communicated within the need and requirement statements will not be clear to those they are intended; resulting in the needs and requirements of those for whom the system is being developed will not be met.

If a project has embraced the use of language-based modeling tools to facilitate their approach to systems engineering, the characteristics and rules in this guide still apply. The use of language-based modeling tools does not provide an exemption from performing the needs and requirements definition and analysis activities required to make a project successful, tool or not. In fact, the use of such tools for doing the necessary underlying analysis, is key to helping ensure the needs and requirements have the characteristics and rules defined in this Guide.

1.6 Definitions

This section defines several fundamental terms (Ryan, Wheatcraft, Dick and Zinni; 2014) defined in the context of how they are used within the RWG products. For example, the term “needs” by itself in the RWG products refer to the needs included in a well-formed Integrated Set of Needs and the term “requirements” refer to the requirements contained within a well-formed set of Design Input Requirements having the characteristics defined in this Guide.

When describing system development, some form of distinction is commonly made between lifecycle concepts, needs, and requirements as shown in Figure 3. Lifecycle concepts, needs, and requirements should be developed for entities at all levels of the organization and system architecture. The activities associated with the development and transformations shown in Figure 3 are discussed in depth in the NRM and GtNR.



Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 3: Entity-relationship Diagram for Customers, Concepts, Entities, Needs, and Requirements Terms

Note: In the definitions below, the term “customer” is used to refer to the organizations or persons requesting or procuring a work product, and/or will be the recipient of the work product when delivered. Customers are key stakeholders that exist at multiple levels of an organization and may be internal or external to the enterprise. As such, there can be multiple customers.

1.6.1 Entities

Needs and *requirements* apply to an *entity*, which could exist at any level of the organization or the system architecture in the context of a problem or opportunity fulfilled by the entity. Since terms such as 'product', 'SOI', 'system', 'subsystem', and 'system element' are level-specific, a general term is needed that can apply at any level of the organization or architecture and to any single item at that level. For this, the term 'entity' is used which has lifecycle concepts, needs, and requirements which the entity fulfills.

*An **entity** is a single item to which a concept, need, or requirement applies: an organization, business unit, project, supplier, service, procedure, SOI (system, subsystem, system element), product, process, or stakeholder class (user, operator, tester, maintainer, etc.).*

There are three general types of entities—*physical or software* entities such as the engineered systems to be developed; *process* entities such as procedures or work instructions; and *business or human* entities such as business units, users, customers, developers, suppliers, and other stakeholders.

1.6.2 Concepts

*A **concept** is a textual or graphic representation that concisely expresses how an entity can fulfill the problem, threat, or opportunity it was defined to address within specified constraints with acceptable risk that provides a business in terms of people, process, and products.*

*A set of **lifecycle concepts** includes multiple concepts across the lifecycle for how the organization (and stakeholders within an organization) expects to manage, acquire, define, develop, build/code, integrate, verify, validate, transition, install, operate, support, maintain, and retire an entity.*

Lifecycle concepts can be defined from the perspective of the organization, the SOI, or the macro-system in which the organization and SOI exist. Refer to the NRM for more details concerning lifecycle concepts.

As discussed in the INCOSE SE HB lifecycle stages for a SOI include concept, development, production, utilization, support, and retirement. From a holistic perspective, concepts concerning the approaches the project team will implement these lifecycle stages must defined early in the project both from a project management perspective as well as a systems engineering perspective.

1.6.3 Needs

Needs are well-formed textual statements of expectations for an entity stated in a structured, natural language from the perspective of what the stakeholders need the entity to do, in a specific operational environment, communicated at a level of abstraction appropriate to the level at which the entity exists.

*A **need statement** is the result of a formal transformation of one or more sources or lifecycle concepts into an agreed-to expectation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.*

Based on the sources and set of lifecycle concepts, through formal needs analysis, needs are defined using a formal transformation process involving decomposition, derivation, elaboration, diagrams, and architectural and analytical/behavioral models.

Note: The use of the term "needs" as used in this Guide refers to the needs contained within an Integrated Set of Needs defined for the SOI as discussed in Sections 1.8.

1.6.4 Requirements

Requirements are well-formed textual "shall" statements that communicate in a structured, natural language what an entity must do to realize the intent of the needs from which they were transformed.

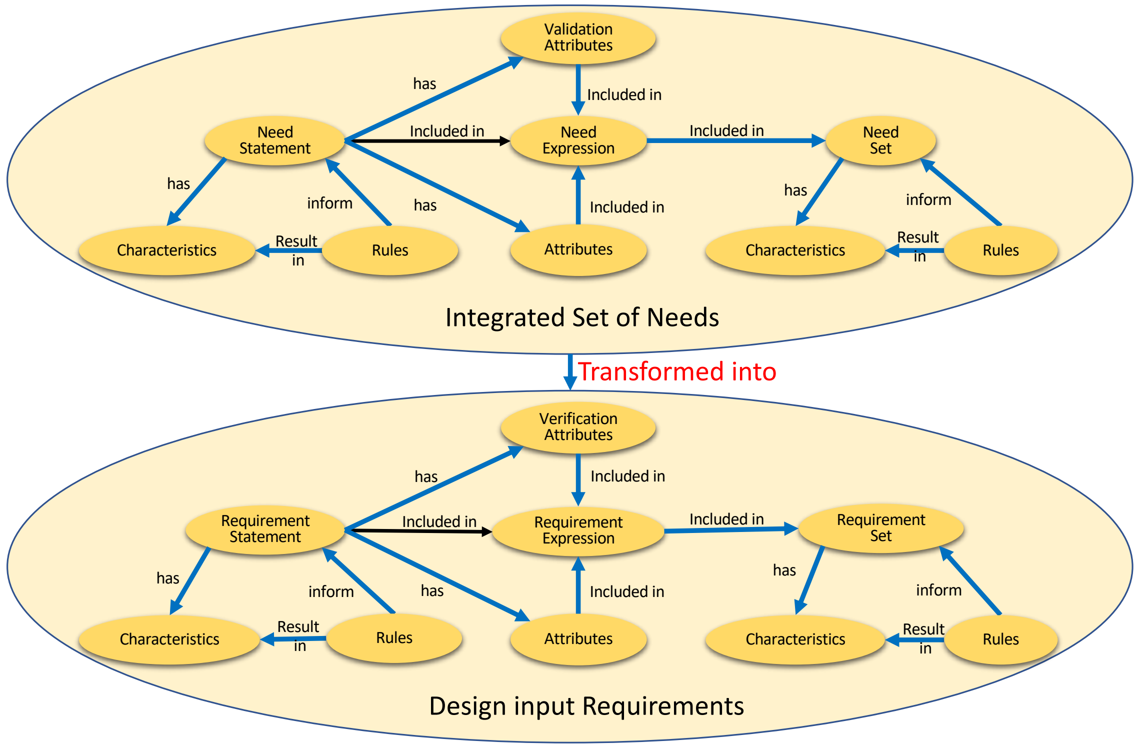
*A **requirement statement** is the result of a formal transformation of one or more sources, needs, or* higher-level *requirements into an agreed-to obligation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.*

Requirements definition is an activity which, through formal requirements analysis, determines specifically what the entity must do to meet the sources, needs, or higher-level requirements they are being transformed from using a formal transformation process involving decomposition, derivation, elaboration, diagrams, and architectural and analytical/behavioral models.

Note: The use of the term "requirements" as used in this Guide refers to the requirements contained within a set of Design Input Requirements defined for the SOI as discussed in Sections 1.8.

1.6.5 Attributes

As illustrated in Figure 4, need and requirement statements are supported by associated attributes that aid in the definition and management of a need, sets of needs, a requirement, and sets of requirements. These attributes also aid in achieving many of the characteristics defined in this Guide.



Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 4: Entity-Relationship Diagram for Needs and Requirements Terms

*An **attribute** is additional information associated with an entity which is used to aid in its definition, understanding, and management.*

Well-chosen attributes, properly defined and tracked, can make the difference in being able to correctly interpret and manage needs and requirements definition throughout the system lifecycle and adjust accordingly—or finding out late in the program the needs or requirements were defective in the first place. When errors in the needs and requirements—or errors in their interpretation—are discovered late in the program, they can be expensive and time-consuming to fix.

Appendix E includes a listing of attributes mapping those attributes to the characteristics defined within this Guide to which they contribute. *Refer to the NRM for a more detailed discussion on attributes as applied to needs and requirements.*

1.6.6 Needs and Requirement Expressions

Although each need statement and requirement statement are individually important, needs expressions and requirements expressions are more than just well-formed textual statements that are written succinctly in a standard format having the characteristics defined in this Guide. The full expression comprises a statement and its associated supporting attributes.

*A **need expression** includes a need statement and a set of associated attributes.*

*A **requirement expression** includes a requirement statement and a set of associated attributes.*

1.6.7 Patterns

Patterns represent the structure that every well-formed need or requirement statement should conform with.

*A **requirement pattern** or **need pattern** is represented by a series of building blocks (also called pattern slots) including all the elements envisioned to represent a well-formed, singular, and complete need or requirement.*

Several rules, especially R1, are related to the necessity, for needs and requirements, to conform with one and only one pattern. Appendix C provides more information on the concept of pattern and includes some well-known examples.

1.6.8 Sets of Needs and Sets of Requirements

Although each individual need and requirement expression is important, it is ultimately the set of needs and resulting sets of requirements that will describe what the entity must do and be, and it is the set of needs and/or set of requirements that most often will be agreed-to as a contractual obligation.

*A **need set** is a well-formed set of agreed-to need expressions for the entity (enterprise/business unit/system/subsystem/system element/process) and its external interfaces.*

Within the NRM, GtNR, GtVV, and this Guide this set of needs is referred to as an **Integrated Set of Needs** as shown in Figures 2 and 3. This Integrated Set of Needs is well-formed, having the characteristics defined in this Guide, communicating the scope of effort to which the system of interest will be validated against.

*A **requirement set** is a well-formed set of agreed-to requirement expressions for the entity (enterprise/business unit/system/subsystem/system element/process) and its external interfaces.*

Within the NRM, GtNR, GtVV, and this Guide this set of requirements is referred to as a set of system **Design Input Requirements** as shown in Figures 2 and 3. This set of system Design Input Requirements is well-formed, having the characteristics defined in this Guide and against which the SOI will be verified.

1.7 Needs, Requirements, and the Entity to Which They Apply

When defining needs and requirements it is important to understand the significance of the use of the word “entity”.

In terms of need and requirement statements, the entity that is the *object* of a need statement (“The <stakeholders> need the <entity> to ....”) which, in turn, will be the *subject* of a subsequent requirement statement (“The <entity> shall ......”) transformed from that need.

For example, the need for an entity called a bicycle may incorporate the constraint, "The customers need the bicycle to transport the rider using non-motorized power." This would be transformed into one or more requirements on the bicycle system that would result in the need being met, one such requirement could be, "The bicycle shall transport a rider using the propulsion force supplied by the rider.”

Systems engineers must make clear the applicability of needs and requirements based on the entity they apply. For example:

There will be *project needs and requirements* on a project or organizational elements within the enterprise that will be recorded in a Project Authorization Document (PAD), Project Management plan (PMP), other plans and procedures that may take the form:

The <stakeholders> need the <project> to …….
The <stakeholders> need the <xxxx team> to …….

The <project> shall …….
The <xxxxx team> shall…….

There will be *supplier needs and requirements* on a supplier, vendor, or contractor that will be recorded in Statements of Work (SOW) and Supplier Agreements (SA) that may take the form:

The <stakeholders> need the <supplier/contractor> to …….

The <supplier/contractor> shall …….

There will be *procedural needs and requirements* concerning actions the person or organization responsible for conducting steps within a procedure resulting in those actions (for example, a system verification or system validation procedure or a test procedure), such as the following:

The <stakeholders> need the <operator, technician, engineer> to verify the SOI meets <this requirement> per the defined success criteria using the defined system verification approach using the defined system verification method.

The <operator, technician, engineer> shall <stimulate the SOI in some manner>.
The <operator, technician, engineer> shall <record the results of the stimulation>.

There will be *system needs and requirements* on the SOI that are recorded within the SOI set of Design Input Requirements or design output specifications as shown in Figures 2 and 4, which provide expectations concerning the design and production of a SOI:

> The <stakeholder> needs the <SOI > to provide the capability to <do something in a given operational environment with a required performance of xxxxx>.
>
> The <SOI> shall <perform some function with the desired performance under some operating condition>. (Design input)
>
> The <SOI component> shall be manufactured to <the physical dimensions shown in drawing xyz?>. (Design output).

Needs and requirements for the different entities must not be mixed together within a single set of needs and a single set of requirements. Each set of needs and each set of requirements must only include statements that relate to the single entity to which the set applies. This is important because for each entity there is an expectation that objective evidence can be obtained which can be used to assess, with some level of confidence, that the entity has met the requirements (system verification) or needs (system validation) of that entity. If the sets are blended system verification and system validation can be much harder, which can lead to confusion among project staff. See also R3.

1.8 Needs vs. Requirements

1.8.1 Stakeholder Expectations, Needs, and Requirements

Various guides, textbooks, and standards refer to stakeholder "expectations, needs, and requirements", "stakeholder needs and requirements", or "user needs and requirements" as if they are the same; often combining them in a single document, such as a User Requirements Document (URD), Program [or Project] Requirements Document, or Stakeholder Requirements Document (StRD)—communicating the needs and requirements in the form of "shall" statements.

This can result in confusion as to their scope, in terms of what an is "expectation" versus what is a "need" as opposed to what is a "requirement". All three can have different contractual meanings—and the end SOI can be different than what the customer expected if all three are not clarified and agreed to, as to what idea, function, capability, feature, or concept is in which category. Some organizations communicate stakeholder expectations in terms of user stories, use cases, user scenarios, operational scenarios, Concepts of Operation (ConOps) or Operational Concepts (OpsCon).

For some practitioners, "stakeholder expectations" are communicated in terms of stakeholder needs and requirements; for example, ISO/IEC/IEEE 29148 [25] defines "stakeholder requirements" in terms of "stakeholder needs."

ISO/IEC/IEEE 29148 states that **stakeholder needs** concentrate on the system's purpose and behavior and are described in the context of the operational environment and conditions. ISO/IEC/IEEE 29148 does not specify any particular form for communicating stakeholder needs—indeed, it states that they often lack definition, analysis, and possibly consistency and feasibility.

Stakeholder requirements are defined as a more formal form of stakeholder needs, transformed from the stakeholder needs, no matter their form, refining and evolving the stakeholder needs using a ConOps to aid the understanding of the stakeholder concerns at the organizational level and system OpsCon from the system perspective. Ideally, this transformation results in a set of objectively adequate, structured, and more formal statements of stakeholder needs, goals, and objectives. As requirements, ISO/IEC/IEEE 29148, states that the stakeholder requirements should be well-formed having the characteristics of well-formed requirements.

The purpose of stakeholder needs and requirements is to represent a user-oriented view of the system as viewed externally; what do the stakeholders need from the system, what do they need the system to do, how do they plan to interact with the system, and what interactions the system has with its external environment.

As the phrases suggest, stakeholder needs and stakeholder requirements are defined, managed and owned by the stakeholders or their representatives. This can be confusing in that customers, users, operators, regulators, and developers are all potential stakeholders.

Whenever the phrase "stakeholder, user, or customer needs or requirements" is used, the reader should assume what is meant are needs or requirements owned by the stakeholders that defined them. In some domains, "user needs and requirements" and "customer needs and requirements" are specifically addressed instead of, or in addition to, the more generic "stakeholder needs and requirements" designation. Different organizations may use slightly different terms for what may or may not be the same concept or entity.

1.8.2 System Requirements

ISO/IEC/IEEE 29148 states that **system requirements** specify, in a way that is measurable, from the supplier's perspective, what characteristics, attributes, and functional and performance requirements the system is to possess, to satisfy stakeholder requirements *[needs]*. Unless the customer has placed a constraint on the system, the system requirements should be defined in logical terms and not imply any specific physical implementation.

The system requirements are transformed from the stakeholder, *user-oriented view* of desired capabilities into a *technical view* of a solution that meets the operational *needs of the user*.

The system requirements are system-level technical requirements defined and owned by the stakeholders responsible for the transformation, communicating what the system of interest must do to meet the needs. Who the stakeholders are can vary depending on the organization's development concept. They could be the customer (internal or external - depending on perspective) or the developing organization (internal or external - again depending on perspective). If the system requirements are defined by the customer, these represent the customer-owned system requirements. If defined by the developing organization, these represent the developer-owned system requirements.

The customer-owned or developer-owned system requirements are meant to be the system requirements that are inputs into the system architecture and design process activities.

In some cases, the customer may have done the design and what are referred to as customer requirements are in fact the customer-owned design output specifications to which the SOI is to be manufactured or coded to.

This distinction is important in that "stakeholder, user, or customer needs" and associated stakeholder, user, or customer requirements represent a *stakeholder, user, or customer perspective* of what they *need* the SOI to do as viewed externally, while the "customer-owned" or "developer-owned system requirements are a *technical perspective* that communicates what the stakeholders, users, or customers *require the SOI to do to meet their needs*.

Often, the requirements from the customer included in a contract with a supplier are “customer-owned system requirements” or “customer-owned design output specifications”.

1.8.3 Common Issues

Below are common issues frequently observed within sets of stakeholder needs and requirements and sets of customer-owned system requirements:

- Stakeholder needs and requirements are treated as the same thing, including both in the same set—for example, User Needs Document or User Requirements Document or Stakeholders Requirements Document.
- While stakeholder needs are referred to as a “thing”; the form in which they are communicated is not defined. Some forms used by stakeholders to communicate their needs include text, use cases, user stories, operational scenarios, and voice (during elicitation). There is currently no defined standard form for documenting stakeholder needs in existing standards.
- There are multiple stakeholders as well as different sets of stakeholders at different levels of the organization and system architecture, resulting in multiple sets of stakeholder needs and stakeholder requirements. The different sets can contain needs and requirements that conflict, or at least be inconsistent with, other sets of stakeholder needs and stakeholder requirements.
- As stated in ISO/IEC/IEEE 29148, stakeholder needs are often not well formed, and there is no standard for the form in which the stakeholder needs are communicated. With this in mind, what are stakeholder requirements transformed from the stakeholder needs validated against?
- Customer/stakeholder requirements, and customer/stakeholder-owned system requirements are often poorly formed and do not have the have the characteristics of well-formed requirements. In addition, the sets are often incomplete not including all system requirements that reflect the stakeholder needs and requirements of the developing organization.
- Not all stakeholders transform their needs into stakeholder requirements and may use alternate forms to communicate their expectations.
- Often stakeholder needs, stakeholder requirements, and stakeholder-owned system requirements have not undergone sufficient analysis to determine feasibility.
- Stakeholder requirements and system requirements communicate different perspectives yet are expected to have the same characteristics of well-formed requirements; which means that it is often difficult to determine the differences among the various sets, what role they play, and expectations for system verification and system validation. In this case it is difficult to tell the difference if both start with “The <system> shall …….”
- As requirements, it is common for organizations to not understand and not make a distinction between stakeholder requirements and system requirements, including them in the same set. To them they are both “requirements”. This is problematic in that ISO/IEC/IEEE 29148 and ISO/IEC/IEEE 15288 state that the system is verified against the system requirements and validated against the stakeholder requirements. If the stakeholder requirements are in the same set with the system requirements, and indistinguishable from the system requirements, what the system validated against.
- Formal lifecycle concepts and needs definition activities are sometimes avoided or abbreviated. Skipping lifecycle concepts and needs definition before defining system requirements put an engineering organization at risk of misconstruing the intent of the SOI

which may result in development of a system that fails system validation and creates potential rework.

- When recording needs, the need statements are written as "shall" statements and calling these "stakeholder requirements" rather than needs.
- Stating stakeholder needs and stakeholder requirements that are too vague to verify or validate against.
- Often "stakeholder requirements" do not have the characteristics of well-formed requirements as defined in this Guide, even though ISO/IEC/IEEE 29148 says they should.

Examples:

"The <stakeholders> *need* the <SOI> to be compliant with government safety standards and regulations."

This stakeholder need is communicated as a stakeholder requirement:

"The <SOI> *shall* be compliant with government safety standards and regulations."

Although well-intended, both statements are too vague to verify or validate the system against because each is ambiguous as to which specific standards and regulations. *Restating a poorly formed stakeholder need as a stakeholder requirement with a "shall" does not result in a well-formed requirement.*

For a well-formed stakeholder need statement and resulting stakeholder requirement, the project team would need to complete an engineering analysis to determine which specific standards and regulations apply and to determine and communicate their needs concerning with which specific standards and regulations the SOI must comply.

An example resulting stakeholder need and corresponding stakeholder requirement *[need]* would then state:

"The <stakeholders> *need* the <SOI> to be compliant with government safety regulation <xyz, section 1.2.3."

"The <SOI> *shall* be compliant with government safety regulation <xyz, section 1.2.3."

However, what often happens is that this stakeholder requirement is copied and pasted into the set of system requirements as written.

Another common issue is when needs are written using a "shall", but the entity is a stakeholder rather than the SOI. For example, there may be a need:

"The <stakeholders> need the <SOI> to provide the capability to enable <users> to <perform a given function >."

A common error is to rewrite the need statement as a system requirement without changing the subject of the statement (the appropriate entity) stating that:

"The <user> shall be able to <perform a given function>."

This is not an appropriate form for a requirement on the SOI, in that the entity in the subject of the revised statement is the user, rather than the SOI. In addition, there is no mention of the conditions in which the function is to be performed, nor the required performance characteristics associated with the performance of the function.

Alternatively, another common issue is to "transform" the stakeholder need into a stakeholder requirement by simply making the SOI the subject of the statement:

> "The <SOI> shall provide the capability to enable <users> to <perform a given function >." or
> "The <SOI> shall enable <users> to <perform a given function >." or
> "The <SOI> shall be able to <perform a given function>."

While the form may be acceptable from a need perspective, in each case, the resulting stakeholder requirement does not have the characteristics of well-formed requirements - much more analysis is required to write an appropriate well-formed requirement statement than to simply change the subject of the need statement.

The project team would need to undertake an engineering analysis to determine what the system must to do such that the intended users will have the capability to perform the given function, under the given conditions, with the required performance.

Based on that analysis, the project team would then write one or more system requirements on the SOI that, when implemented by the design, would result in the user being able to perform the function stated in the need statement under the stated conditions with the required performance.

1.8.4 Terminology Used In This Guide

Because of the above issues, stakeholder real-world expectations in the form of stakeholder needs and requirements (or other forms of expression), problem, mission statement, goals, objectives defined within the *Business or Mission Analysis Process* and the *Stakeholder Needs and Requirements Definition Process* activities described in ISO/IEC/IEEE15288 and the INCOSE SE HB are treated as inputs into the lifecycle concepts analysis and maturation and needs definition activities conducted at the project/system level as discussed in the NRM and GtNR as shown in Figure 5.

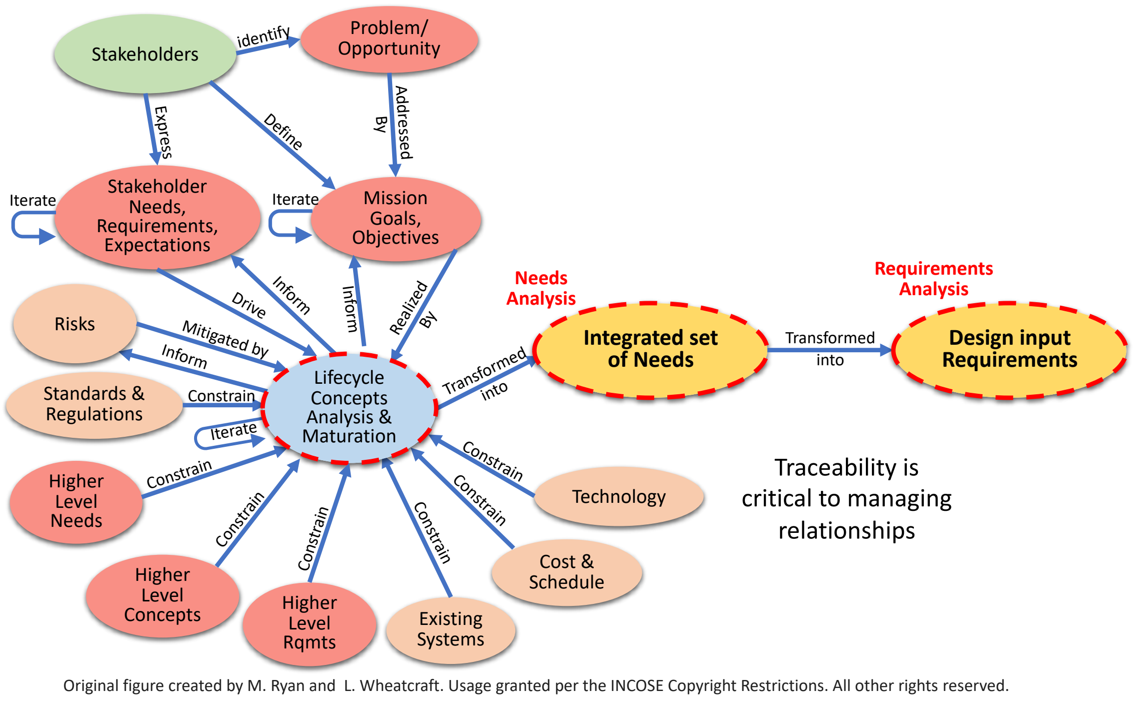


Figure 5: Inputs to the Lifecycle Concept and Needs Definition Activities

Stakeholder/customer-owned system requirements are also treated as higher level requirements that are inputs into the lifecycle concepts analysis and maturation and needs definition activities.

By performing the activities discussed in the NRM and GtNR, the result is a set of feasible lifecycle concepts. The set of lifecycle concepts and the other sources are transformed into an Integrated Set of Needs that are correct, consistent, complete, and feasible. The Integrated Set of Needs is then transformed into a set of system level Design Input Requirements that are correct, consistent, complete, and feasible.

1.8.4.1 NEEDS AND REQUIREMENTS PERSPECTIVES

The approach taken in the NRM, GtNR, and this Guide, which is consistent with ISO/IEC/IEEE 29148, is that the Integrated Set of Needs is written from the perspective of the what the *stakeholders need to do* using the SOI, while the Design Input Requirements that are transformed from the needs are written from the perspective of *what the SOI must do* to meet the need(s) from which they were transformed. Figure 6. illustrates these different perspectives.

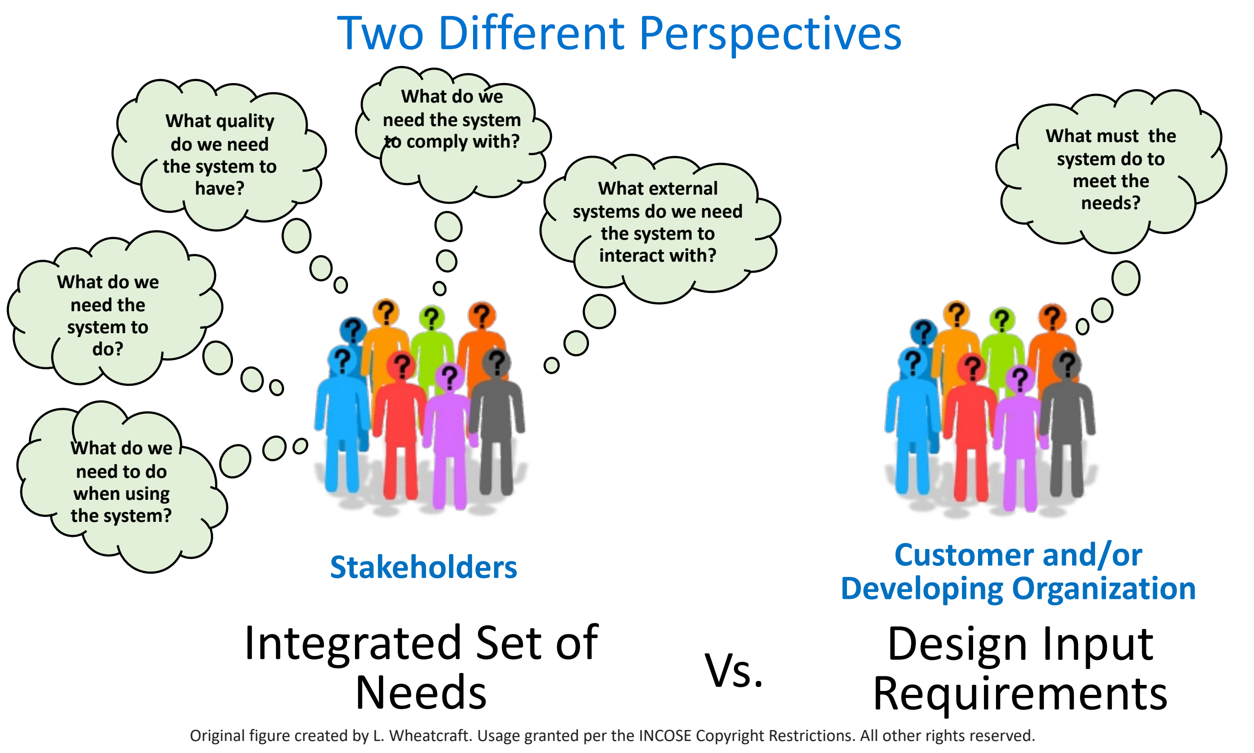

Two Different Perspectives
What quality do we need the system to have?
What do we need the system to comply with?
What external systems do we need the system to interact with?
What do we need the system to do?
What do we need to do when using the system?
What must the system do to meet the needs?
Stakeholders
Customer and/or Developing Organization
Integrated Set of Needs
Vs.
Design Input Requirements
Original figure created by L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 6: Needs vs Requirements - Different Perspectives

1.8.4.2 INTEGRATED SET OF NEEDS

The Integrated Set of Needs represents an integrated stakeholder, customer / acquirer view of the SOI, communicating the perspective of the SOI as viewed externally by the stakeholders (black box).

The Integrated Set of Needs communicates the agreed-to, baselined, and configuration-managed stakeholder real-world expectations concerning the system's intended use, intended users, and intended operating environment against which the system will be validated. **It is this well-formed Integrated Set of Needs that defines the scope of the project.** While the individual need statements do not include the word "shall", the Integrated Set of Needs, as a set, is invoked on the project as a "shall" within the project authorization document or statement of work, as applicable.

This Integrated Set of Needs is transformed into the system-level Design Input Requirements by the organization responsible for doing so. Who the organization is can vary depending on the organization's development concept—customer (internal or external—depending on perspective) or the developing organization (internal or external—again depending on perspective).

Unlike the stakeholder needs discussed earlier, the Integrated Set of Needs is well formed having the characteristics described in this Guide.

A key point is that the Integrated Set of Needs is owned and managed by the customer (the person or organization that owns the problem and is paying for the SOI). ***It is only the customer that has a right to define their needs — not the suppliers nor the developers.*** However, the definition of the Integrated Set of Needs is an iterative process involving the stakeholders— including the users, operators, suppliers, developers, and other key stakeholders as appropriate. Although in many cases this is subcontracted to a third-party or even the supplier, in all cases the customer must retain responsibility for agreeing the needs.

Project success and system acceptance and approval for its intended use is dependent on the system passing system validation as defined by the Integrated Set of Needs.

1.8.4.3 System Design Input Requirements

Because the Design Input Requirements are transformed from the Integrated Set of Needs, their quality is dependent on the quality of the needs from which they are transformed. If the Integrated Set of Needs does not have the characteristics defined in this Guide, nor will the resulting Design Input Requirements.

To avoid ambiguity, when writing needs and requirements, the reader is cautioned to be aware of the differences between terms, gain agreement among the stakeholders including the developing organization, and clarify the role each has in the SOI lifecycle activities.

This is important because, if the project makes changes in any way midstream—either to get back on schedule, incorporate new needs from the stakeholders, or descoped to reduce cost or schedule—it will be much easier to do this when the Integrated Set of Needs is clearly defined separate from the requirements. Deleting a requirement is one thing—but accidentally removing a need that looks like a requirement can be far more serious, resulting in failed system validation.

To help distinguish needs from requirements, the needs statements should not include the word "shall". Using "shall" in need statements results in confusion as to whether the statement is a need, or whether it is a requirement transformed from a need.

When communicating needs, one approach to avoid this confusion is to use the format:

"The <stakeholders> *need* the system to …" or "The <stakeholders> *need* to …"

Some organizations may use other verbs such as "would like", "prefer", "should", or "may" to convey priority or criticality—it may be better, however, to always use "need or need to" in the need statement and use the attributes of the need expression to convey priority or criticality and to make clear what is mandatory and against which the realized system will be validate against.

When communicating requirements, the requirements have the same general form but use a different verb phrase, which always includes the word “shall” for a requirement statement:

“The <SOI> *shall* …”

Requirement statements use the word “shall” to make clear they are requirements. Some organizations may use other verbs such as “must”, “will”, or ‘should”, “may” to convey priority or criticality—it may be better, however, to always use “shall” in the requirement statement and use the attributes of the requirement expression to convey priority or criticality and to make clear what is mandatory, binding, and against which the system will be verified.

Using these distinct formats helps make clear what is a need statement and what is a requirement statement.

An example of the **approach taken in the RWG products** is to include in the Integrated Set of Needs, agreed-to, well-formed need statements such as:

“The <stakeholders> *need* the <SOI> to comply with <government safety regulation xyz, section 1.2.3>.”

In the past, rather than defining and getting alignment and agreement on the Integrated Set of Needs, many would communicate this need as a requirement “shall” statement and include it in the set of system level Design Input Requirements.

“The <SOI> *shall* comply with <government safety regulation xyz, section 1.2.3>.”

However, this is not a good practice because 1) it is not well-formed, not having the characteristics defined in this Guide and 2) an analysis must be done, and resulting requirements defined, concerning what the system must do to meet the intent of the need.

What should happen when transforming the need into corresponding system level requirements, is that the responsible project team for the transformation would do an engineering analysis assessing each requirement in the standard:

- Which standards and regulations are applicable for the SOI (as opposed to the organization developing the SOI).
- Which requirements within these standards and regulations apply to the specific SOI under development.
- Whether the requirements within these standards and regulations are design inputs or requirements on the design team.

Based on that analysis, the project team would then:

- Derive specific Design Input Requirements that, when implemented by the design, will result in the intent of the parent requirements within the regulation referenced in the need to be met.
- For those standard or regulatory requirements that are design outputs, derive specific design output specifications that when implemented by the realized system, will result in the intent of the parent requirement within the regulation or standard referenced in the need to be met.
- Ensure there is traceability to the needs concerning the standard as well as traceability as the design input requirement is allocated to lower-level system elements for implementation.
- Define system verification attributes (success criteria, method, approach).

Refer to the NRM and GtNR for additional guidance on defining needs and requirements that address requirements within standards and regulations.

Further examples concerning the differences between how needs and requirements are communicated are included in the descriptions of the characteristics and rules in this Guide as well as the NRM.

Refer to the NRM Section 4 for a detailed discussion on the activities associated with defining an Integrated Set of Needs for an SOI, and Section 6 for a detailed discussion on transforming those needs into a well-formed set of Design Input Requirements. Figure 5 shows a graphical representation of this transformation.

1.9 Quality of Needs and Requirements

Defining needs and requirements is not an exercise in writing, but an exercise in engineering. Well-formed needs and requirements having the characteristics defined in this Guide will result from following the rules defined in this Guide *as well as* performing the activities associated with the definition of the needs and requirements and inclusion of attributes as discussed in the NRM and GtNR.

The underlying analysis from which a need or requirement was derived is as important as how well the need or requirement statement is formed. Each is necessary, but not sufficient on its own, both are necessary to achieve the characteristics defined in this Guide.

A requirement can be well-formed following the rules in the GtWR, yet not have the characteristics defined in the GtWR—it may not be correct nor feasible.

1.9.1 Verification Of The Needs And Requirements

As discussed in Section 1.10, needs verification and requirements verification involves verifying that the need statements and requirement statements and sets of needs and sets of requirements have the characteristics of well-formed statements and sets of statements resulting from following the rules such as those in this Guide or similar organizational guide or standard. This may differ from the common misconception that "verifying requirements" involves ensuring that a requirement is satisfied by a realized SOI— however, that activity is referred to as “system verification” rather than “requirements verification”.

Verification of needs and requirements is what is commonly referred to as assessing the quality of the need and requirement statements and sets of needs and requirements. Refer to the NRM, Section 5 for a more detailed discussion on needs verification and the NRM Section 7 for a more detailed discussion on requirements verification. Some would call this a "quality assurance” check on the needs and requirements statements (and their sets).

While some needs verification and requirements verification activities must be done manually, others may be able to be automated depending on the capabilities of the applications in the project toolset. To what extent the project can rely on automated verification depends on a tradeoff between effort and risk. The need for, and importance of, using automation for needs and requirements verification will increase as systems become more complex and the number of needs and resulting requirements increase.

NLP/AI applications provide the capability to automate needs and requirements verification to some extent. Many of these tools use a subset of the rules defined in this Guide as a basis for assessing the quality of the need and requirement statements and integrated sets of needs and requirements. These tools can be used as both a “digital assistant” to aid in the writing of need and requirement statements as well as to assess the quality of individual need and requirement statements and set of needs and requirements based on the subset of rules used.

Currently many of these tools focus on the quality of requirement statements and not on the quality of need statements. Hopefully, in the future, the tools will address both need statements and requirement statements and understand the differences between the two. Another current limitation is that some of these tools require the requirements be transferred over to the vendor's servers, which may limit their application on government or military programs.

Many of the NLP/AI applications provide a "score" concerning the quality of the requirement statements and sets of requirements based on criteria defined by the project team, as well as identify specific defects based on the subset of rules implemented within the tool or selected by the user. The project team will determine how this score will be used in the definition and management of the requirements and requirements verification activities, including whether certain gated entry criteria will include a certain score for passing the gate or review.

While these applications can identify defects, the project team still must do the analysis to address and correct these defects. Given these applications do not address all the characteristics and rules within this Guide, the project team will still need to address defects associated with the other rules not addressed by the NLP/AI applications within their toolset. In addition, many of the needs verification and requirements verification activities discussed in the NRM and GtNR are more activity based (such as analysis, traceability, and allocation), and thus must be done manually by the project team to ensure correctness, completeness, and consistency and to give the team a head start to address defects associated with these activities.

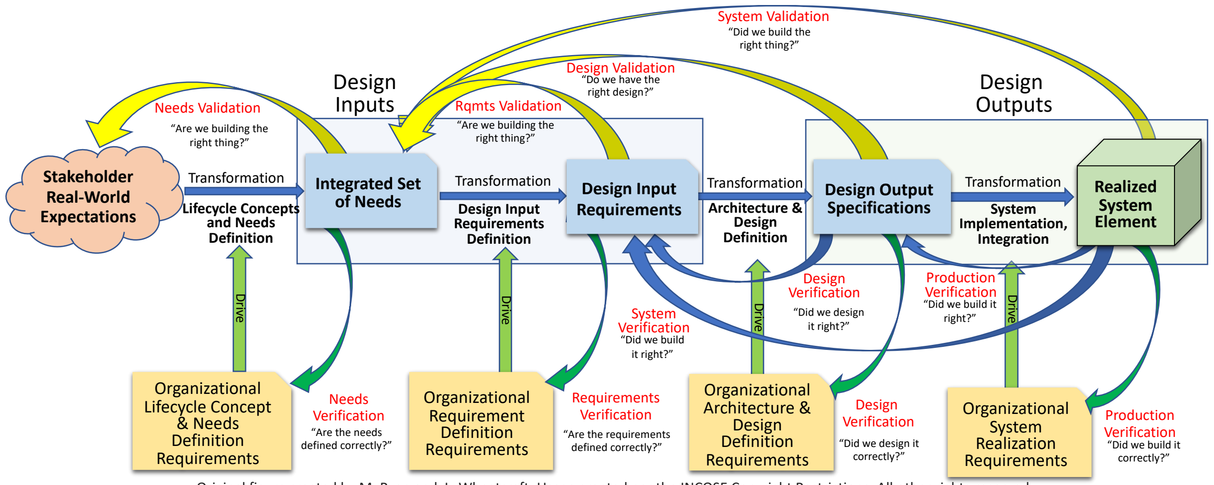
1.9.2 Validation Of The Needs And Requirements

Validation of a need statement determines whether a need statement clearly communicates the intent of the lifecycle concepts or source from which it was derived or transformed. Validation of a requirement statement determines whether the requirement statement clearly communicates the intent of the need, higher-level requirement, or source from which it was derived or transformed. In short, the correct need or requirement statement was authored, regardless of its grammatical correctness, such that it has the characteristics defined in this Guide.

A need or requirement statement can be well-formed but can speak to the wrong need or requirement. The need or requirement may contain incorrect values, it may not be feasible, or it may not communicate the true intent of the lifecycle concept, need, or higher-level requirement from which it was derived or transformed. This would be a defect, which could result in a defective set of implementing requirements as well as in failed system validation. Refer to the NRM Section 5 for a more detailed discussion on needs validation and Section 7 for a more detailed discussion on requirements validation.

Unlike needs verification and requirements verification, needs validation and requirements validation *confirm that the intent of the source is effectively communicated*. This cannot be done without the project team doing the analysis manually—currently none of the NLP/AI applications have the capability to do this type of analysis.

1.10 Needs and Requirements in the Context of Verification and Validation


System Validation
"Did we build the right thing?"
Design Validation
"Do we have the right design?"
Design Inputs
Design Outputs
Needs Validation
"Are we building the right thing?"
Rqmts Validation
"Are we building the right thing?"
Stakeholder Real-World Expectations
Transformation
Lifecycle Concepts and Needs Definition
Integrated Set of Needs
Transformation
Design Input Requirements Definition
Design Input Requirements
Transformation
Architecture & Design Definition
Design Output Specifications
Transformation
System Implementation, Integration
Realized System Element
Drive
Drive
Drive
Drive
System Verification
"Did we build it right?"
Design Verification
"Did we design it right?"
Production Verification
"Did we build it right?"
Organizational Lifecycle Concept & Needs Definition Requirements
Needs Verification
"Are the needs defined correctly?"
Organizational Requirement Definition Requirements
Requirements Verification
"Are the requirements defined correctly?"
Organizational Architecture & Design Definition Requirements
Design Verification
"Did we design it correctly?"
Organizational System Realization Requirements
Production Verification
"Did we build it correctly?"

Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 7: Needs and Requirements in the Context of Verification and Validation

When formulating needs and requirements it is important to understand the role of those statements within the context of verification and validation across the lifecycle.

Once the Integrated Set of Needs is verified and validated, all subsequent artifacts are validated against the needs, as shown in the top part of Figure 7. In turn, once the resulting Design Input Requirements are verified and validated, all subsequent artifacts are verified against those Design Input Requirements, as shown in the lower part of Figure 7.

Refer to the NRM Section 2 for a detailed discussion concerning verification and validation across the lifecycle and the context in which the terms are used.

1.11 Conditional Clauses and Ubiquitous Requirements

As defined in ISO/IEC/IEEE 29148, a requirement is a statement which translates or expresses a need and its associated constraints and conditions. Where a condition is "a measurable qualitative or quantitative attribute that is stipulated for a requirement and that indicates a circumstance or event under which a requirement applies." Conditions further qualify a requirement, enabling a well-formed requirement to be defined in a manner that supports verifiability, and may limit the implementation in design.

ISO/IEC/IEEE 29148 provides the following examples:

- When signal x is received [condition], the <system> [subject] shall set [action] the signal received bit [object] within 2 seconds [constraint of action].
- At sea state 1 [condition], the Radar System [subject] shall detect [action] targets [Object] at range out to 100 nautical miles [constraint of action].

Requirements without a conditional clause are referred to as "Ubiquitous" requirements which are requirements that do not require a condition or context which triggers the desired <system> response and which are always applicable (for example, physical characteristics and quality requirements (-ilities)).

When defining requirements describing an action to be performed by a system, subsystem or system element, it is a common error to treat all requirements as ubiquitous; failing to clearly communicate the condition in which the requirement applies or context which triggers the desired system response. Even some physical properties, such as the power consumption, might vary from one state/mode to another (the power consumption in the power-saving mode would invariably be lower than the consumption in nominal mode), resulting in non-ubiquitous requirements. This can result in requirements that are not Unambiguous (C3), Complete (C4), Correct (C8), nor Verifiable/Validatable (C7).

In many cases at the system level the focus is on general capabilities the system is to provide. The use of ubiquitous requirements in this context is often appropriate at this level. However, as these requirements flow down (are allocated) to lower-level subsystems and system elements, the inclusion of specific conditional statements is warranted addressing states and modes in which requirements are applicable as well as triggering events to which a requirement is responding.

Whenever states, modes, and trigger definitions or transitions are involved, they must be defined in a project dictionary (or ontology, or model) ensuring their consistent use throughout the sets of requirements and other SE artifacts.

In this context, to ensure consistency, the terminology used in the sets of requirements must be the same as is used within the models and design output artifacts. The use of only defined terminology in conditional clauses aids in consistency of requirements and the models they were derived. Refer to appendix D for a summary of those rules in this guide that require, to some extent, a project dictionary to be evaluated.

Addressing state definitions and transitions within the conditional clauses also enables the development of simulations of the system behavior.

This concept has been commercialized in approaches such as “EARS” (Easy Approach to Requirements Syntax) (Mavin 2009) and within several NLP/AI tools that are used to assess requirements quality.

This Guide addresses the use of conditional clauses in several of the rules:

- R1 – Structured Statements
- R11 – Separate Clauses
- R18 – Single-Thought Sentence
- R27 – Explicit Conditions
- R28 – Multiple Conditions

Appendix C on Requirement Patterns also addresses the use of conditional clauses.

1.12 Organizing and Managing Needs and Requirements

1.12.1 The Problems of Complexity

Modern systems are becoming increasingly complex and software-intensive, which presents major challenges for an organization’s ability to effectively manage all the data, information, and artifacts across the lifecycle for these types of systems.

Historically, organizations defined and recorded needs and requirements in the form of “documents”. As systems become more complex and regulated, the sheer volume of documentation has become overwhelming, especially in terms of configuration management, change control, completeness, correctness, maintenance, and consistency. Due to this increasing complexity, there are more people involved in the development of these systems, often spread over different geographical locations. This may result in many of the documents being developed and managed within silos with limited collaboration.

Due to these issues, it is nearly impossible to keep all the needs and requirements, distributed across the various documents in sync with each other as well as in sync with other SE artifacts across the lifecycle. In addition, it is difficult to keep the needs and requirements within these documents current, correct, and consistent, leading to the situation in which there is no real authoritative source of truth (ASoT).

For highly regulated systems, the amount of documentation that must be developed, maintained, and supplied to regulators to show compliance has become overwhelming. Inconsistencies in these documents can result in a system that fails system validation and is not approved for its intended use by the regulatory agencies.

1.12.2 Managing Complexity

To address these issues, the trend is to move from a document-centric to a data-centric practice of SE. As such, some organizations are moving away from legacy documents associated with needs and requirements such as Stakeholder Needs Document, User Requirement Document, Program Requirements Document, System Requirements Document, Software Requirements Document, Functional Requirements Document, etc., as the source of ground truth. In a data-centric practice of SE, needs and requirements are defined, recorded, and managed electronically in sets within the project's SE toolset. Traditional documents such as those listed above would then, if necessary, be generated as reports from within the toolset.

As shown in Figure 7, for each SOI there is an Integrated Set of Needs, a set of Design Input Requirements, and a set of design output specifications (the "design") for the SOI. As discussed in the NRM, there is a set of lifecycle concepts, Integrated Set of Needs and set of Design Input Requirements for not only the integrated system, but also for each subsystem and system element within the integrated system architecture. These sets are managed within the SE toolset to allow both horizonal traceability across the lifecycle as well as vertical traceability between levels. This traceability is not just to ensure all the needs and requirements are captured in the design, but also is critical to help ensure the individual needs and requirements and sets of needs and requirements have many of the characteristics defined within this Guide.

As recommended within several of the rules (R29, R39, R41, and R42) defined in this Guide, within the sets it is a best practice for organizations to organize their needs and requirements within the sets by type or category to aid in the definition and management of the sets.

Refer to the NRM and GtNR for a more in-depth discussion concerning the data-centric practice of SE, traceability, organizing needs and requirements, and managing needs and requirements across the lifecycle.

1.13 Guide Organization

This guide focuses on the writing of need and requirement statements and addresses the following aspects of needs and requirements:

- Characteristics of individual need and requirement statements.
- Characteristics of sets of needs and requirements.
- Rules for individual need and requirement statements that help to formulate statements that have these characteristics.
- Rules for sets of needs and requirements that help to formulate sets that have these characteristics.

- Activities and concepts from the NRM that aid in the definition of need and requirement statements that have these characteristics.
- Attributes from the NRM that aid in the definition and management of need and requirement statements.
- Patterns that each well-formed need and requirement statement should follow.

This guide is organized as follows:

- Section 2 defines the characteristics of well-formed individual need and requirement statements, provides rationale for the characteristics, and provides guidance for helping understand the characteristics.
- Section 3 defines the characteristics of well-formed sets of needs and requirements, provides rationale for the characteristics, and provides guidance for helping understand the characteristics.
- Section 4 defines the rules for individual need and requirement statements and sets of needs and requirements that help to formulate need and requirement statements and sets of needs and requirements that have the characteristics defined in this Guide. Included with each rule is an explanation of the rule and examples of the application of the rule.
- Appendix A lists the sources and references used to write this document.
- Appendix B lists acronyms and abbreviations used in this document.
- Appendix C provides an overview of the concept of patterns and lists examples of patterns that can be used for different types of requirement statements.
- Appendix D contains a mapping table showing the applicability of each rule to need statements and requirement statements; it also classifies the rules in the guide into those requiring a data dictionary to be assesses and those not needed such a dictionary.
- Appendix E contains cross reference matrices mapping the rules to the characteristics discussed in this Guide as well as mapping concepts and activities discussed in the NRM to the characteristics, and mapping the attributes discussed in the NRM, Section 15 to the characteristics they aid in achieving.
- Appendix F includes a comment form.

Section 2: Characteristics of Individual Need and Requirement Statements

In defining needs and requirements, care should be exercised to ensure the need or requirement statement is well-formed. This section defines the characteristics of well-formed individual need and requirement statements, provides rationale for the characteristics, and provides guidance for helping to understand and achieve the characteristics.

In some cases, the characteristics can be achieved by following the rules, however in most cases the achievement of the characteristics is dependent on not only following the rules, but also doing the activities and analysis discussed in the NRM and GtNR and through the use of attributes defined in the NRM.

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.

2.1 Grouping Characteristics

The characteristics defined in this Guide are in harmony with those listed in the INCOSE SE Handbook, the SEBoK, and ISO/IEC/IEEE 29148 and only slightly different to those in ISO/IEC/IEEE 15288.

The authors acknowledge there are dependencies and apparent overlap among several of the characteristics (Carson 2018). Such issues are addressed within either the rationale or guidance provided for each characteristic.

Given the definition for need statements and requirement statements include the phrases “formal transformation” and “agreed-to obligation”, the characteristics for individual need and requirement statements discussed in this guide can be grouped as follows:

Formal Transformation. Given the need and requirement is a result of a formal transformation, the following characteristics of a well-formed need or requirement have been derived: • C1 - Necessary • C2 - Appropriate • C5 - Singular • C8 - Correct • C9 - Conforming	***Agreed-to Obligation***. Since the need and requirement is to be a part of a fair agreement to meet an obligation, the following characteristics of a need or requirement have been derived. • C3 - Unambiguous • C4 - Complete • C6 - Feasible • C7 - Verifiable/Validatable

The number of characteristics may seem overwhelming. To address this, slightly different, shorter groupings of characteristics have been proposed and are in use by some organizations.

2.1.1 SMART

Originally SMART was defined to apply to goals and objectives, and over time has been used to describe key characteristics of individual need and requirement statements. The original form was "specific, measurable, assignable, realistic, and time-related" (Doran 1981). The following is an adaptation of SMART that is more appropriate for defining well-formed needs and requirements.

- *Specific* - Singular (C5), concise, simple, clear, Complete (C4 individual), Consistent (C11 Set use of terms), Unambiguous (C3), Understood one way, Conforming (C9)
- *Measurable* - Singular (C5), concise, simple, clear, Complete (C4 individual), Consistent (C11 Set use of terms), Unambiguous (C3), understood one way, Conforming (C9)
- *Appropriate* (C2) - appropriate to level.
- *Realistic* - Feasible (C6 Individual) and (C12 - Set), Achievable within constraints (cost, schedule, technology, ethics, legal, regulatory, resources, risk).
- *Traceable* - Necessary (C1), Complete (C10 set), identifiable, linked, sufficient, Consistent (C11 with other related requirements).

2.1.2 C^{3}F

C^{3}F used frequently within the NRM when referring to well-formed sets of needs and requirements.

- *Correct (C15)* **-** Appropriate (C2), Unambiguous (C3), Complete (C4), Singular (C5), Verifiable/validatable (C7), Correct (C8 - individual), Conforming (C9), Comprehensible (C13), and Able to be Validated (C14).
- *Complete (C10)* **-** N*ecessary* (C1) - needed, traceable, sufficient.
- *Consistent (C11)* - use of terminology and with other related requirements.
- *Feasible (C12)* **-** Feasible (C6 Individual)*,* Achievable within constraints (cost, schedule, technology, ethics, legal, regulatory, resources, risk)

Correctness, completeness, and consistency are the three most common characteristics used to assess requirements quality by NLP/AI applications, while feasibility is more likely to be assessed by engineers.

2.1.3 VAN

VAN represents a condensed listing reflecting three main characteristics that needs and requirements must have.

- *Verifiable* - ***(C7) -*** Appropriate (C2), Unambiguous (C3), Complete (C4 individual), Singular (C5), Correct (C8 - Individual), Conforming (C9), Consistent (C11 Set), Comprehensible (C13 - Set), and Able to be Validated (C14 - Set), Correct (C15 - Set)**.**
- *Achievable* - Feasible (C6 Individual) and (C12 - Set) within constraints (cost, schedule, technology, ethics, legal, regulatory, resources, risk)**.**
- *Necessary* **(C1)** - needed, traceable, sufficient, Complete (C10 set)

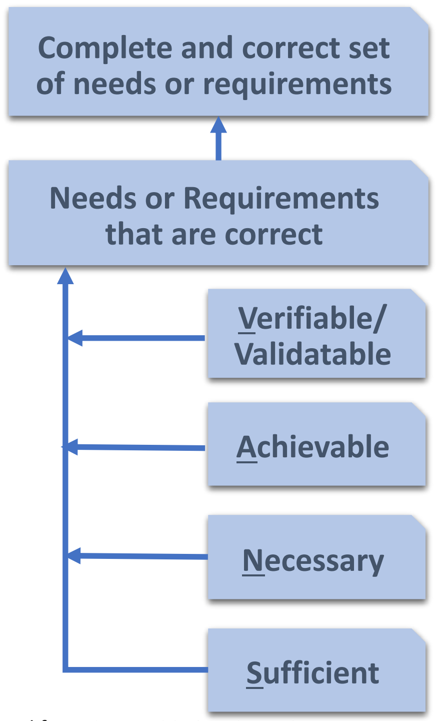
2.1.4 VANS

VANS is derived from (Carson 2018) representing four main characteristics, making a distinction between "necessary" and "sufficient". These four characteristics take into consideration the dependencies and overlap of the 15 characteristics discussed in this Guide.

- *Verifiable/Validatable (C7) - Appropriate (C2), Unambiguous (C3), Complete (C4), Singular (C5), Conforming (C9), Consistent (C11), Comprehensible (C13), and Able to be Validated (C14).*

- *Achievable - Feasible (C6 and C12) within constraints (cost, schedule, technology, ethics, legal, regulatory, resources, risk)*
- *Necessary (C1) - needed, traceable, Complete (C10) (set)*
- *Sufficient - Complete (C4 and C10).*

Carson proposes that these four characteristics result in need and requirement statements that are Correct (C8). Carson also proposes that, collectively, if each need and requirement statement is Correct (C8), then the Integrated Set of Needs and set of requirements in which it is a part will be both Complete (C10) and Correct (C15) as shown in Figure 8. It should be noted, however, that it is quite possible for all needs or requirements in a set to each be correct but for the set to not be complete.


Complete and correct set of needs or requirements
Needs or Requirements that are correct
Verifiable/ Validatable
Achievable
Necessary
Sufficient

Derived from Carson 2018 – Figure 2. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 8: Complete and Correct Sets of Needs and Requirements.

Even though there are dependencies and overlap between characteristics, it is instructive to address each of the individual 15 characteristics of needs and requirements elaborated in this Guide. Organizations can then choose how they will address each of these characteristics best suited to their culture, processes, domain, and product line.

2.2 C1 - Necessary

Definition:
The need statement or requirement statement defines capability, characteristic, constraint, or quality factor *needed or required* to satisfy a lifecycle concept, need, source, or higher-level requirement.

Rationale:

If the statement under review is not included in the set of needs or set of requirements, a deficiency in capability or characteristic will exist which cannot be fulfilled by implementing other needs or requirements in the set.

Realization of every need and requirement requires resources, effort, and cost in the form of development, review, management, implementation, design verification, design validation, system verification, system validation, maintenance, and disposal.

Unnecessary needs and requirements can lead to non-value-added work, additional cost, and unnecessary risk. Unnecessary needs or requirements inappropriately constrain the available solution space and cause extra program expense for developing, managing, implementing, verifying the unnecessary requirements, and validating the unnecessary needs. In the worst case, unnecessary needs or requirements may over-constrain and compromise the overall SOI performance, leading to infeasible solutions which fail to satisfy necessary needs and resulting requirements. Including unnecessary needs and requirements may also result in a set of needs or set of requirements that is not Feasible (C12).

A need or requirement is not necessary (not needed in the set of needs or set of requirements) if the:

- need or requirement can be removed, and the remaining set will still result in the entity's lifecycle concept or needs being satisfied;
- intent of the need or requirement will be met by the implementation of other needs or requirements;
- need or requirement cannot be traced back to a source, need, or higher-level requirement; or
- author cannot communicate a valid reason (rationale) for the need or requirement.

Guidance:

The formal transformation of a lifecycle concept into a need must result in a need addressing a specific aspect of the lifecycle concept that is necessary to meet the mission, goals, objectives, stakeholder expectations, regulatory requirements, drivers, constraints, and risks that are addressed in the lifecycle concepts.

Each need, individually or in combination with other needs in the set, must be sufficient to satisfy the intent of a specific aspect of the of the lifecycle concept or other source from which it was derived. "Sufficient" encompasses both the characteristics "Correct" (C8 and C15) and enhances it to ensure the need is not only an "accurate" representation of the specific aspect of the lifecycle concept or other source, but it is also sufficient to ensure the specific aspect of the lifecycle concept or other source that will be satisfied when the SOI realizes the needs.

The formal transformation of a need into a one or more requirements that individually or in combination with other requirements, must be both *necessary* and *sufficient* (Carson 2018) to satisfy a specific stakeholder need, source, or higher-level requirement from which it was transformed.

A requirement is considered sufficient if it, along with any siblings (common children of a single parent requirement), satisfies its parent requirement with acceptable margin (as determined by the program/project). For example, if the parent is a functional/performance requirement, compliance with the required functionality and performance level described by the set of sibling requirements should ensure conformance to the parent requirement by the integrated product.

"Sufficient" both encompasses the characteristic "Correct" (C8 and C15) and enhances it to ensure the requirement is not only an "accurate" representation of the entity need from which it was transformed but it is also sufficient to ensure the need is satisfied by the resulting requirement or child requirement when the SOI fulfills the requirements.

The development of child requirements via either decomposition, derivation, or elaboration must result in child requirements that are both necessary and sufficient to meet the intent of the need or source from which it is transformed or the allocated parent requirement in response to which the child requirement is being defined. The resulting child requirements can exist in different requirement sets for each subsystem or system element to which the parent requirement was allocated.

In many cases these child requirements have a dependency, where a change in one could require a change in one or more of the others. This is true when the child requirements are in response to a budgeted quantity (performance or quality characteristic) as well as interface requirements. It is important to link these dependent child requirements together (establish traceability between the dependent child requirements). Refer to the NRM Section 6 concerning allocation/budgeting and interface requirements.

Only necessary needs and the resulting requirements should be included in the need and requirement sets. Once each need or requirement is proven to be necessary, the set of needs must then be a sufficient solution for the agreed-to lifecycle concepts and the set of requirements must then be a sufficient solution to the set of needs. Together, the sets of needs and requirements form the design inputs against which the design outputs will be verified and validated. See characteristic (C10 - Complete) concerning completeness of sets of needs and requirements)

Caution should be taken to avoid gold plating, needs creep, and requirements creep. Refer to the NRM, Sections 4.6.3.4 and 6.2.6.3 for advice on avoiding gold plating and Section 14.2.51 on managing needs and requirements creep.

There is no such thing as a “self-derived” need or requirement. A need must always be able to be traced to a source such as a lifecycle concept, driver or constraint, system need, or mission statement, goal, objective, risk, trade study, or stakeholder expectation defined during lifecycle concept and needs definition activities discussed in the NRM and GtNR. Similarly, each requirement must be able to be traced to a need, higher-level requirement, or source.

While all requirements in the set should be demonstrated that they are necessary, that does not mean they are all equal. To address this, the NRM defines the attributes of Priority (A34), Criticality/Essentiality (A35), Risk (of implementation) (A36), and Key Driving Requirement (A38) to help manage the set of requirements over time, especially if the project must descope the requirements to address schedule and cost issues. The NRM provides a more detailed explanation of each of these and how they can be used.

One approach used by some organizations to limit the number of requirements to only those that are necessary and sufficient is to take a “zero based” or “minimum viable product” (MVP) approach. Such approaches start by only including needs and requirements that are high priority and are critical or essential for an MVP. Then additional requirements are only included if they add value in the context of the mission, goals, and objectives as well as drivers, constraints, risks, lifecycle concepts, and scenarios defined for the entity.

For market focused product development, many companies will aim to elicit as much information as possible from stakeholders to help define the product being built. A process of stakeholder needs and requirements analysis is used where they are considered based on their priority, criticality, prevalence, urgency, and the value realized by satisfying that need or requirement. Through the process of product management, a market can be defined consisting of stakeholders with similar product needs. At this point, a decision can be made on which needs and requirements to include as necessary to satisfy the market needs.

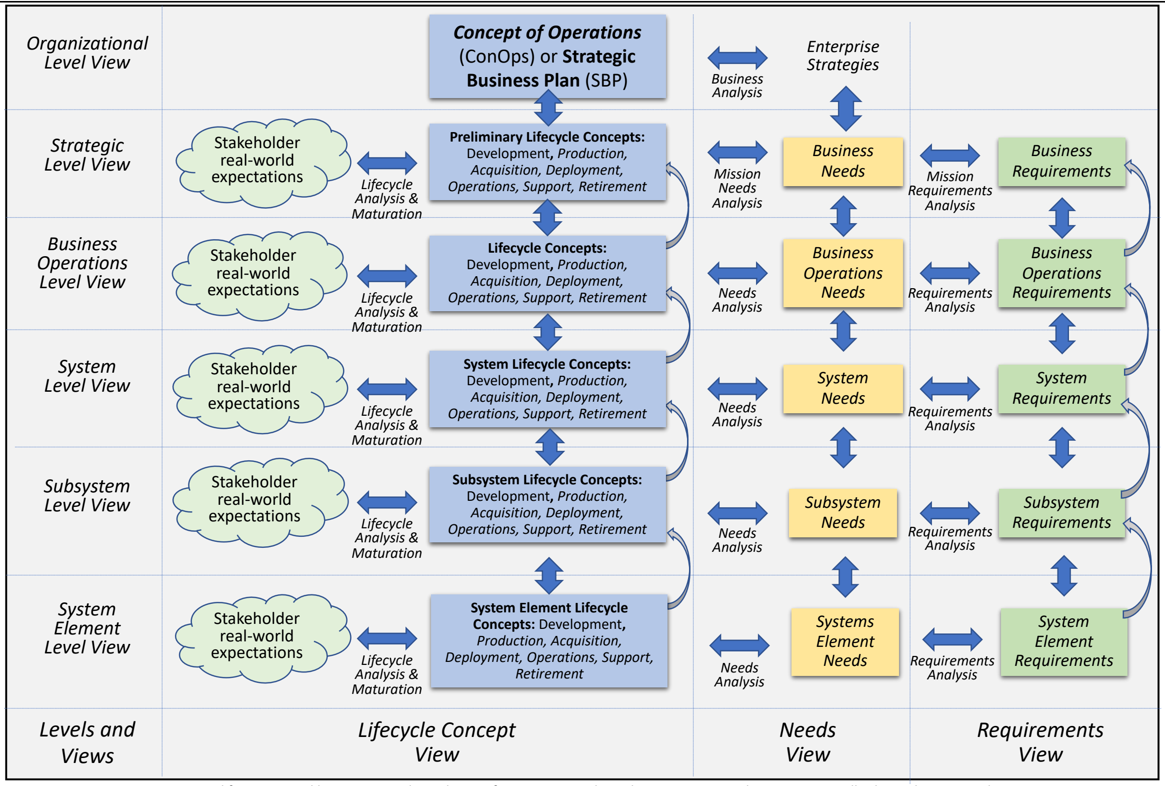
In some cases, organizations will include goals within a need or requirement set that are not mandatory nor binding but do add value for some stakeholders. In the Integrated set of needs there would be a statement such as "The <stakeholders> would like the <SOI> to ……. If the stakeholders agree to fund this goal, then it would be turned in to a binding need statement "The <stakeholders> need the <SOI name> to ……>. In the set of Design Input Requirements goals are sometimes included in the set expressed as "should" statements that are not binding but highly desirable. In some cases, a requirement statement will be written as "The <SOI> shall <perform some function> with a <performance of "x" and a goal of "y">. Where "y" is a better performance than x. In this case "x" is a minimum threshold, with a desire to achieve "y" *if feasible*. The inclusion and management of goals is beyond the scope of this Guide.

To summarize, if the need or requirement cannot be traced to a source or rationale cannot be articulated, it is not necessary. The inclusion of rationale and other attributes defined in the NRM, Section 15, such as *trace to source or* higher-level, for each need and requirement also aids in communicating the necessity and intent of the need or requirement.

2.3 C2 - Appropriate

Definition:
The specific intent and amount of detail of the need or requirement statement is appropriate to the level (the level of abstraction, organization, or system architecture) of the entity to which it refers.

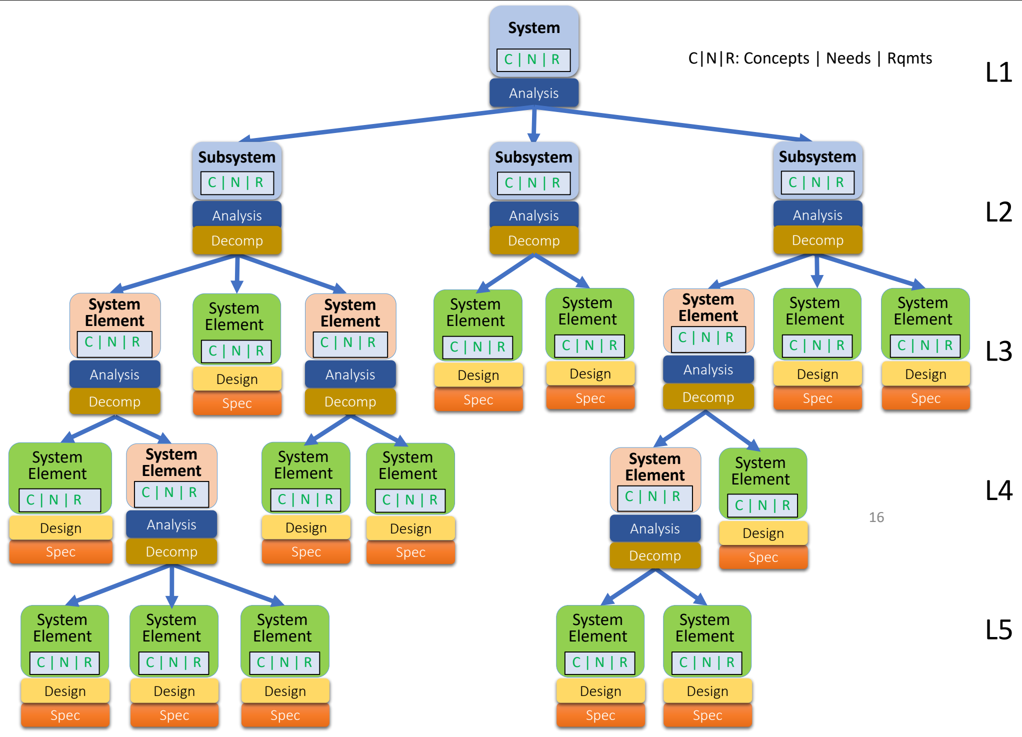
Rationale:
The wording of needs is often stated at a higher level of abstraction than is appropriate for a requirement. *See the example under R31*. As shown in Figure 9, levels can also refer to levels within an organization or levels within an architecture—system, subsystems, or system elements. Needs and requirements may be defined at any level of the organization or system architecture; however, as a rule, a need or requirement should be expressed at the level of the entity to which it refers, and that entity must be able to be validated or verified to have met the need or requirement at that level. Requirements for lower-level entities stated within the requirements set for a higher-level entity may seem like implementation. When design details are stated as design inputs, there is often no definition of the "why" and what the parent requirement is that the design detail requirement is addressing. In this case, the real higher-level entity requirement may not be stated and thus not properly allocated to the next lower-level entities. When this is the case, the requirement should be moved down to the appropriate level entity set of requirements and the missing parent requirement added to the higher-level entity's set of requirements. Conversely, higher level entity requirements stated improperly within a lower-level entity's set of requirements may not have been allocated properly to the other entities at the next level of the architecture, resulting in missing child requirements for those entities. A need or requirement stated at the wrong level for an entity is either not Correct (C8) or may not be verifiable/ validatable (C7) at that level. Refer to the NRM for an in-depth discussion on levels.


Organizational Level View
Concept of Operations (ConOps) or Strategic Business Plan (SBP)
Business Analysis
Enterprise Strategies
Strategic Level View
Stakeholder real-world expectations
Lifecycle Analysis & Maturation
Preliminary Lifecycle Concepts: Development, Production, Acquisition, Deployment, Operations, Support, Retirement
Mission Needs Analysis
Business Needs
Mission Requirements Analysis
Business Requirements
Business Operations Level View
Lifecycle Concepts: Development, Production, Acquisition, Deployment, Operations, Support, Retirement
Needs Analysis
Business Operations Needs
Requirements Analysis
Business Operations Requirements
System Level View
System Lifecycle Concepts: Development, Production, Acquisition, Deployment, Operations, Support, Retirement
System Needs
System Requirements
Subsystem Level View
Subsystem Lifecycle Concepts: Development, Production, Acquisition, Deployment, Operations, Support, Retirement
Subsystem Needs
Subsystem Requirements
System Element Level View
System Element Lifecycle Concepts: Development, Production, Acquisition, Deployment, Operations, Support, Retirement
Systems Element Needs
System Element Requirements
Levels and Views
Lifecycle Concept View
Needs View
Requirements View

Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 9: Levels of Lifecycle Concepts, Needs, and Requirements

Guidance:



Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 10: Levels of a System – Hierarchical View

Unless there is a good reason, a subject of the need or requirement statement should refer to the entity at the level at which the statement is being expressed (not higher or lower). Referring to Figure 10, needs and requirements for an architectural entity should be expressed within each architectural entity's set of needs or requirements.

A key consideration when writing and reviewing need and requirement statements is whether the subject (entity) of the statement is the subject (entity) of the set of statements at that level.

Refer to Section 1.7 for a discussion concerning needs and requirements and the entity to which they apply. Refer to Section 1.8 concerning the differences between needs and requirements.

Design input, design-to "what" requirements must not be any more detailed or specific than is appropriate for the level of the entity at which they are stated, not "how" the requirement should be met via design. However, design output, build-to/code-to "how" requirements purpose is to communicate to the builders/coders the agreed-to design, so they will reflect implementation.

To avoid confusion, many organizations refer to design output requirements as "specifications" that often include parts lists, drawings, wiring diagrams, plumbing diagrams, labeling diagrams and requirements, logic diagrams, algorithms, Computer-aided Design (CAD) files, or STL files (for 3D printing). *For the purposes of this Guide, the focus is on design input needs and requirements as shown in Figure 2*.

The Design Input Requirements avoid placing unnecessary constraints on the design at the given level. The goal for Design Input Requirements is to be implementation independent. There may be cases where there is good rationale for stating implementation. When this happens, the rationale must be included to make it clear why the specific implementation needs to be stated as a constraint to the design.

A useful question to ask of a requirement is "for what purpose?" or "why?" If the requirement is expressed in implementation terms (design output), the answer to this question may be the real requirement (design input).

In the first example (how), verification could be by "inspection" counting the number of engines. In the second example (what), by asking the question "Why?", the real requirement is stated, and verification would be done by "test", disabling one engine during flight and ensuring the operational requirements for the aircraft can still be met. (Note: missing from this example are the operational conditions to which this requirement applies.)

Another useful question is to ask of a requirement is "what do you want to verify?" For example:

> The <aircraft> shall include Flight_Performance Instrumentation." as opposed to:
>
> The <aircraft> shall measure <xxxx Flight_Performance parameter> having the characteristics defined in <Flight_Performance Data Dictionary yyyy> during operations.

In the first example, there could be instrumentation (how), but it may not obtain the needed flight performance parameters. Verification could be by "inspection" indicating instrumentation can be observed. In the second example (what), verification would be done by "test" observing each flight performance parameter has the characteristics defined in the Flight_Performance Data Dictionary.

It is good practice, once the next level entity requirements are written, for the project team to do a "leveling" exercise, to look at each requirement for the entity and determine whether it is at the appropriate level or should be moved down to a lower-level entity's set of requirements or moved up to a higher-level entity's set of requirements.

As noted earlier, an indication when writing and reviewing statements is whether the subject of the statement is the subject of the set of statements for that entity at that level.

2.4 C3 - Unambiguous

Definition:

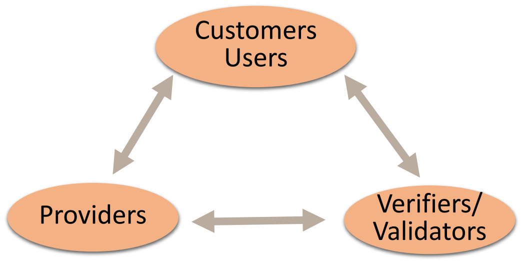
Need statements and requirement statements must be stated such that their intent is clear and can be interpreted in only one way by all intended audiences.

Rationale:

A need statement and a requirement statement must lend itself to a single interpretation of intent.

These statements are effectively an agreement between two parties communicating intent from one to the other and forming the basis of verification and validation that the intent was met. An agreement is difficult to enact unless both parties are clear on the exact obligation.

As shown in Figure 11, the intent of a need or requirement must be understood, in the same way, by all intended audiences across the lifecycle.



Original figure created by L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 11: Needs and Requirements Understood in the Same Way

Ambiguity leads to interpretations of a need or requirement not intended by the author leading to problems such schedule slips, budget overruns, or a failure of the SOI to pass system validation and not be accepted for its intended use; which could result in litigation and financial loss.

An ambiguous need or requirement is neither Correct (C8) nor Verifiable/Validatable (C7).

Guidance:

When writing a need or requirement statement, ask whether it could be interpreted more than one way. It is common for people to interpret words based on their own set of filters, biases, and assumptions. For example, if a person states: "I need a cream <u>for</u> 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"?

Often, people will use common words by themselves without using a modifier that helps the receiver understand the context, and thus the intended meaning of the word used. It is a bad practice to assume that the meaning of the words used are understood the same way by all intended audiences.

(Note: Refer to the NRM, Section 2.10, on a more detailed discussion on effective communications.)

For needs, ask whether the need is Validatable (C7), that is, whether it is stated in such a way that sufficient evidence can be obtained that the need has been met based on the wording of the need statement without having to interpret the stakeholder intent or make assumptions of that intent.

For a requirement ask whether the requirement is Verifiable/Validatable (C7), that is, whether it is stated in such a way that sufficient evidence can be obtained that the requirement has been met based on the wording of the requirement without having to interpret the intended meaning or make assumptions as to the intended meaning.

It is useful for the stakeholders who are involved in the implementation of the needs and resulting requirements or system verification and system validation to be involved in the definition, review, and baseline of the needs and resulting requirements. When they see needs or requirements that are ambiguous and the stakeholder's intent is not clearly communicated, they can identify the issue and suggest an alternate, unambiguous wording of the need or requirement statement.

As a minimum, it is recommended that the need or requirement owner(s) along with the development team and those involved in system verification and system validation do a walkthrough of the need set or requirement set to ensure that needs and requirements are understood as intended, individually and as a set. As discussed in Section 1.9, this activity is referred to as needs validation or requirement validation.

Because a need expression or requirement expression includes both the statement and a set of attributes, attributes can be used to help address any ambiguities that may still exist even in the most carefully worded need statement or requirement statement.

Due to the limitations of language, it may prove difficult to completely remove all ambiguity. In this case, the use of the Attribute, A1- Rationale, can be used to include contextual information to better understand the reason and source of the requirement and may provide additional insight of the intent, helping to reduce ambiguity and provide some increased ability to apply the "reasonable person" test. Supporting information or commentary on how and why the requirement was formed can be included in the rationale. By including rationale in each need and requirement expression, readers do not have to assume this information—it is clearly stated for all to see and understand in the same way.

In addition, Attribute, A39 - Additional Comments, may be used to add additional information beyond what was included in the rationale. Additional comments can be used to document possible issues with the need or design input requirement, any conflicts, status of negotiations, actions, design notes, implementation notes, etc. Further, evaluation and prototyping of the system concept may have identified important guidelines for the implementation of the need or requirement. This information may be useful as the need or requirement is being reviewed and serves as a place to document information not addressed by other attributes.

Note that the information contained within the attributes is meant to be informative in a contractual sense (not normative), because of this, the word 'shall' must not be used in any of the text contained in attributes to help ensure the amplifying information is not inadvertently interpreted to be an additional requirement.

When text only makes it difficult to communicate the intent of complex requirement, the inclusion of a diagram or trace to a model may help remove the ambiguity. See R23.

Ambiguity of individual need and requirement statements can be assessed during early system verification and design verification activities discussed in the NRM and GtVV.

2.5 C4 - Complete

Definition:
The need statement sufficiently describes the necessary capability, characteristic, constraint, conditions, or quality factor to meet the lifecycle concept or source from which it was transformed. The requirement statement sufficiently describes the necessary capability, characteristic, constraint, conditions, or quality factor to meet the need, source, or higher-level requirement from which it was transformed.

Rationale:
An agreement is not useful unless the obligation is complete and does not need further explanation. A well-formed need statement needs no further amplification to implement its intent and to define system validation success criteria, method, and approach. A well-formed requirement statement needs no further amplification to implement its intent and to define system verification success criteria, method, and approach. Examples include: • Functional requirements must include a performance characteristic to be complete and verifiable. This can result in multiple requirements for a function, each with a different performance characteristic. • Interface requirements should include a reference to the location of the agreement that defines how the entity needs to interact with the entity to which it interfaces (for example, an Interface Control Document (ICD). • Terms used within a need statement or requirement statement must be defined uniquely and unambiguously within the project glossary or Data Dictionary. • Requirements based on a standard or regulation must clearly communicate what the system must do to meet the intent of the standard or regulation requirement from which it was derived. An incomplete need statement or requirement statement is not Verifiable/Validatable (C7) nor Correct (C8) due to missing information (the need or requirement fails to address either “what”, “how well”, or “under what conditions”).

Guidance:
To be complete, functional/performance requirements must have observable functions (“what”), measurable performance (“how well”) and a statement of conditions (“under what conditions” - for example, triggering events, environments, states and modes). Such requirements may be insufficient to satisfy the need or higher-level requirement because of the omissions of these types of information that are needed to make clear the intent. While fully appreciating that a need or requirement may require some context, the need statement or requirement statement itself should be a complete sentence that does not require reference to other need or requirement statements to be understood in its basic form. In another aspect of completeness, requirements should not refer to one another, nor should the understanding of the requirement assume the existence of a previous or subsequent requirement, nor the fact that the need or requirement is contained within a specific section of a document (that is, despite the content of the heading of the section containing a need or requirement, a complete statement shall include or repeat the name of the SOI, even if it is already mentioned in the

heading—see R25). This is especially important when requirements are managed within an application or database.

A need or requirement can refer to other documents that elaborate on the “how well” or “under what conditions” (for example an ICD, or data dictionary, or another external source).

Referring to these types of documents within a requirement statement is warranted in that when dealing with interactions between systems, it is critical that all parties involved are consistent and refer to the same, agreed to definition on how the systems will interact across an interface boundary. It would be dangerous if each party included their interpretation of the interaction within their requirements, which may be inconsistent to another party’s interpretation as it appears in their interface requirements. If they each include their own interpretation, they may not be able to interact as needed.

In the case of data dictionaries, the same logic applies. When parameters, messages, and commands have a role in an interaction, it is critical that all parties involved in the interactions understand the agreed-to characteristics of each entity involved in the interaction.

When making referrals to other documents, requirements must refer specifically to the sections within the documents that apply.

Incomplete needs and requirements can be prevented through the use of patterns for the need or requirement statement. By following a specific pattern or template for a requirement statement, requirements tend to be “more complete”, since these patterns provide guidance for specific information the requirement statement should include. Even if there is a pattern defined for ubiquitous needs and requirements, most requirements will not be ubiquitous. Refer to R1 on the use of a pattern or template and Appendix C for more detailed discussion on patterns and templates.

Except under exceptional circumstances, baselined requirement statements must not contain To Be Defined (TBD), To Be Specified (TBS), or To Be Resolved (TBR) clauses. TBx should only be used during the analysis and definition process to indicate ongoing work but should not be in the final requirement set, particularly when that set is to be contained as a baseline in a contractual arrangement. Resolution of the TBx designation may be iterative, in which case there should be an acceptable period for the TBx item to be resolved, determined by risks and dependencies. If the baselined set of requirement contains requirements with a TBx item, it must be made clear in supplier or vendor agreements (for example, SOWs) who is responsible to resolve these items and when. Refer to the NRM Section 14.2.4 for a detailed discussion concerning TBx management and managing unknowns. It should be noted that this requires NLP/AI tools to be flexible and tailorable; that is, a rule that must be reinforced at other later stages must be able to be ignored in the early stages.

Completeness of individual need and requirement statements can be assessed during early system verification and design verification activities discussed in the NRM and GtVV.

2.6 C5 - Singular

Definition:
The need statement or requirement statement should state a single capability, characteristic, constraint, or quality factor.

Rationale:
The formal transformation from a lifecycle concept into a need can be a many-to-one, one-to-one or a one-to-many transformation, however, the resultant need statement(s) must each represent a single thought, aspect, or expectation. Similarly, the formal transformation of a need, source, or allocated parent requirement into a requirement can be a many-to-one, one-to-one, or a one-to-many transformation so the resultant requirement statement(s) must each represent a single thought, aspect, or expectation. The effectiveness of several process activities associated with needs and requirements definition—such as decomposition, derivation, allocation, traceability, verification, and validation—depend on being able to identify singular statements. For instance, the system verification information defined for a requirement can be far more precise when that requirement addresses a single capability, characteristic, constraint, or quality factor. Additionally, when verifying/validating the SOI against a need or requirement with multiple aspects, if one aspect passes, but others fail, it is difficult to assess the verification/validation status of the SOI against that need or requirement as a whole. A need or requirement with multiple thoughts is difficult to allocate and to trace to a higher-level requirement or source. A nonsingular need or requirement is neither Verifiable/Validatable (C7) nor Correct (C8). Requirements patterns are useful ways to ensure singularity since each need or requirement should conform with one and only one pattern (see Appendix C).

Guidance:
Keep the need statement or requirement statement limited to one quality, characteristic, capability, function, or constraint that is appropriate to the level at which the statement refers. Understand how the statements fit into the allocation, traceability, verification, and validation philosophy for the project. Use the project standard patterns or templates for writing singular need and requirement statements. Although a single need or requirement should consist of a single function, quality or constraint, it may have multiple conditions under which the requirement is to be met. Avoid the use of the word "and", R19, when it ties together multiple thoughts (phrases in the sentence), each of which may be allocated, and the SOI verification and validation success criteria, method and approach are different for each. The presence of the conjunction "and" in a need or requirement statement (out of the condition block, where the AND should be understood as logical connector) should always prompt the writer to consider whether or not the statement is singular – if not, then multiple needs or requirements should be defined, each addressing a singular thought. There are exceptions to the use of "and". Some organizations allow the use of "and" to join two actions that will always be allocated, traced, and verified or validated together. For example, there may be a requirement for a lit match to be extinguished and disposed into a container. Given that these two actions need to always be performed together (one action is never to be completed without the other), organizations may allow both actions to be communicated within a single requirement. In that case, however, it would be best to combine the two actions with a logical "AND"—see R15.

2.7 C6 - Feasible

Definition:

The need or requirement can be realized within entity constraints (for example: cost, schedule, technical, legal, ethical, safety) with acceptable risk.

Rationale:

There is little point in agreeing to an obligation for a need or requirement that is not feasible. Agreeing to a need or requirement that cannot be realized with acceptable risk within constraints often results in project cost overruns and schedule slips. Inherently unachievable needs and requirements, such as 100% reliability, are at best a waste of time, and at worst lead to needlessly expensive solutions.

The need or requirement is considered feasible if, when considered along with other needs or requirements for a single system element (that is, a set of entity needs or requirements), it does not cause an unacceptable cost, schedule, or risk impact during the entity's lifecycle.

An infeasible need or requirement cannot be satisfied because it (Carson 2018):

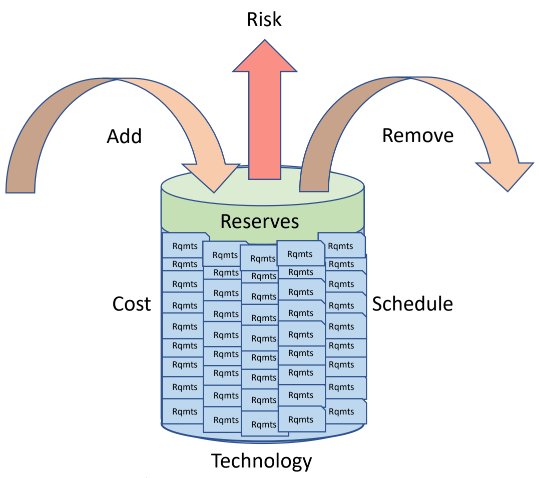
a. breaks the laws of physics,
b. violates laws or regulations in an applicable jurisdiction,
c. conflicts with another requirement and cannot be concurrently satisfied, or
d. leads to excessive program risk because of technical immaturity or inadequate margin with respect to program cost and schedule as a function of lifecycle phase.

An infeasible need or requirement is neither verifiable (C7) nor correct (C8).

Guidance:

Feasible implies the existence of a possible solution to satisfying a concept, need, or requirement.

As stated in the definitions for needs and requirements, requirements are transformed from needs. With this in mind, if the organization has not defined a feasible set of lifecycle concepts and assessed the maturity of the critical technologies, then the needs derived from those concepts may not be feasible, nor will the requirements transformed from those needs be feasible.



Original figure created by L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 12: Needs or Requirements Feasibility Bucket

Therefore, in most cases determining whether the need or requirement is feasible is difficult without an underlying assessment and analysis of the proposed lifecycle concepts from which the needs are derived, and requirements transformed.

As shown in Figure 12, before allowing a need into the set, an assessment of the feasibility of the chosen lifecycle concepts must be made in terms of the drivers and constraints including the maturity of critical technologies, cost, schedule, resources, regulations, higher level requirements, and risk.
If not feasible within the stated constraints with acceptable risk, the need and resulting requirement should not be included in the set. Doing so can negatively impact cost and schedule and can result in a requirement that the entity cannot be verified to meet. The measurement of feasibility is not always easy to assess as it is based on the degree of risk in successfully implementing the requirement within program constraints. See the NRM for a more detailed discussion of the feasibility bucket shown in Figure 12.

A useful tool for assessing risk is the use of Technology Readiness Levels (TRLs) to determine and compute the maturity of an essential technology—a lower TRL represents more risk to the project than a higher TRL. As discussed in the NRM, as part of risk assessment and during lifecycle concept analysis and maturation, essential technologies are identified, their TRL and Advancement Degree of Difficulty (AD^2) assessed, and a technology maturation plan developed that will result in the essential technology maturity to advance such that it will be feasible in time consistent with the system development schedule. Low technical maturity technologies needed for required performance represent a risk to the project and as such this risk must be managed closely.

Other useful tools include modeling and prototyping to evaluate the feasibility of individual needs and requirements and subsets of needs and requirements. This means that the design team must be included in an integrated, multidiscipline, collaborative team responsible for the lifecycle concept and needs definition activities to assess the feasibility of a lifecycle concept in terms of physical implementation of the functional or conceptual model as discussed in the NRM and GtNR.

In some cases, we can recognize and avoid needs and requirements that are clearly impossible or unrealistic (see R26). More often, feasibility must be examined for sets of needs or requirements associated with a single entity to ensure there is no conflict within the set of requirements (C12).

Feasibility of individual need and requirement statements can also be assessed during early system verification and design verification activities discussed in the NRM and the GtVV.

2.8 C7 – Verifiable/Validatable

Definition:
The need statement is structured and worded such that its realization can be validated to the approving authority's satisfaction. The requirement statement is structured and worded such that its realization can be verified to the approving authority's satisfaction.

Rationale:

Unless a *need statement* is written in a way that allows requirement validation, design validation, and system validation, there is no way to tell whether it has been satisfied and that the expectation has been met.

- Each need statement must include the necessary information such that validation success criteria can be defined, and the SOI can be validated such that sufficient evidence can be gathered to determine whether the success criteria have been met, that is, there is no ambiguity regarding what the need statement communicates and there are no missing characteristics within the need statement, that is, it has the characteristics for well-formed need statements as defined in this Section.
- An unvalidatable need can result in multiple, objective observers (for example, requirement writers, architects, designers, or testers) interpreting the need differently making it difficult to validate that the requirement, design, and SOI meets the need.

Unless a *requirement statement* is written in a way that allows design verification and system verification, there is no way to tell if it has been satisfied and that the obligation has been met.

- Each requirement statement must include the necessary information such that verification success criteria can be defined, and the SOI can be verified such that sufficient evidence can be gathered to assess whether the success criteria has been met, that is, there is no ambiguity regarding what the requirement statement communicates and there are no missing characteristics within the requirement statement, that is, it has the characteristics for well-formed requirement statements as defined in this Section.
- An unverifiable requirement can result in multiple, objective observers (for example, designers or testers) interpreting the requirement differently, making it difficult to verify the design and SOI meets the requirement.

Verifiability and validatability is a necessary condition for establishing the other characteristics of need and requirement statements defined in this Section. Therefore, verifiability and validatability should be addressed as the initial criterion and a basis for ensuring these other characteristics.

Guidance:

It is important not to confuse the phrases “need verification”, “need validation”, “requirement verification”, and “requirement validation” as opposed to “design verification”, “design validation” “system verification”, or “system validation” as shown in Figure 7 and discussed in Section 1.9.

This characteristic, verifiable/validatable, is about design and system verification and requirement, design, and system validation—showing that a requirement can be validated against the need from which it was transformed, the design can be verified/validated such that, when realized, will result in a SOI that meets a requirement/need, and verifying/validating that the realized SOI meets a requirement/need.

Write each requirement/need statement in a way that allows the design or system to be verified/validated that the requirement/need has been met by one of the four standard verification/validation methods (inspection, analysis, demonstration, or test). Essentially, this means ensuring that each need and requirement statement is well-formed having all the other characteristics defined in this Section.

2.9 C8 - Correct

Definition:

The need statement must be an accurate representation of the lifecycle concept or source from which it was transformed.

The requirement statement must be an accurate representation of the need, source, or higher-level requirement from which it was transformed.

Rationale:

Correct implies "no errors" both from the perspective of the inclusion of incorrect information, the omission of required information, and avoidance of ambiguous wording. These aspects are similar to those in other disciplines such as NLP with the classical human filters of generalization, deletion, and distortion.

Incorrect information can mean having the wrong:
- values,
- functions,
- conditions, or
- other characteristics identified in the need or requirement.

An incorrect need can result in a need that does not reflect the intent of the lifecycle concept or sources from which it was transformed. An incorrect need can result in incorrect requirements transformed from that need. An incorrect requirement can result in a requirement that does not reflect the intent of the need, source, or higher-level requirement from which it was transformed.

The need or requirement cannot be correct if it does not have the characteristics: Necessary (C1), Unambiguous (C3), Complete (C4), Feasible (C6), Verifiable/Validatable (C7), and Conforming (C9).

Guidance:

Needs are transformed from a source (such as lifecycle concepts, stakeholder expectations, drivers and constraints, goals, objectives, and risks) defined as part of the lifecycle concept and needs definition activities discussed in the NRM and GtNR.

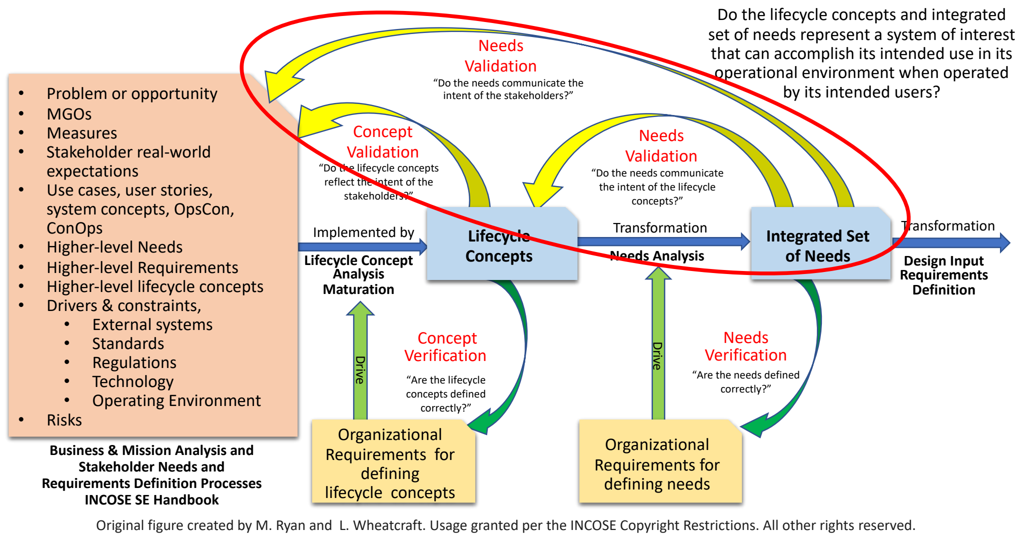

Problem or opportunity
MGOs
Measures
Stakeholder real-world expectations
Use cases, user stories, system concepts, OpsCon, ConOps
Higher-level Needs
Higher-level Requirements
Higher-level lifecycle concepts
Drivers & constraints,
External systems
Standards
Regulations
Technology
Operating Environment
Risks
Business & Mission Analysis and Stakeholder Needs and Requirements Definition Processes INCOSE SE Handbook
Needs Validation
"Do the needs communicate the intent of the stakeholders?"
Do the lifecycle concepts and integrated set of needs represent a system of interest that can accomplish its intended use in its operational environment when operated by its intended users?
Concept Validation
"Do the lifecycle concepts reflect the intent of the stakeholders?"
Needs Validation
"Do the needs communicate the intent of the lifecycle concepts?"
Implemented by
Lifecycle Concept Analysis Maturation
Lifecycle Concepts
Transformation
Needs Analysis
Integrated Set of Needs
Transformation
Design Input Requirements Definition
Drive
Concept Verification
"Are the lifecycle concepts defined correctly?"
Drive
Needs Verification
"Are the needs defined correctly?"
Organizational Requirements for defining lifecycle concepts
Organizational Requirements for defining needs
Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

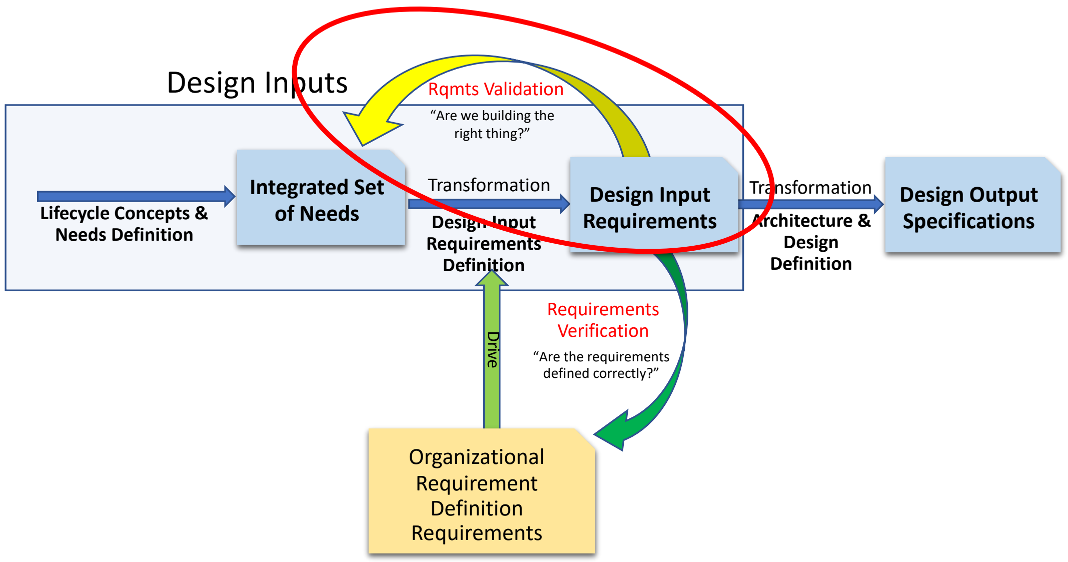
Figure 14: Needs Validation

As shown in Figure 14, the resulting needs transformed from these sources must be validated to ensure that the need statement communicates the right thing (intent) and reflects an accurate and unambiguous:

- interpretation of the concept, stakeholder expectation, drivers and constraints, goals, objectives, risks, and other sources from which it was transformed;
- understanding of the problem or opportunity and underlying goals and objectives;
- representation of any model or diagram from which the need was extracted so the need traces to the model; and
- representation of any underlying analysis and assumptions that were part of the transformation.

Use a defined development and management process to ensure accuracy of the transformation in the context of the individual need as well as the other needs in the set.

Requirements are transformed from a need, source, or higher-level requirement. The resulting requirements must be validated to ensure that the requirement statement communicates the right thing and that achievement of the requirement, as written, will result in meeting the intent of the need, source, or higher-level requirement from which it was transformed.


Design Inputs
Rqmts Validation
"Are we building the right thing?"
Lifecycle Concepts & Needs Definition
Integrated Set of Needs
Transformation
Design Input Requirements Definition
Design Input Requirements
Transformation
Architecture & Design Definition
Design Output Specifications
Requirements Verification
"Are the requirements defined correctly?"
Drive
Organizational Requirement Definition Requirements

Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 15: Requirements Validation

As shown in Figure 15, validate that the requirement statement reflects an accurate and unambiguous:

- interpretation of the need, source, or higher-level requirement from which it was transformed;
- representation of the model or diagram from which the requirement was extracted so the requirement traces to the model or diagram; and
- representation of the underlying analysis and assumptions that were part of the transformation.

As discussed in Section 1.9, perform the needs and requirements validation described in the NRM and GtNR to ensure correctness of the transformation in the context of the individual need and requirement statements as well as the complete sets of needs and requirements.

Correctness of individual need and requirement statements can also be assessed during early system verification and design verification activities discussed in the NRM and GtVV.

2.10 C9 - Conforming

Definition:

Statements and expressions of individual needs and requirements should conform to an approved standard pattern and style guide or standard for writing and managing needs and requirements.

Rationale:

When needs and requirements within the same organization have the same look and feel, each need statement and requirement statement is easier to be written, understood, and reviewed.

Further, when conforming to an approved standard, the quality of the individual need statement and requirement statement will improve, as will the quality of the need set, and the requirement set.

For the derivation of a need from a lifecycle concept to be formal, the structure of the resultant need statement must also be formal.

- For example, all needs may be required to be structured according to a specific pattern defined by the organization for the type of need statement: "The <stakeholders> need the <subject clause> to <action verb clause> <object clause>, <optional qualifying clause>."

For the transformation from a need statement to a requirement statement to be formal, the structure of the resultant requirement statement must also be formal.

- For example, all requirement statements may be required to be structured according to a specific pattern defined by the organization for the type of requirement statement: "When <condition clause>, the <subject clause> shall <action verb clause> <object clause>, <optional qualifying clause>."

Conforming to standards and templates for need statements and requirement statements:

- provides a template for wording associated with the different types of needs and requirements;
- provides consistent wording prevents ambiguity for engineers being exposed to new types of needs and requirements they are managing; and
- will help to identify missing information resulting in the characteristics of Complete (C10), Unambiguous (C3) and Verifiable/Validatable (C7).

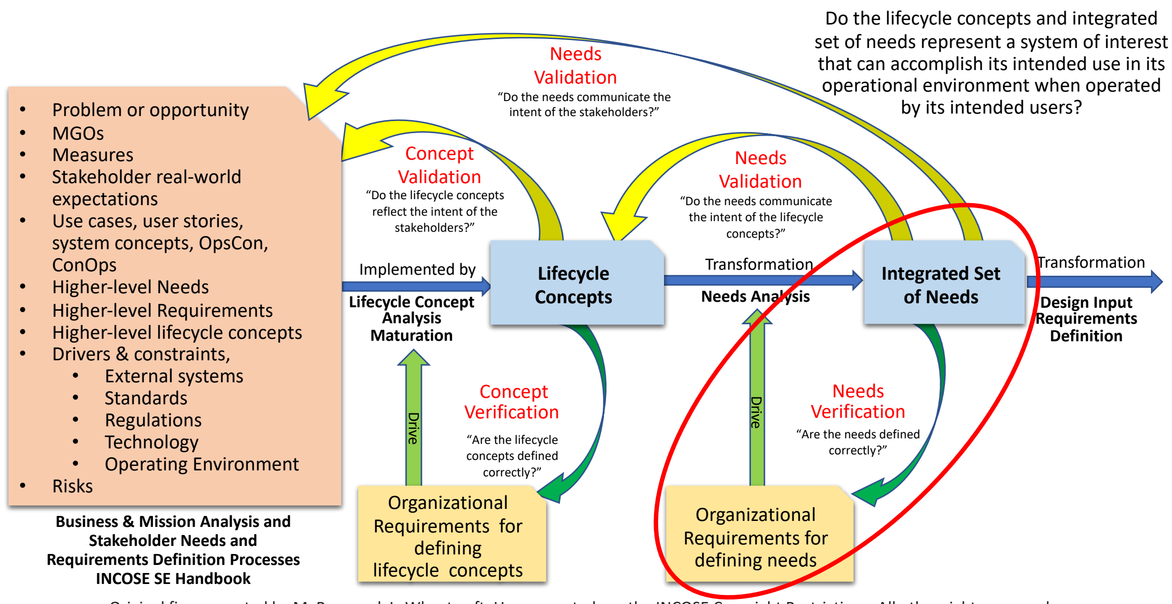
See R1 for more detail on need statements and requirement statements formats and Appendix C for more detailed information on the use of templates and patterns.

Guidance:

Organizations must have well-defined standards, processes, and methods for needs and requirements definition and management. These standards and processes must include rules for defining well-formed need statements and requirement statements, checklists to assess the quality of the need statements and requirement statements, and templates for allowable structures for need statements and requirement statements. For example, all need statements, and requirement statements may be required to have the characteristics and be written in accordance with the rules contained in this Guide. 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.

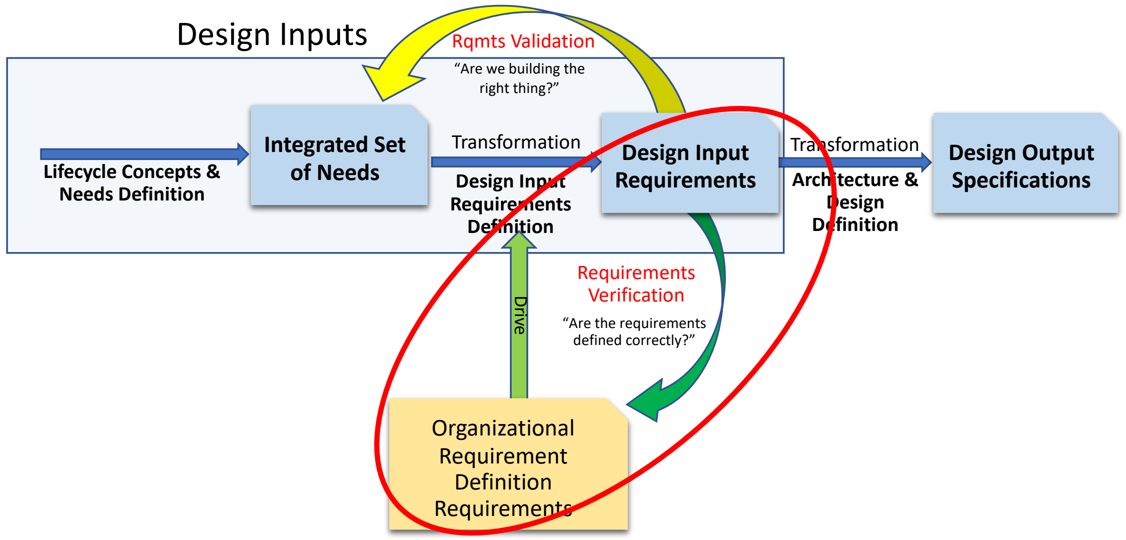
In many instances, there are applicable government and industry standards with which compliance is required. When developing needs and requirements for a customer, the customer may have their own process, standards, checklists, and patterns. The organization responsible for defining the need statements and requirement statements may need to conform to the customer's processes and standards.

Once a set of needs or a set of requirements has been defined, the organization must have an approval, baselining, and configuration management process defined. As shown in Figures 16 and 17, this process must include verification of the needs and requirements quality per the organization's defined standards. It is this verification that will help establish this characteristic.


Do the lifecycle concepts and integrated set of needs represent a system of interest that can accomplish its intended use in its operational environment when operated by its intended users?
Needs Validation
"Do the needs communicate the intent of the stakeholders?"
Problem or opportunity
MGOs
Measures
Stakeholder real-world expectations
Use cases, user stories, system concepts, OpsCon, ConOps
Higher-level Needs
Higher-level Requirements
Higher-level lifecycle concepts
Drivers & constraints,
External systems
Standards
Regulations
Technology
Operating Environment
Risks
Business & Mission Analysis and Stakeholder Needs and Requirements Definition Processes INCOSE SE Handbook
Concept Validation
"Do the lifecycle concepts reflect the intent of the stakeholders?"
Implemented by
Lifecycle Concept Analysis Maturation
Lifecycle Concepts
Needs Validation
"Do the needs communicate the intent of the lifecycle concepts?"
Transformation
Needs Analysis
Integrated Set of Needs
Transformation
Design Input Requirements Definition
Drive
Concept Verification
"Are the lifecycle concepts defined correctly?"
Drive
Needs Verification
"Are the needs defined correctly?"
Organizational Requirements for defining lifecycle concepts
Organizational Requirements for defining needs

Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 16: Needs Verification


Design Inputs
Rqmts Validation
"Are we building the right thing?"
Lifecycle Concepts & Needs Definition
Integrated Set of Needs
Transformation
Design Input Requirements Definition
Design Input Requirements
Transformation
Architecture & Design Definition
Design Output Specifications
Drive
Requirements Verification
"Are the requirements defined correctly?"
Organizational Requirement Definition Requirements

Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 17: Requirements Verification

Refer to the GtNR or similar guide for documenting an organization's standards, processes, checklists, and work instructions for defining lifecycle concepts, needs, and requirements.

Refer to both the NRM and GtNR for detailed guidance on validation and verification of needs and requirements.

Section 3: Characteristics of Sets of Needs and Requirements

This section defines the characteristics of sets of needs and sets of requirements, provides rationale for the characteristics, and provides guidance for helping understand the characteristics.

The achievement of the characteristics in this section is dependent on not only following the rules, but also doing the activities and analysis discussed in the NRM and GtNR and through the use of attributes defined in the NRM.

In addition to writing individual need and requirement statements that have the characteristics defined in the previous section, the following characteristics of a well-formed set of needs and set of requirements must also be considered.

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.

The characteristics discussed in this Guide for sets of needs and requirements can be grouped as follows:

***Formal Transformation**.* Given the set of needs and requirements is the result of a formal transformation, the following characteristics of the need and requirement set have been derived: • C10 - Complete • C11 - Consistent • C15 - Correct	***Agreed-to Obligation**.* Since the set of need and requirements is to be a result of a fair agreement to meet an obligation, the following characteristics of the set have been derived: • C12 - Feasible • C13 - Comprehensible • C14 - Able to be validated

3.1 C10 - Complete

Definition:
The set of needs and set of requirements for an entity should stand alone such that it sufficiently describes the necessary capabilities, characteristics, functionality, performance, drivers, constraints, conditions, interactions, standards, regulations, safety, security, resilience, and quality factors without requiring other sets of needs or sets of requirements at the appropriate level of abstraction.

Rationale:
A set of needs or set of requirements is incomplete when there are needs or requirements missing. Missing needs or requirements can result in significant shortcomings in the delivered product pertaining to needed functionality/performance, robustness, quality, or conformance resulting in a failure to validate the entity meets its needs and verify the entity meets its requirements. A set of needs or requirements cannot be complete if any individual need or requirement is not complete (C4).

If the formal derivation of needs from agreed-to lifecycle concepts and other sources results in individual needs that are necessary, the set of needs must be a sufficient and "accurate" representation of the lifecycle concepts and other sources from which they were transformed—that is, all the needs that are necessary have been included to implement the lifecycle concepts and other sources, and that any needs that are unnecessary have been excluded.

If the formal transformation of needs, higher-level requirements, and other sources into requirements results in individual requirements that are necessary, the set of requirements must be a sufficient and "accurate" representation of the set of needs, higher-level requirements, and other sources for the entity from which they were transformed — that is, all necessary requirements have been included and all unnecessary requirements have been excluded.

The resulting set of requirements will not be complete if the set of needs from which they were transformed was not complete.

Guidance:

The goal is to clearly communicate the needs for an entity via a minimum set that are necessary and sufficient and no more and clearly communicate the requirements for an entity via a minimum set that is necessary and sufficient and no more. This applies no matter the level in the architecture the entity exists.

Necessity and sufficiency of the requirement set can only be determined with respect to the set of needs and other sources of requirements for the entity, and the higher-level requirements allocated to the entity.

Completeness must also address the possible conditions a system may experience, including anomalies, safety, security, and resilience.

Completeness therefore requires an additional analysis of all possible conditions to ensure that the required behavior, functionality, and capabilities for specific conditions is consistent with stakeholder expectations. These are the kinds of analysis (what if cases, mis-use cases, and loss scenarios) that might not be addressed as part of initial analysis because the focus was on desired behavior, functionality, and capabilities under expected conditions. The resulting needs and requirements derived from this additional analysis must be included for the set of needs and set of requirements to be complete.

Off-nominal conditions may not arise until the system development is underway (or worse yet - after deployment) and represent risk across the lifecycle and difficulty in ensuring completeness. Unspecified off-nominal conditions can be managed using an "else" statement in the need statements and requirement statements conditional clauses. However, the required behavior under these off-nominal scenarios must be validated by the stakeholders.

For each entity, the set of needs represents a complete definition of the stakeholder expectations, stakeholder needs and requirements, goals, objectives, lifecycle concepts, drivers, constraints, and risks to be mitigated whether explicitly stated or undocumented implicit expectations.

Addressing implicit stakeholder expectations, needs, and requirements is a key part of lifecycle concept and needs definition activities and transforming the resulting set of needs into a set of requirements that when realized will result in an entity that meets those needs, even if not originally stated. *Any implicit needs should be captured and included explicitly in the set of needs. It is the responsibility of those defining the set of needs, to establish bi-directional traceability between each need and its source.*

A complete set of requirements set represents a complete transformation of the set of needs for an entity as well as a complete transformation of higher-level requirements that were allocated to the entity into a necessary and sufficient set of child requirements, that when implemented, will result in the intent of the allocated higher-level requirements being met.

It is the responsibility of owners of the allocated requirements to ensure these child requirements exist and the responsibility of the owners of the receiving entity to 1) validate the allocations were correct, 2) develop a necessary and sufficient set of child requirements, 3) provide bi-directional traceability between the child requirements and their parent and 4) notify the owners of the higher-level entity whenever a parent requirement is missing.

As an example, a set of software requirements may not be complete because not all relevant child requirements may be considered by the software architect if hardware limitations (for example, bandwidth, reliability, latency) are ignored. The hardware requirements and limitations may be missing or inconsistent with the assumed software architecture and software requirements. Therefore, it is important to consider the macro system of which the software is a part and include hardware stakeholders that can be the source of these missing needs and requirements.

Stakeholders are a primary source of needs and requirements; leaving out a relevant stakeholder could result in missing or incorrect needs and the resulting requirements, resulting in expensive and time-consuming rework.

With increasingly complex, software-intensive systems it is almost impossible for a person or even a group of people to completely understand and manage every aspect of an entity. A key tool to help define and manage these complex systems is the use of diagrams and modeling to ensure what is needed is included and what is not needed is excluded.

In the case of interactions across interface boundaries, the interface requirements need to refer to where the interaction between the entity and other entities is defined. In the case of standards and regulations, needs should reference the specific requirements that apply to the SOI, rather than the whole document. The set of requirements would then include derived requirements concerning what the entity must do to meet the need.

Completeness can be facilitated through traceability, allocation, and budgeting. It is the responsibility of those defining the set of and resulting set of requirements to establish bi-directional traceability between each need and lifecycle concept and other sources the needs were derived and between each requirement and the need, higher-level requirements, or other source the requirement is being transformed.

Completeness of the sets of needs and requirements can be assessed during early system validation as well as during design validation activities discussed in the NRM and GtVV.

Completeness of the sets of needs and requirements can also be facilitated through the use of templates for organizing sets of needs and requirement. Each organization will define types or categories in which a need or requirement fits, based on how they may wish to organize the requirements. Organizing by type/category is useful because it allows the stakeholders to view the sets of needs and requirements from a variety of perspectives. Each of these perspectives represents unique needs and requirements. Completeness can be aided by addressing each perspective. See R42 for examples of use of types/categories of needs and requirements such as functional/performance, fit (operational), form (physical attributes), quality (-ilities), and compliance (regulations and standards).

Refer also to the NRM and GtNR for additional guidance concerning completeness of sets of needs and sets of requirements.

3.2 C11 - Consistent

Definition:
A set of needs and a set of requirements is consistent if contains individual needs or requirements that are: • unique; • do not conflict with or overlap with others in the set; • makes use of homogeneous units and measurement systems; and • are developed using a consistent language (that is, the same words are used throughout the set to mean the same thing); and use terms that are consistent with the architectural model, project glossary, and project data dictionary.

Rationale:
Conflicting needs and requirements lead to an infeasible solution space, and, if not identified early in the development process, can lead to expensive rework. For the transformation to be formal, the resultant set of individual needs and requirements must not conflict and must be consistent with each other. Even if individual needs or requirements are unambiguous, the inconsistent use of terms, abbreviations, units, and measurement systems in different requirements results in ambiguity in the set of needs and set of requirements. Needs and requirements that are inconsistent and conflicting with other needs and requirements are not Correct (C8). Needs and requirements that are inconsistent and conflicting with other needs and requirements also result in a set of needs or requirements that are not Feasible (C12), are not Comprehensible (C13), and that are not Able to be Validated (C14).

Guidance:
Frequently, customer and other relevant stakeholder expectations or design constraints conflict with one another and need to be reconciled. It is a challenge to identify conflicts between needs and requirements when the set of needs and set of requirements is large, as is often the case in today's increasingly complex, software-intensive systems. It is important to avoid duplicate needs and requirements. Given the number of needs and requirements there may be multiple people inputting needs or requirements into the sets. A common issue is each of these people may use slightly different words to communicate a need or requirement that is addressing a single intent or action. This can lead to duplicate needs or requirements in the set. For example, one person may focus on requirements for an aircraft concerning airspeed, while another my person may add a requirement concerning flight performance measures, listing the specific flight performance measures in a data dictionary. Some duplication can be avoided through defining approved terms in a data dictionary or glossary and restricting the use of terms used in the various artifacts (including needs and requirements), to only those that are approved. Using NLP/AI tools to assist in the crafting of need and requirement statements can help enforce the use of only approved terms. As discussed in the NRM and GtNR, it is important to identify and link dependent needs and requirements that have relationships with other requirements, either directly or indirectly. This is

especially an issue when one need or requirement is changed without making a corresponding change to the other dependent need(s) or requirement(s).

Concepts to help prevent this type of inconsistency include allocation, budgeting, and linking (trace) dependent requirements to each other. When allocating and budgeting functionality, performance, or quality requirements to lower-level entities of the architecture, in many cases the resulting child requirements are dependent. To ensure consistency throughout the development lifecycle, these dependent requirements need to be linked together to ensure consistency is maintained when changes occur.

It can be difficult to identify conflicts merely through the language used to express individual need and requirement statements, but it can be made easier by classifying and grouping like needs and requirements together. This can be facilitated through the use templates for organizing sets of needs and requirement. Grouping like needs and requirements together will make it easier to identify inconsistencies.

See R42 for examples of use of types/categories of needs and requirements such as functional/performance, fit (operational), form (physical attributes), quality (-ilities), and compliance (regulations and standards).

Another strategy is to have the need statement or requirement statement within a common type or category to follow the defined pattern defined for that type or category as discussed in Appendix C. Using the pattern specific for a given type or category of need or requirement will aid in consistency.

Another key tool is the use of diagrams and other models that show the relationships and dependencies within sets of needs and sets of requirements as well as between needs and the resulting requirements. Using a software tool, such as diagraming and modeling, to manage relationships and dependencies can help in identifying conflicts and manage these relationships and dependencies.

Consistency in needs and requirements wording is greatly assisted using a centralized domain ontology, glossary, and data dictionary that is shared among all stakeholders.

Use an NLP/AI supplication to evaluate need and requirement statements within a set using the defied ontology, glossary, data dictionary, patterns, and templates to help establish consistency. Some applications of this type can be used as a digital assistant during the formulation of a need or requirement statement to both ensure the proper pattern or template for a given need or requirement type is used as well as help ensure consistent use of terminology.

Pay special attention to interface requirements to make sure they are consistent with the interface requirements for other entities with which the subject entity interacts. Ideally, both entities are managed within the same tool allowing interface requirements to be linked in applicable pairs and traced to a common definition concerning the specific interaction addressed within the pair of interface requirements (such as through the use of an ICD or a data dictionary).

Consistency within and between the sets of needs and requirements can be assessed during early system validation as well as during design validation activities discussed in the NRM and GtVV.

3.3 C12 - Feasible

Definition:
A set of needs and a set of requirements is feasible if it can be realized within entity constraints (such as cost, schedule, technical) with acceptable risk.

Rationale:

Just as there is little point in agreeing to an obligation for an individual need or requirement that is not feasible, the set of needs and set of requirements must be achievable within the appropriate constraints including cost, schedule, and risk.

If feasibility is not addressed early in the development process, it can lead to wasted effort and cost.

A set of needs or set of requirements cannot be feasible if any of the individual needs or requirements are not feasible (C6). Feasibility of a specific need statement or requirement statement must be determined in the context of all needs or requirements for a specific entity set and can be evaluated without regard to the sources from which they were derived.

If not feasible within the stated constraints with risk appropriated for that lifecycle stage, the set of needs will not be feasible nor will be the resulting set of requirements that are transformed from that set of needs.

A set of requirements may not be feasible due to inconsistency (C11), physical impossibility, or excessive program risk.

Guidance:

While individual need statements and requirement statements may be feasible, they may not be so when placed in combination with others. That is, the combination of feasible individual needs or requirements does not necessarily sum to a feasible set of needs or feasible set of requirements.

For example, the following are feasible, individual requirements for a laptop computer: weighs less than 0.9 kg, has a storage capacity of 1 TByte, has 4 GByte of RAM, has a wireless network interface, has an Ethernet network interface, has two USB-C interfaces, has an HDMI interface, can be dropped from 1 meter without damage, can survive in temperatures of ±50°C, and retails for less than $900. While each of those individual requirements seems perfectly feasible, even at first glance to a non-expert, we cannot be so readily sure that the set is feasible (that is, all requirements can be met simultaneously). As soon as we go past a couple of dimensions, our human intuition quickly deserts us.

If the related individual needs or requirements are distributed throughout the need or requirement set, it is even more difficult to assess the feasibility of the set. As was stated for individual need statements and requirement statements, determining feasibility of a set of needs or set of requirements is not always completely known and is often assessed in terms of acceptable risk consistent with the development lifecycle stage.

In the solution space, there should be at least one, and preferably multiple, feasible sets of lifecycle concepts that will result in the problem being solved or opportunity to be realized that drove the need for the SOI. The lifecycle concepts should be technically feasible (such as in terms of technology maturity level and advancement), and feasible within the constraints of the project (such as cost, schedule, technical, ethical, and regulatory compliance) with a level of risk consistent with the lifecycle stage.

As discussed in the NRM, as part of risk assessment and during lifecycle concept analysis and maturation critical technologies are identified, their TRL and Advancement Degree of Difficulty (AD^2) is assessed, and a technology maturation plan is developed that will result in the critical technology maturity to advance such that it will be feasible in time to meet the system development schedule. These activities represent a risk to the project and as such this risk must be managed closely.

Needs within the same entity set of needs are analyzed together with respect to program constraints and technology maturity to determine feasibility.

Before allowing a set of needs to be baselined, an assessment of the feasibility of the chosen lifecycle concepts should be made in terms of the drivers, project constraints (cost, schedule, technology), and risk.

Feasibility of a specific requirement must be determined in the context of all requirements for a specific entity set and cannot be evaluated without regard to the higher-level requirements and other dependent child requirements of that higher-level requirement.

Requirements within the set of requirements for the same entity are analyzed together with respect to project constraints, technology maturity, and risk to determine feasibility.

Feasibility of the sets of needs and requirements can be assessed during early system validation as well as during design validation activities discussed in the NRM and GtVV.

Additional characteristics mapped to feasible include “Consistent” (C11) and “Able to be validated” (C14).

An approach that can be used is to consider the set of needs or requirements to be a “bucket” shown in Figure 12 which is bound by cost, schedule, and technology. When adding individual needs or requirements to the bucket, whether the individual needs or requirements “fit” within these constraints must be determined. When the bucket is full, the addition of any more needs or requirements puts the project at increased risk. The bucket analogy is also useful for addressing change. If a bucket is full, and someone wants to add an additional need or requirement to the bucket, the question of feasibility must be addressed. Can the bucket be made bigger? Can something of lower priority be taken out? Is the project willing to accept more risk? If the answer is no, then the project must say no to the change. Refer the NRM for a more detailed discussion concerning the use of both the needs bucket and requirements bucket when addressing feasibility and risk.

A sensitivity analysis may be a useful tool for the user and requirement writer to determine the 'realm of the possible" when aggregating many needs or requirements that may be individually feasible, while not feasible in sum. Through modeling, the users may be able to vary the needs or requirements while analyzing the resulting system performance. This would allow optimization of the need or requirement set, reducing some (or all) to lower but yet acceptable levels, to improve overall system performance.

3.4 C13 - Comprehensible

Definition:

The set of needs and the set of resulting requirements must each be written such that it is clear as to what is expected of the entity and its relation to the macro system of which it is a part.

Rationale:

An agreement is difficult to enact unless both parties are clear on the exact obligation and the expected outcome(s) as a result of the realization of the entity the set of needs and requirements represents.

These sets must therefore be written such that the relevant audience can understand what is being communicated by the individual needs and requirements as well as the sets of needs and requirements.

To be comprehensible, the sets of needs or requirements must be Complete (C10), Consistent (C11), Feasible (C12), and Able to be Validated (C14).

A set of requirements is comprehensible if and only if all individual requirements in the set are Necessary (C1), Unambiguous (C3), Complete (C4), Feasible (C6), Verifiable/validatable (C7), and Correct (C8).

Guidance:

Information needed to understand the context of the needs and requirements should be included within the set. This includes useful information attached to the set that defines the context defined during lifecycle concept and needs definition as well as attributes of the needs and requirements such as rationale.

In a document-centric practice of SE, this information is commonly included in the introductory material at the beginning of a documented set of needs or requirements. In a data-centric practice of SE, this information can be documented with in the tool and linked to the set such that it can be included when the set is outputted in the form of a document.

Using a data-centric practice of SE, it will be easier for the architects, designers, builders, coders, and testers if the underlying analysis (including models) is provided along with the sets of needs and requirements, or if the system engineers rely on a defined project data dictionary where the information can be centralized.

Using an integrated, collaborative, multidiscipline project team performing the development process activities concurrently will help stakeholders comprehend what is expected of the entity and its relation to the macro system of which it is a part.

Coordinating with and getting agreement of the stakeholders and getting their feedback during development activities will help ensure the stakeholders and developers are on the same page.

Having a well-defined baseline process and involving the stakeholders in that process helps ensure comprehensibility of the sets of needs and requirements.

Continuous validation of artifacts across the lifecycle also will help ensure the artifacts are comprehensible.

Comprehensibility of the sets of needs and requirements can be assessed during early system validation as well as during design validation activities discussed in the NRM and GtVV.

3.5 C14 - Able to be Validated

Definition:
It must be possible to validate that the set of needs will lead to the achievement of the product goals and objectives, stakeholder expectations, risks, and lifecycle concepts within the constraints (such as cost, schedule, technical, legal and regulatory compliance) with acceptable risk. It must be possible to validate that the set of requirements will lead to the achievement of the set of needs and higher-level requirements within the constraints (such as cost, schedule, technical, and regulatory compliance) with acceptable risk.

Rationale:

Referring to Figure 5, the transformation of lifecycle concepts into the Integrated Set of Needs and transformation of the Integrated Set of Needs into sets of Design Input Requirements must be formal and able to be validated, not just for individual needs and requirements, but also for the sets of needs and requirements.

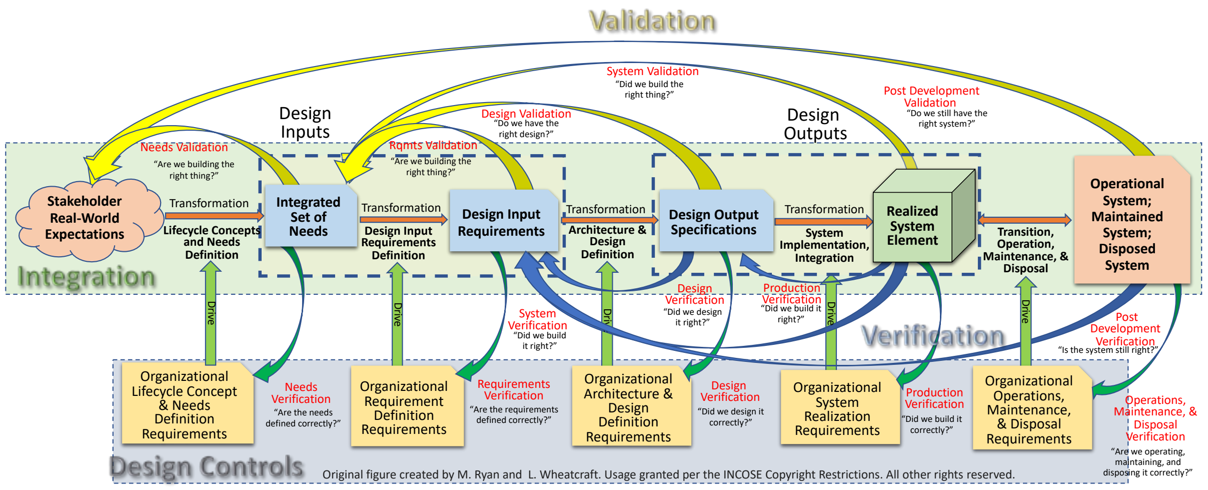

Validation
System Validation
"Did we build the right thing?"
Post Development Validation
"Do we still have the right system?"
Design Validation
"Do we have the right design?"
Design Inputs
Design Outputs
Needs Validation
"Are we building the right thing?"
Rqmts Validation
"Are we building the right thing?"
Stakeholder Real-World Expectations
Transformation
Lifecycle Concepts and Needs Definition
Integrated Set of Needs
Transformation
Design Input Requirements Definition
Design Input Requirements
Transformation
Architecture & Design Definition
Design Output Specifications
Transformation
System Implementation, Integration
Realized System Element
Transition, Operation, Maintenance, & Disposal
Operational System; Maintained System; Disposed System
Integration
Drive
System Verification
"Did we build it right?"
Design Verification
"Did we design it right?"
Production Verification
"Did we build it right?"
Verification
Post Development Verification
"Is the system still right?"
Organizational Lifecycle Concept & Needs Definition Requirements
Needs Verification
"Are the needs defined correctly?"
Organizational Requirement Definition Requirements
Requirements Verification
"Are the requirements defined correctly?"
Organizational Architecture & Design Definition Requirements
Design Verification
"Did we design it correctly?"
Organizational System Realization Requirements
Production Verification
"Did we build it correctly?"
Organizational Operations, Maintenance, & Disposal Requirements
Operations, Maintenance, & Disposal Verification
"Are we operating, maintaining, and disposing it correctly?"
Design Controls
Original figure created by M. Ryan and L. Wheatcraft. Usage granted per the INCOSE Copyright Restrictions. All other rights reserved.

Figure 18: Verification and Validation Across the Lifecycle

As shown in Figure 18, it must be able to be shown at any lifecycle stage that achievement of the Integrated Set of Needs will result in a SOI that can be validated to meet the stakeholders' real-world expectations. That is, "Are we building the right thing?"

It must also be able to be shown at any lifecycle stage that achievement of the set of Design Input Requirements transformed from the Integrated Set of Needs will result in a SOI that can be validated to satisfy those needs - Are we building the right thing?

For a set of needs or requirements to be able to be validated, the set must be Complete (C10), Consistent (C11), Feasible (C12), Comprehensible (C13), and Correct (C15).

Refer to the NRM, GtNR, and GtVV for more details on validating sets of needs and sets of requirements.

Guidance:
The system lifecycle operational scenarios, concepts, and use cases—from which the needs were transformed can be used as test cases to validate the sets of needs, sets of requirements, the design, and built or coded SOI.

In other words, the needs can be validated to meet the lifecycle concepts and other sources from which they were derived, and the requirements can be validated to meet the needs, higher-level requirements, and other sources from which they were transformed by running tests based on the use cases or operational scenarios developed during lifecycle concept and needs definition activities discussed in the NRM and GtNR. Doing so will result in evidence that the intended use, goals, objectives and stakeholder expectations have been met within the agreed-to drivers and constraints with acceptable risk.

Design validation is conducted to ensure the transformation of the Design Input Requirements into the design and resulting design output specifications will result in a system that meets its intended purpose/use in its operational environment as defined by the Integrated Set of Needs. Thus, design validation goes back to the needs to make sure they have been met—as a set.

System validation is conducted to ensure the built and verified system meets its intended purpose/use in its operational environment when operated by its intended users and does not enable unintended users to use the system in an unintended way. Thus, system validation goes back to the Integrated Set of Needs, lifecycle concepts, and other sources to make sure they have been met—as a set.

Refer to the NRM and GtVV for more details on design and system validation.

For the set of needs, ask the questions:

“Do the needs and set of needs clearly and correctly communicate the agreed-to lifecycle concepts, drivers, constraints, risks, and stakeholder expectations?”

“Have we correctly and completely captured the needs the system must address?”

“Are we building the right thing?”

For the set of requirements ask the questions:

“Will the entity developed by this set of requirements satisfy the needs?”

“Are we building the right thing?”

These questions concerning the sets of needs and sets of requirements are a major focus during early system validation as well as during design validation and system validation activities discussed in the NRM and GtVV.

3.6 C15 - Correct

Definition:
The set of needs must be an accurate representation of the lifecycle concepts or sources from which it was transformed. The set of requirements must be an accurate representation of the needs, sources, or higher-level requirements from which it was transformed.

Rationale:
A correct set of needs implies “no errors” both from the perspective of the inclusion of incorrect information, the omission of required information, and avoidance of incorrect individual need statements. An incorrect Integrated Set of Needs does not reflect the true intent of the lifecycle concept or sources from which it was transformed. A correct set of requirements implies “no errors” both from the perspective of the inclusion of incorrect information, the omission of required information, and avoidance of incorrect individual

requirement statements. An incorrect set of requirements does not reflect the intent of the Integrated Set of Needs, sources, or higher-level requirements from which it was derived.

The set of needs or the set of requirements cannot be correct if it does not have the characteristics for sets of needs and sets of requirements: C10 - Complete, C11 - Consistent, C12 - Feasible, C13 - Comprehensible, and C14 - Able to be validated.

Guidance:

As shown in Figure 5, the Integrated Set of Needs is transformed from various sources (such as lifecycle concepts, stakeholder expectations, drivers and constraints, goals, objectives, and risks), defined as part of the lifecycle concept and needs definition activities discussed in the NRM and GtNR.

As shown in Figure 14, validate that the resulting set of needs reflects an accurate and unambiguous interpretation of the:

- concepts, stakeholder expectations, drivers and constraints, goals, objectives, risks, etc. from which it was transformed;
- problem or opportunity and underlying goals and objectives;
- the models or diagrams from which the individual needs were extracted so the needs trace to the models and diagrams; and
- underlying analysis and assumptions that were part of the transformation.

Also validate that the achievement of the set of needs, as written, will result in the realized system meeting the intent of the sources from which it was transformed.

As shown in Figure 15, validate that the resulting set of requirements reflects an accurate and unambiguous interpretation of the:

- need, source, or higher-level requirement from which it was transformed;
- models or diagrams from which the individual requirements were extracted so the requirements trace to the models or diagrams; and
- underlying analysis and assumptions that were part of the transformation.

Also validate that the achievement of the set of requirements, as written, will result in the realized system meeting the intent of the needs, sources, or higher-level requirements from which it was transformed.

Use a defined development and management process to ensure accuracy of the transformation in the context of the individual needs and requirements as well as the set of needs and set of requirements transformed from those needs.

Perform the needs and requirements validation described in the NRM and GtNR to ensure correctness of the transformation in the context of the individual need and requirement statements as well as the complete sets of needs and requirements.

Correctness of integrated sets of needs and sets of requirements can also be assessed during early system verification and design verification activities discussed in the NRM and GtVV.

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 type of 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 or qualifying clause 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 about patternscan 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: During Login, 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: During Login, 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 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 and characteristics: 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.] The characteristics of the Audio received from a <xxxx> source is defined in the ICD.]

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.

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 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. As a system level requirement, it will be allocated to the lower-level subsystems and system elements that are involved in computing the current time and displaying the current time. 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 articles "a" or "an".

Elaboration:

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:

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: When powered on, 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: When powered on, 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 ATM shall display relevant Customer information. [This is unacceptable because it does not make explicit which elements of information 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: *[Split into as many requirements as necessary to be explicit (and to avoid and open-ended statement in accordance with R9. Note that the types of customer information to be displayed also 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: See also the guidance of R9, R17, R18, R22, and R28. Note also that the above examples are incomplete in that expected performance and conditions have not been included.]

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 conditions have not been included.]

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 <an 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: While <condition>, 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: While <condition>, the 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. [This is unacceptable because it inserts a phrase in such a way that the object of the action is separated from the verb.] Improved: The Navigation_Beacon shall provide to each Maritime_User, Augmentation_Data having the characteristics defined in <ICD xyz> at an accuracy of less than or equal to 20 meters. [This rewrite places the basic action and object of the action in an unbroken clause followed by the sub-clause describing performance. Note that: “Navigation_Beacon”, and “Maritime_User”, must be defined in the glossary or data dictionary. Note also that there may well be a number of conditions (such as location, mode or state) under which the data is to be provided to the user. Further, when discussing accuracy, precision also needs to be addressed. These and other issues may also be defined in an ICD.]

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: While <condition>, the Weapon_System shall *storing* the location of all ordnance.

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

Improved: While <condition>, 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: While 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: While 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: While <condition>, 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: While <condition>, 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 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.

[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 navigation data.]

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

[The positioning of the comma now makes it clear that the accuracy relates 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: While 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)

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

While 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: When < off-nominal condition>, the Operation_Logger shall record the Warning_Message produced by the system within <performance measure>.

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

Improved: When <off-nominal condition>, the Operation_Logger shall record the 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:

Personal 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 the use of the glossary to define Operating_Hours. Alternatively, the unacceptable requirement could be stated as a need to always be available during operating hours. 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 while considering what is feasible, practical, and verifiable.]

Unacceptable: The <SOI> shall never fail.

[Never implies “never, ever” under any conditions. This is unacceptable because “never, ever” under any conditions is impossible to verify without an infinite number of verification or validation activities.

Improved:

The <SOI> shall have a Reliability of greater than or equal to 99% over its Operational_Life.

Or

The <SOI> shall have a Mean Time to Failure of 1000 Operating_Hours.

[Note use of the glossary to define Operating_Hours and Operational_Life. Alternatively, the unacceptable requirement could be stated as a need for high reliability. 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 while considering what is feasible, practical, and verifiable.]

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 containing the information defined in <some standard or contract deliverable listing>.

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 their 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. The use of "their" is acceptable as it is clear what the meaning is.]

Unacceptable: When a pedestrian signals their 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].

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: When <off-nominal condition, 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: When <off-nominal condition, the Operation_Logger shall record each Warning_Message <within some performance value>.

[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.]

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

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

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.

<Any conditions or performance must also be included in the requirements statement for it to be complete.>

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 both design the system to as well as verify the system against. 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)'

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: While in operation,_the Pumping_Station shall maintain the flow of water at 100 liters per second.

[This is unacceptable because we do not know whether a solution that addresses more or less than the specified quantity is acceptable. As stated, “100” can be interpreted as “100.000”. It would be difficult and expensive to design to and verify against this point value.]

Improved: (addressing the range defect.)

While in operation, the Pumping_Station shall maintain Water_Flow at 100 ±10 liters per second.

[Now the range of acceptable flow performance is clear. Given that the condition “while in operation”, this requirement applies whenever the Pumping_Station is active—no matter how long. Another consideration is whether there is sufficient water at the inlet to the pumping station to establish the required flow rate.]

Improved: (Addressing the water level at the inlet and what triggers the pump to being operation.)

When the water level rises above the inlet to the Pumping_Station, the Pumping_Station shall maintain the Water_Flow at 100 ±10 liters per second.

When the water level drops below the inlet to the pumping station, the Pumping_Station shall cease operations.

Unacceptable: The Flight_Information_System shall display per <Display Standard xyz> 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 per <Display Standard xyz> 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.

“Threshold” vs. “Goal” or “Objective” requirements.

In some cases, measurable performance is communicated using a minimum acceptable “threshold” value as well as including a higher target value as a “goal” or “objective”.

This leads to the question as to what an acceptable approach is to communicate threshold values vs objective or goal values within a need or requirement statement.

The answer depends on whether the system is being developed in-house or will be contracted out to a supplier. A key question to address is what will be the value the project or supplier is going to be held accountable to meet, i.e., what value will the system be verified or validated against? What value is required to have been met for acceptance?

Using the threshold/objective approach. The customer has a need for a system with a certain performance parameter. Current, state-of-the-art performance is not good enough to meet the customer's needs. To address this, the customer does an analysis as to what minimal value will meet their needs. This value would be what is communicated within the need or requirement statement and is what the developed system would be verified and validated to meet.

However, in many cases the customer would realize increased value if the developed system had a performance greater than the minimum threshold value stated in the need or requirement.

In this case, the customer would do more research and analysis to determine what additional performance may be possible based on technologies being developed either in house or in the marketplace. They could use the concept of technology readiness levels (TRLs) to assess risk. (Refer to the NRM for a more detailed discussion on TRLs; the lower the TRL the higher the risk.)

Based on their research and analysis the customer would identify a higher value that they think is technically feasible given the TRL, the available time and available budge with acceptable risk. In this case, the minimal value would be defined as the threshold value which is the minimum value for acceptance and define the higher possible value as the goal or objective value.

If an in-house project, the customer could include both threshold and objective values in the same requirement statement (shall), or separate requirement/goal statements (shall/should) explaining in the rationale that the threshold value is the minimum acceptable (from a verification standpoint), but the project would like to get closer to the "objective or goal" value. "The system shall <perform some function> with a [performance parameter threshold value] of xxxx, with an objective [or goal] value of yyyy." Or "The system shall obtain the [parameter threshold value] of xxxx." and "The system should obtain an objective [or goal] value of yyyy."

While the requirement could be written as "greater or equal", communicated as a minimum, or as a range, often developers will do only the minimum. Using the threshold/objective (or goal) approach the intent of the customer is clearer. In both cases the intent must be clearly communicated within the rationale attribute for the need or requirement.

Alternately, the customer could state the greater goal/objective as the required value. However, this could add risk of schedule and budget overruns given the uncertainties of realizing a system that can meet this value, especially for low TRL technologies.

When issuing a Request for Proposal (RFP) or Request for Tender (RFT), to suppliers, the customer would state the requirement in the threshold/objective(goal) form. In the proposal the suppliers would be asked to propose a value they think is feasible and will commit to.

Some suppliers may bid the threshold value, others the objective value, and others somewhere in between. The customer would evaluate each of the proposals and everything else being equal and their confidence in the ability of the bidder to meet their proposed value, the customer would select the supplier that bids the best value for the parameter (closest to the objective(goal) value). From a contract perspective, this bid value becomes the required value against which the system will be verified against, and acceptance is based.

Examples:

Unacceptable: While in operation, the <SOI> shall use minimum power.

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

Improved: While in operation, the <SOI> shall consume less than or equal to 50W of Main_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 <unit of measure>.

[Where xxx represents the required threshold value, including the appropriate units. The project team would access the feasibility of the value prior to baselining the requirement.]]

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 less than or equal to the value defined in <Clause xyz of Standard XYZ>.

[While it is common practice to reference a standard in cases like this, there may be issues in the wording of the standard as it applies to your specific SOI. Also, what happens if the values in the standard change or an assessment of feasibility has not been done. To avoid these issues, it would be preferable for the project team to do the analysis necessary to compute a feasible value and include that value in the requirement statement.

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. For the last example, it would be acceptable for a need to reference a specific clause in a standard, and then when transforming the need into a requirement, the project team would do the necessary analysis to determine feasible requirements that would meet the intent of the need.

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: When <condition>, 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: When <condition>, the Pump shall remove greater than 99% of the Fluid from the Tank in less than 3 hours of continuous operation.

4.13 Uniformity of Language

4.13.1 R36 – Consistent Terms and Units

Definition:
Ensure each term and unit of measure used throughout need and requirement sets as well as associated models and other SE artefacts developed across the lifecycle are consistent with the project’s defined ontology.

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.)

Appendix A: References

The following references were consulted when developing this Guide. When possible, the source is referenced in the text. In some cases, concepts and information from several sources was used as a basis of the text but are not directly attributable to any single source. Referenced documents are applicable only to the extent specified herein.

1 ANSI/AIAA G-043-2012e, *Guide to the Preparation of Operational Concept Documents*.

2 Carson, R. S., et al. (2004). Requirements Completeness. Proceedings of INCOSE 2004. Wiley.

2a. Carson, R. S. (2015). Implementing Structured Requirements to Improve Requirements Quality. Proceedings of INCOSE. Seattle, WA USA: Wiley.

3 Carson, R. S., Noel, R. A., “Formal Requirements Verification and Validation,” INCOSE IS 2018.

4 Dick, J. and J. Chard, “The Systems Engineering Sandwich: Combining Requirements, Models and Design”, *INCOSE International Symposium IS2004*, July 2004.

5 Dick, J. and Llorens, J., "Using Statement-level Templates to Improve the Quality of Requirements", *International Conference on Software and Systems Engineering and Applications. ICSSEA 2012*, Paris, France.

6 Doran, G., *Management Review*, November 1981.

7 Génova, G, Fuentes J.M., Llorens, J., Hurtado, O., and Moreno, V., “A Framework to Measure and Improve the Quality of Textual Requirements", *Requirements Engineering*, Vol 18, 2013.

8 Gilb, T. *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering and Software Engineering Management Using Planguage*, Elsevier Butterworth-Heinemann, 2005.

9 Heitmeyer, C., J. Kirby, and B. Labaw, “The SCR Method for Formally Specifying, Verifying, and Validating Requirements: Tool Support”, *Proceeding ICSE '97 Proceedings of the 19th international Conference on Software Engineering*, pp. 610-611, Boston, Massachusetts, USA — May 17 - 23, 1997.

10 Hull, E., K. Jackson, J. Dick, *Requirements Engineering*, Springer, 2011.

11 INCOSE-TP-2003-002-04 2015, Jan 2015 Systems Engineering Handbook, Fourth Edition, I

12 INCOSE-TP-2018-001-01, 2018, INCOSE Integrated Data as a Foundation of Systems Engineering, prepared by the Requirements Working Group, INCOSE.

13 INCOSE-TP-2021-002-01.1, 2022 Needs and Requirements Manual, prepared by the Requirements Working Group, INCOSE

14 INCOSE-TP-2021-004-01, 2022 Guide to Verification and Validation, prepared by the Requirements Working Group, INCOSE

15 INCOSE -TP-2021-003-01, 2022, Guide to Needs and Requirements, prepared by the Requirements Working Group, INCOSE

16 ISO/IEC/IEEE 15288, Systems and software engineering — System lifecycle processes, First edition 2015-05-15

17 ISO/IEC/IEEE 29148, Systems and software engineering — Lifecycle Processes — Requirements engineering, Second edition, 2018-12.

18 Mavin, A, Wilkinson, P, Harwood, A, Novak, M, *EARS (Easy Approach to Requirements Syntax)*, RE09, IEEE, August 2009

19 Ryan, M.J., “An Improved Taxonomy for Major Needs and Requirements Artefacts”, *INCOSE International Symposium IS2013*, June 2013.

20 Ryan, M., Wheatcraft, L, Dick, J, and Zinni, R, “An Improved Taxonomy for Definitions Associated with a Requirement Expression”, *Systems Engineering / Test and Evaluation Conference SETE2014*, Adelaide, 28-30 April 2014.

21 Ryan, M., Wheatcraft, L., “On the Use of the Terms Verification and Validation”, *INCOSE International Symposium IS2017*, July 2017.

22 *Simplified Technical English (STE)*, Specification (ASD-STE100), ASD Belgium, http://www.asd-ste100.org/.

23 US Code of Federal Regulations: Title 21, Part 820, Quality System Regulation

24 Wheatcraft, L., Ryan, M., Dick, J. “On the Use of Attributes to Manage Requirements”, *Systems Engineering Journal*, Volume 19, Issue 5, September 2016, pp. 448-458.

25 Wiegers, K.E., *Software Requirements*, Redmond, WA: Microsoft Press, 2003.

26 Wiegers, K.E., *More About Software Requirements*, Redmond, WA: Microsoft Press, 2006.

Appendix B: ACRONYMS AND ABBREVIATIONS

AI	**Artificial Intelligence**
CM	Configuration Management
ConOps	Concept of Operations
COTS	Commercial off-the-shelf
GtNR	Guide to Needs and Requirements
GtVV	Guide to Verification and Validation
GtWR	Guide to Writing Requirements
ICD	Interface Control Document
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
KDN	Key Driving Need
KDR	Key Driving Requirement
NLP	Natural Language Processing
NRM	Needs and Requirements Manual
OpsCon	Operational Concept
PM	Project Management
PMP	Project Management Plan
RMT	Requirement Management Tool
RWG	Requirements Working Group
SA	Supplier Agreement
SAFe	Scale Agile Framework®
SE	Systems Engineering
SE HB	Systems Engineering Handbook
STE	Simplified Technical English
SOI	System of Interest
SOW	Statement of Work
SysML	Systems Modeling Language
TBD	To Be Determined
TBS	To Be Supplied
TBR	To Be Resolved
TRL	Technology Readiness Level
UML	Unified Modeling Language

Appendix C: Patterns

C.1 Introduction to Patterns

The concept of patterns is mentioned in several sections of this Guide. Patterns are used to help provide a more uniform way of writing need statements and requirement statements, thus enhancing consistency (C11), and facilitating the use of a controlled vocabulary (data dictionary). This Appendix provides additional elaboration on the use of patterns when defining well-formed needs statements and requirement statements as is discussed in rule R1.

The notion of patterns (also called boilerplates or templates) was initially applied within the Future Surface Combatant (FSC) defense project in the United Kingdom in 1998 (Dick and Llorens, 2012) as an aid to solve several difficulties when writing different types of textual requirements (timeliness, etc.). Once requirement writers were shown example patterns for requirement statements based on a given type, the difficulty in writing a properly formed requirement statement was largely overcome.

“Structured and normalized” examples of types of requirements can be defined in patterns. A pattern may be structured as a sequential list of placeholders, including words, along with syntactic or semantic restrictions.

For example, a basic pattern for defining a well-formed requirement statement is:

“The <entity> shall <do what>, <how well>, <under what conditions>.”

“The <stakeholders> need the <entity><what>, <how well>, <under what conditions>.”

These placeholders are generally called *pattern slots*. The need or requirement statement text is written per a need or requirement pattern that is appropriate to the type of need or requirement and what is to be communicated. The following example represents a requirement statement and the associated candidate pattern. In this example, the requirement pattern has 6 pattern slots. Out of those pattern slots, some are fixed with the pattern, such as the *shall* or the determiner *a*, while most are placeholders to choose an element from the project data dictionary (such as the name of the *SOI*, the specific *Physical property*) or a global requirements dictionary (such as the *Unit*, the *Operator*).

The Power System shall have an availability greater than or equal to 98%.

The <SOI> <shall> <have> a <PHYSICAL_PROPERTY> <OPERATOR> <NUMBER> <PERCENTAGE/UNIT OF MEASURE>

Currently, the terminology for defining the abstractions that represent common natural language structures at the syntactic and semantic level is not mature. Several approaches are used in the literature (see references at the end of this appendix) with similar intent.

One common approach uses the term “templates”. The term “template” is most often used in the field of requirements management to refer to the structure and organization of a set of needs or set of requirements communicated in a document form. The use of such templates helps ensure that the authors consider the complete range of concerns and corresponding types of requirements when organizing the sets.

For requirement statements, some authors have coined a more detailed term: “statement-level template” to avoid confusion with templates for requirement sets (Dick and Llorens, 2012).

In another approach, Hull et al (2002) uses the term “boilerplates” to refer to a grammatical structure for an individual textual requirement statement.

The term “pattern” has a clear meaning in software (Gamma et al, 1994) and recently systems engineering, usually representing a reusable structure that resolves a problem by instantiating and configuring this structure to become a solution. INCOSE and others promote this term for use in the system engineering lifecycle (INCOSE Patterns Working Group). To align with these uses, this Guide uses the term “patterns” to represent the concept of structured, natural language need statement and requirement statement presenting syntactic and semantic information (restrictions and properties).

C.2 Benefits of Using Patterns

As this Guide describes, there are multiple factors that contribute to the realization of quality need statements and requirement statements and sets of needs and sets of requirements. One important method to assist in achieving quality need and requirement statements and sets of needs and sets of requirements is by the proper definition and agreement of a set of patterns with which the need statements and requirement statements must comply as stated in rule R1.

Ensuring that needs and requirements conform to agreed patterns and being able to identify the specific elements within a need statement or requirement statement contributes to the ability to verify the needs and requirements are well-formed having the characteristics and complying with the rules discussed in this Guide.

This ability is necessary to be able to measure the quality of the need statements and requirement statements and provide actionable information based on any deficiencies related to a specific statement.

The use of patterns enables the development of NLP/AI tools, not only to verify whether a given statement meets a particular pattern, but also to guide the author when forming needs and requirements, as a kind of “digital assistant”. These tools, along with a project ontology (data dictionary), can aid the definition of quality need and requirement statements.

The use of pattens also helps to validate that the requirement statements effectively communicate the intent of the needs from which they were transformed and validate the need statements effectively communicate the intent of the lifecycle concepts and other sources from which they were transformed.

In addition, defining need statements and requirement statements following a set of agreed patterns also makes it easier to:

- write easy-to-read and concise requirements, avoid unnecessary information (R20);
- write atomic (R18) need statements and requirement statements;
- find duplicates (R30) among sets of requirements and needs;
- classify needs and requirements within a set of needs or set of requirements (R29);
- identify missing information within need and requirement statements (*completeness – C4*);
- ensure consistency within a set of needs and requirements (C11); and
- support other activities such as modeling, analysis, and implementation.

Following a set of agreed-upon patterns also leads to a consistent set of needs and set of requirements, and the automatization of several needs and requirements management activities such as the:

- identification of duplicate and conflicting needs and requirements;
- verification of consistency and completeness rules associated with other systems; engineering artifacts generated across all system lifecycles (for example, consistency of requirements versus design as stated in a model and consistent use of terms defined in the project ontology);
- extraction of entities, functions, and properties from need statements and requirement statements;
- suggestion of traces that link needs and requirements in one of set of needs or set of requirements to other sets representing needs and requirements for entities defined at different levels of decomposition;
- suggestion of traces between textual requirements and model elements;
- transformation of textual requirements into models;
- extraction of needs and requirements from documents; and
- translation of need and requirement text to different languages.

Finally, patterns can also facilitate properly defining and teaching a common methodology for defining need statements and requirement statements and leverage the use of a consistent, defined vocabulary.

C.3 Basic structure of a need or requirement pattern

Requirement statement patterns.

As stated previously in this Guide, the structure of requirements statements must be in the form of a complete sentence, the simplest form of which is:

<subject (entity)> shall <verb (action)><object><response>

The *subject of the sentence* is essential because it is the *entity* undertaking the *action* (and is therefore the *entity* that will be subject to any verification/validation activity); the *verb* that comes before the *object* is essential because it is the *action* being performed; the *object* is essential because it must be clear as to which entity is being acted upon, and the *response* is the required result of the action being performed.

Both, subject and objects are therefore normally entities coming from a data dictionary or model and, for a requirement statement, the action is introduced by a "shall" as described in Section 1. Use of "shall" makes it clear that what is being communicated is formal, the statement is a requirement, the statement is legally binding, and the entity will be verified against the requirement.

Expanding on the <*subject*> <*verb*> <*object*><*response*> form, the *response* may be further elaborated to address a measurable outcome:

The <entity> shall <action verb> <object> <measurable outcome>.

In terms of a function/performance requirement, the <*action verb*> <*object*> pair is communicated as a function/object pair (such as, "heat coffee") and the <measurable outcome> is a specific response in terms of a required performance outcome resulting from that function (such as, "to a temperature greater than 120 degrees Fahrenheit".)

For a requirement involving an interaction between two entities (interface requirement) the <action verb> indicates the type of interaction, the <object> is what is involved in the interaction (together with the <subject>), and the <measurable outcome> is a pointer where the specific interaction is defined (e.g., Interface Control Document).

The <SOI > shall establish communications with the <object> as defined in <ICD xyz>.

Need statement patterns.

Need statements have a similar form:

The <stakeholders> need the <entity><entity response>. or

The <stakeholders> need the <entity><action verb> <object> < measurable outcome>.

Because need statements communicate a different perspective as discussed in Section 1, the subject is the stakeholders that have the need and entity is the entity to which the need applies. It is this entity that will be the subject of the requirements that are transformed from the need statement, and it is this entity that will be validated against the need.

For need statements, the outcome may be at a higher level of abstraction and therefore may not be as rigorous a <measurable outcome> as for a requirement statement as discussed in Section 2 and R3—for example, the measurable outcome for a need statement may be more qualitative (even perhaps requiring subject assessment) than the quantitative outcome (requiring objective assessment) of a requirement statement. Once again, the NLP/AI tools aiming at helping engineers define well-formed need statements and requirement statements must take this level of abstraction into consideration.

Qualifying Clauses

Expanding on the <subject> <verb> <object> form, a need statement or requirement statement may also include a <qualifying clause> to provide additional information needed to clearly communicate the intent of the requirement:

The <entity> shall <action verb> <object> <response - measurable outcome> <qualifying clause>.

If a <qualifying clause> is needed to clearly communicate the intent of the action verb, or the object 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 that qualifying clause is included, .e.g., performance associated with the action verb, or for an interface requirement where a pointer to where the specific interaction is defined (such as in an ICD). Example qualifying clauses include: "….at end of Operational _Life", "….after being dropped 3 feet on to a concrete surface." "….throughout the Operational _Life." Note: Operational _Life would be defined within the project data dictionary.)

Condition Clauses

Ubiquitous needs and requirements are those not requiring a condition (they are always *active*) not requiring a condition to trigger the desired *system response* (See Section 1.11 for a more detailed discussion concerning ubiquitous needs and requirements.)

The basic structure for a ubiquitous need statement is as follows:

The <stakeholders> need the <SOI> <system response>.

The basic structure for a ubiquitous requirement statement is as follows:

The <SOI> shall <system response>.

However, a need or requirement is often not ubiquitous and only applies under a certain condition or a set of conditions (such as only applicable in a particular state or mode or in response to a triggering event).

Therefore, a more-complete form for a functional/performance requirement is:

> *When <condition clause, the <entity> shall <action verb> <object> <response - measurable outcome> <qualifying clause>.*

There are several agreed types of condition clauses outlined in a range of methodologies. For example, Mavin (2009) suggests the following four types of condition clause in EARS:

Event-driven requirements:

> *When < trigger>, the <SOI> shall <system response>.*

State-driven requirements:

> *While <precondition)>, the <SOI> shall <system response>.*

Unwanted behavior requirements:

> *If < trigger>, then the <SOI> shall <system response>.*

Optional feature requirements:

> *Where <feature is included>, the <SOI> shall <system response>.*

Complex requirements:

When several conditions apply at the same time, such as the use of a trigger in an *event* block (when) that is only triggered when the SOI is into a given *state* defined in the *while* block, both conditions must be explicitly stated (*See rules R27 and R28*) in order for the need or requirement to be complete (C4).

Beyond the simplicity of EARS (*Easy Approach for Requirements Syntax)*, the most valuable takeaway that readers should gain from this approach is that many of the requirements, specially at sub-system and system element levels, are not ubiquitous, but are normally triggered by a given event, and only when the SOI is in a given state. Articulating requirements always as ubiquitous directly attacks the completeness of both the individual statement (C4) and of the set of requirements (C10).

Organizational Patterns Preferences

While ISO/IEC/IEEE 29148 shows the condition clause at the beginning of the statement and the qualifying clause at the end, some organizations prefer to put the qualifying clause at the beginning of the statement and the condition clause at the end. Either format is acceptable, providing that only one is chosen for a given set of needs or requirements. Organizations should define the desired pattern (R1) in their standards, guides, and processes and then ensure consistency in the pattern being used.

C.4 Building Blocks for Requirement Patterns

When it comes to defining the grammar of a requirement pattern, recursion has to be considered. In that sense, requirement patterns may be made up, if necessary, of smaller patterns (building blocks) that can also be further split in even smaller blocks.

This makes requirement patterns more modular and reusable, since different types of requirements can share some common building blocks, whilst also having unique elements.

For example, some special types of requirements may include (optionally) performance information. If the performance information is represented as a *second level* pattern, several advantages can be gathered:

- This second level pattern might be reused among different *top-level* patterns.

- Variations of the *second level* pattern can be defined yet extending the 'expressivity' of the entire catalog of patterns, with no need of changing the *top-level* ones.

Fortunately, the recursion referred in this appendix is not too deep; just a few levels should be enough to provide a set of flexible and reusable catalog of patterns. Usually, the complex patterns are formed by sub-patterns and the simple patterns are formed by simple restrictions, defining a set of building blocks. The way to construct these patterns catalog (bottom up, top down, iterative, etc.) is left to each organization based on their specific needs.

As an example, Figure C-1 shows several structures of possible sub-patterns to represent a *trigger* slot that, in turn, is used in several of the *top level* patterns:

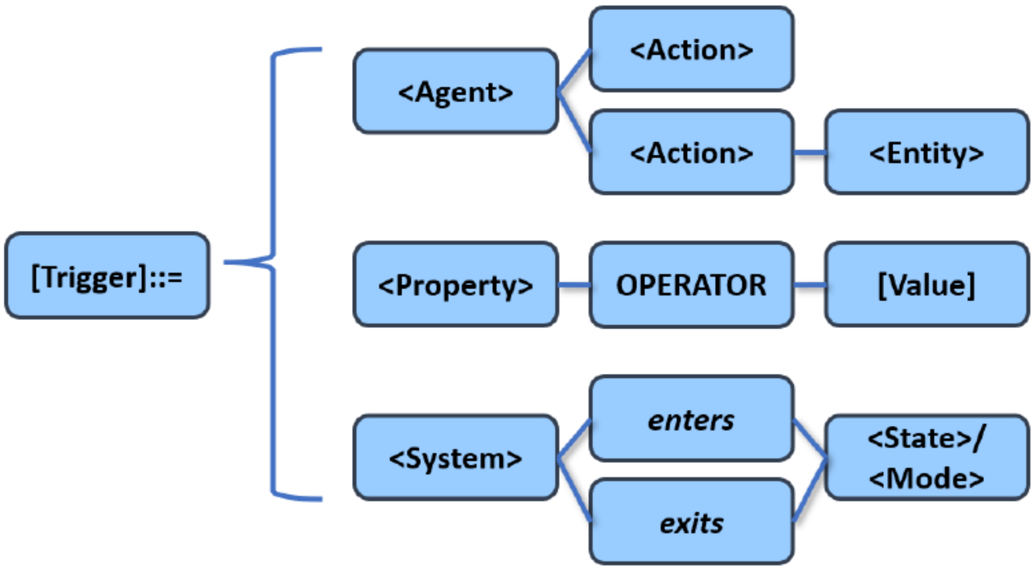


Figure C-1: Sub-patterns for a [Condition].

The notation used in Figure C-1 is as follows:

- Elements in square brackets represent patterns or sub-patterns: e.g. [Trigger], [Value].
- Elements in angle brackets represents the content from the data dictionary which is classified into different classes/clusters: for example, <Agent>, <Property>, <Entity>, <State> or <Mode>.
- Elements in italics represent fixed words or expressions: for example, *enters*, *exits.*
- Elements in capital letters represent different types of syntactic elements: for example, OPERATOR.

Similarly, other sub-patterns should be created to represent more accurately what a *system response* is, including the corresponding *performance information* when needed.

Following with the example above, there is already an element in the definition of the [Trigger] pattern which is not yet a terminal node/slot, this element is the [Value] and therefore, a final level of recursion shall be required. Figure C-2 represents an example of this final pattern:

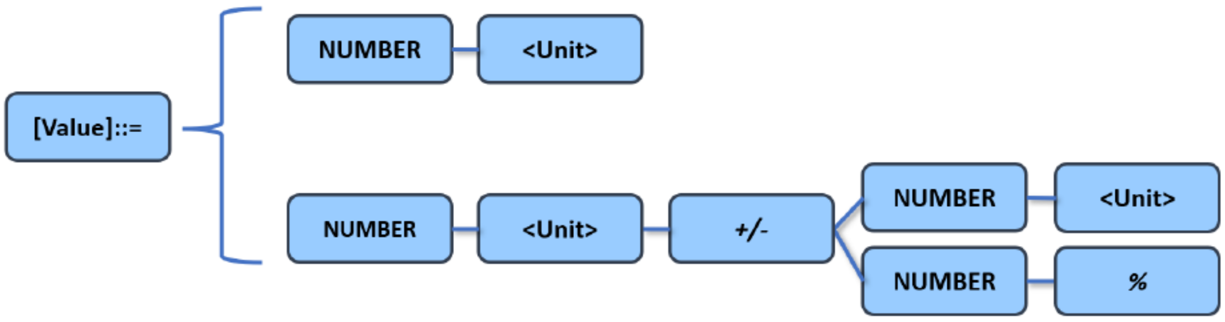


Figure C-2: Sub-patterns for a [Value].

As mentioned, there is no need to reach deeper levels of recursion, ending with a series of terminal pattern slots such as those described in Figure C-3. Those examples of terminal slots, each with a given semantics, shall be always included in the project data dictionary to ensure a consistent use of the vocabulary.

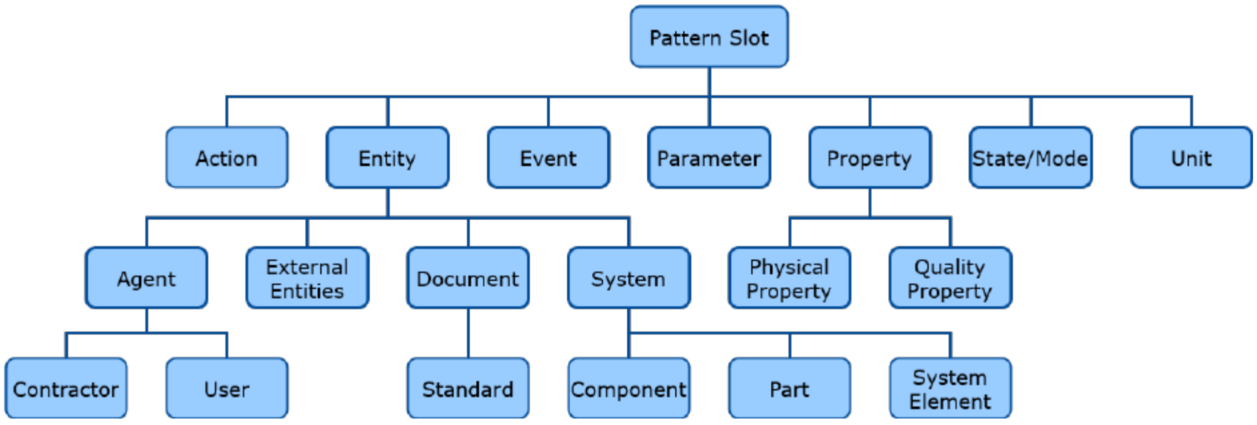


Figure C-3: Terminal building blocks.

Writing requirements based on agreed patterns is a good approach that will result in a consistent set of requirements. However, dealing with consistent patterns is largely pointless if the actual words and concepts used to specify a requirement are not used consistently. Every organization must solve this challenge within the scope of the knowledge management process. This common ontology must be implemented within the schema to be used to document and manage the requirement development and systems engineering process activities across the lifecycle.

Considering these more fine-grained components into a pattern, we could assign detailed patterns for different types of requirements. A necessary first choice for any organization is to clearly define a unique taxonomy of requirement types and their associated patterns (R1).

Examples of this finer decomposition are presented by Ron Carson (Carson, 2015). According to Carson, types of requirements will have different means of expressing the basic requirements elements (who, what, how well, under what conditions).

Possible types of requirements for a SOI may be related to the primary intended use (functional and performance requirements), fitness for use (suitability, e.g., safety, security, resilience and various "ilities"), constraints on the solution ("design" requirements), and natural and induced environments that define conditions of testing, operations, maintenance, transportation, or storage.

Boeing (Carson, 2015) defined a set of four basic requirements patterns addressing unique requirement types to enable quality measurements:

- **Functional/Performance**: The AGENT shall FUNCTION in accordance with INTERFACE-OUTPUT with PERFORMANCE [and TIMING upon EVENT TRIGGER in accordance with INTERFACE-INPUT] while in CONDITION.
- **Suitability**: The AGENT shall exhibit CHARACTERISTIC with PERFORMANCE while CONDITION [for CONDITION DURATION].
- **Environments**: The AGENT shall exhibit CHARACTERISTIC during/after exposure to ENVIRONMENT [for EXPOSURE DURATION].
- **Design**: The AGENT shall exhibit DESIGN CONSTRAINTS [in accordance with PERFORMANCE while in CONDITION].

Each of these types is intended to be both verifiable (capable of being proven satisfied) and traceable to the needs included within the Integrated Set of Needs discussed in Section 1. Each organization involved in defining requirements must establish patterns for different possible requirement types. These Boeing patterns can be further elaborated as follows:

The **Functional/Performance** type addresses the intended use-related functions, behavior, activity of the SOI. This type includes performance attributes to define "how well". If performance is omitted then there is no statement of "how well", and the resulting requirement is incomplete. The template allows an optional clause […] that enables a differentiation of behavior triggered by an "EVENT TRIGGER" with an associated TIMING performance attribute. The "INTERFACE" elements are associated with constraints on how the INPUTS and OUTPUTS to the function are implemented and can be defined by reference to an ICD or other interface constraint reference accessible to all entities involved in the interaction.

The **Suitability** type addresses intended use-related or enabling functions and characteristics associated with "fitness for use" in the intended environment, by intended personnel, and under intended conditions. This type includes such suitability considerations and functionality to meet quality (-ilities such as availability, reliability, maintainability), training, safety, security, and resiliency needs. The optional clause […] addresses the varying need to define the DURATION of the CONDITION being applied to the system, e.g., "reliability" is always defined with both a DURATION and CONDITION.

The **Environment** type is primarily a characterization of the CONDITIONS applicable to a system or its elements while either performing some subset of functions, or its existence in some state or phase. It is important to ensure that the CHARACTERISTIC is verifiably defined. For example, "The system shall survive exposure to xyz ENVIRONMENT" is not a verifiable requirement because "survive" is ill-defined and subject to widely varying interpretation as, "doesn't fail immediately", "doesn't diminish its reliability", "meet all functional and performance requirements", or "doesn't fall apart".

The **Design** type addresses constraints on the solution space. This type can be used to specify allocations of physical attributes, e.g., size, weight, and other physical characteristics, e.g., parts, materials, processes, color. The complete template includes optional […] clauses for Performance (measurable attributes) and CONDITIONS (e.g., "dry weight") because these are sometimes applicable, even for solution characteristics.

"**Interface requirements**" are not a separate type. Requirements associated with interfaces are either of the Functional/ Performance type addressing inputs and outputs (because this type can reference interface constraints), Suitability type, or of the Design type if used to constrain certain classes of interfaces, e.g., “The system shall use communicate digital data conforming to USB 3.0 standard.”

Organizations may define and use other types of requirements and their corresponding patterns. However, it is best to use a short list of types and have clear definitions (to avoid overlap and confusion) and clear related patterns so that completeness can be assessed. And, of course, each pattern must satisfy the basic construct of “the *who* shall *what, how well, under what conditions”* to help ensure completeness and verifiability.

C.5 Other references

Other sources of requirement patterns and boilerplates are:

- Jeremy Dick, Juan Llorens, "Using Statement-level Templates to Improve the Quality of Requirements", *International Conference on Software and Systems Engineering and Applications. ICSSEA 2012*, Paris, France.
- Hull et al: *Requirements Engineering*, Springer, 2012.
- “EARS - Easy Approach to Requirements Syntax”, http://ieeexplore.ieee.org/document/5636542/
- Sophist MASTER templates: https://www.sophist.de/
- The PABRE Catalog: http://www.upc.edu/gessi/PABRE/index.html
- ARTEMIS CRYSTAL EU Research Project: http://www.crystal-artemis.eu/
- Carson, R. S. (2015). Implementing Structured Requirements to Improve Requirements Quality. *Proceedings of INCOSE.* Seattle, WA USA: Wiley.

Appendix D: Rule Applicability Matrix

D.1 The Importance of Tailoring the Rules

The characteristics of needs and requirements presented in this Guide provide a common basis to support the definition of quality needs and requirements and sets of needs and requirements.

The underlying rules defined in this Guide help ensure that needs and requirements possess these characteristics. However, the SE practitioner applying the rules defined in this Guide to a specific project warrants additional guidance as to the selection of the appropriate set of rules when they are applied to need statements versus requirement statements.

In the aftermath of adopting NLP/AI based tools to partially automate the verification of needs and requirements to assess their quality, many SE practitioners have also been expecting from those "digital assistants" help in distinguishing need statements from requirement statements in the need verification and requirement verification activities rather than focusing only on requirements and sets of requirements as has been done in the past.

By including this Rule Applicability Matrix, the objective is to help SE practitioners tailor the rule set to be used to either assess the quality of need statements or requirement statements, providing them with guidance about optional or recommended rules applicable to needs before being applied to the requirements transformed from those needs. Users of NLP/AI based tools will thus benefit from a tailored automated assessment of the quality of need statements or requirement statements with the appropriate thoroughness.

The Rule Applicability Matrix also provides guidance about the need for a Project Data Dictionary, which can be organized either as a glossary, a model, or an ontology, among other kinds of data models. The concept of project data dictionary suggests that there is a data dictionary defined at organizational level, which enables SE practitioners to process those data inputs into engineered knowledge that will then be used across the lifecycle by the organizational project teams that may be using different tools. The data dictionary can include terms and definitions, but also relationships that help precisely define a consistent terminology to describe the SOI, its operating environment, and interactions with other external systems in the operational environment.

This data dictionary is then tailored to the specific project since the organization-wide terminology may need to be tailored within the scope of this project.

D.2 Dimensions of the Rule Applicability Matrix

As described in the paragraphs above, the *Rule Applicability Matrix* provides two dimensions to help engineers select the appropriate set of rules:

- The level of applicability of each rule to needs, requirements, or both.
- The need for a project data dictionary to check that a need or requirement statement complies with each rule and uses consistent terminology.

Level of Applicability

As shown in the following table, each rule is assigned a specific level of applicability:

- **CNCR:** Compulsory for Needs and Compulsory for Requirements,
- **RNCR:** Recommended for Needs and Compulsory for Requirements,
- **CR:** Compulsory for Requirements (Optional for Needs).

These levels suggest that the rules will be applied in any case for requirements, but every organization can decide whether to use a rule or not. The tailoring of the rules should be performed considering the following four considerations: the level of abstraction, the stage in the lifecycle, the level of criticality of the need or requirement, and the criticality of the SOI.

The difference between CN, RN and CR can be addressed as follows:
- 'CN' means the rule is compulsory for needs.
- 'RN' means the rule is recommended for needs but not compulsory. The project may decide not to apply the rule to the set of needs based on the above four tailoring considerations.
- 'CR' means the rule is compulsory for requirements but optional for needs depending on the above four tailoring considerations.

Scope (Project Data Dictionary)

Apart from the applicability, the table also shows if the rule requires a project data dictionary to be assessed. The rules that do not require a project data dictionary can be considered as "cross-project" or even "cross-domain", as they address the syntax of the need or requirement. Therefore, this criterion defines the scope on which the rule is to be applied.

Examples:
- Rule 05 (Use Definite Articles) does not require a project data dictionary, since the nature of the terms involved in the rules ("the", "a", an") do not vary from the organization-wide definition of articles.
- Rule 37 (Define Acronyms), however, requires a project data dictionary, as the acronym can be defined at project level, to characterize a specific feature, function, entity, component, unit of measure or even supersede an acronym used for another entity at organization level.

This scope criterion helps organizations prepare the tailoring of the rules to needs or requirements by defining the key terminology to be used within a project and serves as a set of best practices to be introduced within the organization prior to the definition of needs or requirements and other SE artifacts across the lifecycle.

Considering the importance of knowledge management in an organization across the lifecycle and enabling SE data and information to be captured, stored, and shared throughout the organization, helps set the cornerstone of the underlying processes and activities. The project data dictionary will improve the communication between various stakeholders and ensure the use of consistent terms and concepts across all lifecycle activities.

Applicability of Rules to Needs and Requirements

*(CNCR= Compulsory for Needs and Requirements; RNCR=Recommended for Needs and Compulsory for Requirements; CR=Compulsory for Requirements and Not Applicable to Needs)

Rule #	Rule Name	Rule Short Description	Rule Applicability*	Applicability Comments	Requires Project Data Dictionary, Glossary, Models or Ontology? (Y/N)	Scope Comments
R1	Structured Statements	Need statements and requirement statements must conform to one of the agreed patterns, thus resulting in a well-structured complete statement.	CNCR		N	
R2	Active Voice	Use the active voice in the need statement or requirement statement with the responsible entity clearly identified as the subject of the sentence.	RNCR	Needs: The use of active voice can have a higher tolerance when considering needs, although it is recommended to avoid it in both cases.	N	
R3	Appropriate Subject-Verb	Ensure the subject and verb of the need or requirement statement are appropriate to the entity to which the statement refers.	CNCR		Y	The subject should be the entity rather than a characteristic of the entity.
R4	Defined Terms	Define all terms used within the need statement and requirement statement within an associated glossary and/or data dictionary.	CNCR		Y	
R5	Definite Articles	Use definite article “the” rather than the indefinite article “a.”	RNCR	Needs: Some concepts with a higher level of abstraction might be expressed indefinitely in need statements as they are still not clearly defined. Also see R08 and R09.	N	
R6	Common Units of Measure	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.	CNCR		N	
R7	Vague Terms	Avoid the use of vague terms.	CNCR		N	

Rule #	Rule Name	Rule Short Description	Rule Applicability*	Applicability Comments	Requires Project Data Dictionary, Glossary, Models or Ontology? (Y/N)	Scope Comments
R8	Escape Clauses	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”.	RNCR	Needs: The lower level of precision of need statements might imply the use of no escape clauses. Also see R05 and R09.	N	
R9	Open-Ended Clauses	Avoid open-ended, non-specific clauses such as “including but not limited to”, “etc.” and “and so on”.	CNCR	Also see R05 and R08.	N	
R10	Superfluous Infinitives	Avoid the use of superfluous infinitives such as “to be designed to”, “to be able to”, “to be capable of”, “to enable”, “to allow”.	CR	Needs: The use of superfluous infinitives may be acceptable within a need statement given the higher level of abstraction.	N	
R11	Separate Clauses	Use a separate clause for each condition or qualification.	CNCR		N	
R12	Correct Grammar	Use correct grammar	CNCR		N	
R13	Correct Spelling	Use correct spelling	CNCR		Y	Project-specific vocabulary items can be flagged as misspelled words as they do not belong to any defined dictionary. Therefore, the spellchecking can be regarded as a project-specific aspect.
R14	Correct Punctuation	Use correct punctuation.	CNCR		N	

Rule #	Rule Name	Rule Short Description	Rule Applicability*	Applicability Comments	Requires Project Data Dictionary, Glossary, Models or Ontology? (Y/N)	Scope Comments
R15	Logical Expression s	Use a defined convention to express logical expressions such as “[X AND Y]”, “[X OR Y]”, [X XOR Y]”, “NOT [X OR Y]”.	CNCR		N	
R16	Use of “Not”	Avoid the use of the word “not.”	RNCR	Needs: Need statements might use negative expressions to state the unwanted behavior of the system to develop.	N	
R17	Use of Oblique Symbol	Avoid the use of the oblique ("/") symbol.	CNCR		N	
R18	Single Thought Sentence	Write a single sentence that contains a single thought conditioned and qualified by relevant sub-clauses.	CNCR		N	
R19	Combinator s	Avoid combinators that join clauses, such as “and”, ”or”, ”then”, ”unless”, ”but”, ”as well as”, ”but also”, ”however”, ”whether”, ”meanwhile”, ”whereas”, ”on the other hand”, and ”otherwise.”	CNCR		N	
R20	Purpose Phrases	Avoid phrases that indicate the “purpose of“, “intent of”, or “reason for” the need statement or requirement statement.	CNCR		N	
R21	Parenthese s	Avoid parentheses and brackets containing subordinate text.	CNCR		N	
R22	Enumeratio n	Enumerate sets explicitly instead of using a group noun to name the set.	RNCR	Needs: Early-stage needs might include group nouns to name a set of entities or functions that is appropriate to the level of abstraction being communicated.	N	
R23	Supporting Diagram, Model, or ICD	When a need or requirement is related to complex behavior, refer to the supporting diagram, model, or ICD.	RNCR	The supporting diagram model, or ICD helps define the context.	Y	
R24	Pronouns	Avoid the use of personal and indefinite pronouns.	CNCR		N	

Rule #	Rule Name	Rule Short Description	Rule Applicability*	Applicability Comments	Requires Project Data Dictionary, Glossary, Models or Ontology? (Y/N)	Scope Comments
R25	Headings	Avoid relying on headings to support explanation or understanding of the need or requirement.	CNCR		N	
R26	Absolutes	Avoid using unachievable absolutes.	CNCR		N	
R27	Explicit Conditions	State conditions' applicability explicitly instead of leaving applicability to be inferred from the context.	RNCR	Needs: 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.	Y	If a specific condition is applicable, it should be included within the need statement to ensure the intent clear
R28	Multiple Conditions	Express the propositional nature of a condition explicitly for a single action instead of giving lists of actions for a specific condition	CR	Needs: 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.	N	
R29	Classification	Classify needs and requirements according to the aspects of the problem or system it addresses.	CNCR		Y	The classification used for needs and requirements is defined at organization or project level. The classification schema is as such a project data dictionary.
R30	Unique Expression	Express each need and requirement once and only once.	CNCR		N	
R31	Solution free	Avoid stating implementation in a need statement or requirement statement unless there is rationale for constraining the design.	CNCR		N	
R32	Universal Qualification	Use "each" instead of "all", "any" or "both' when universal quantification is intended	RNCR	Needs: May be allowed in needs but not accepted in requirements	N	

Rule #	Rule Name	Rule Short Description	Rule Applicability*	Applicability Comments	Requires Project Data Dictionary, Glossary, Models or Ontology? (Y/N)	Scope Comments
R33	Range of Values	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.	CR	Requirements: Some needs can be expressed with "TBD" values (See R7), as the expected value for a characteristic is still unknown at need definition stage, while requirements require to have a proper definition of value ranges and tolerances to help its derivation into the lower levels.	N	
R34	Measurable Performanc e	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.	RNCR	Needs: Performance targets might be defined with a lower level of precision when expressing needs, without the ability to measure them.	N	
R35	Temporal Dependenc ies	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”.	RNCR	Needs: Temporal dependencies can be expressed with indefinite expressions at need definition stage and then precise in an accurate and non-ambiguous way as they are transformed into requirements.	N	
R36	Consistent Terms and Units	Ensure each term and unit of measure used throughout need and requirement sets as well as associated models and other SE artefacts developed across the lifecycle are consistent with the project’s defined ontology.	CNCR		Y	The definition of the terms and units of measure to be used requires a project data dictionary. See also R4 & R6.
R37	Acronyms	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.	CNCR		Y	

Rule #	Rule Name	Rule Short Description	Rule Applicability*	Applicability Comments	Requires Project Data Dictionary, Glossary, Models or Ontology? (Y/N)	Scope Comments
R38	Abbreviations	Avoid the use of abbreviations in needs and requirement statements as well as associated models and other SE lifecycle artefacts.	CNCR		N	
R39	Style Guide	Use a project-wide style guide for individual need statements and requirement statements.	CNCR		Y	
R40	Decimal Format	Use a consistent format and number of signification digits for the specification of decimal numbers.	CNCR		N	
R41	Related Needs and Requirements	Group related needs and requirements together.	CNCR		Y	See also Rule R29
R42	Structured Sets	Conform to a defined structure or template for organizing sets of needs and requirements.	CNCR		N	

Appendix E: Cross Reference Matrices

Rules to Characteristics Cross Reference Matrix

			Characteristics for Individual needs and requirements									Characteristics for Sets of needs requirements					
			Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be validated	Correct
Quality Focus	**Rule**	**Subject**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**
Accuracy	**R1**	Structured Statements			X	X			X	X	X						
	R2	Active Voice		X	X	X			X								
	R3	Appropriate Subject-Verb		X	X				X			X				X	
	R4	Defined Terms			X				X				X		X	X	X
	R5	Definite Articles			X				X								
	R6	Common Units of Measure			X	X			X	X							
	R7	Vague Terms			X	X			X								
	R8	Escape Clauses			X				X								
	R9	Open-ended Clauses			X	X	X		X								
Concision	**R10**	Superfluous infinitives			X				X								
	R11	Separate Clauses			X	X			X	X							
Non-ambiguity	**R12**	Correct Grammar			X				X	X	X						
	R13	Correct Spelling			X				X								
	R14	Correct Condition			X					X							
	R15	Logical Expressions			X				X								
	R16	Use of “Not”			X				X	X							
	R17	Use of Oblique Symbol			X				X								
Singularity	**R18**	Single-thought Sentence			X		X		X		X				X		
	R19	Combinators			X		X										
	R20	Purpose Phrases	X				X										
	R21	Parentheses					X										
	R22	Enumeration			X		X										
	R23	Supporting Diagram, Model or ICD			X	X	X										
Completeness	**R24**	Pronouns			X	X			X								
	R25	Headings				X											
Realism	**R26**	Absolutes						X	X	X				X			
Conditions	**R27**	Explicit Conditions				X			X	X							
	R28	Multiple Conditions			X				X								
Uniqueness	**R29**	Classification										X	X				
	R30	Unique Expression	X								X		X				
Abstraction	**R31**	Solution Free		X													
Quantifiers	**R32**	Universal Qualification			X				X	X							
Tolerance	**R33**	Range of Values			X	X		X	X	X				X			
Quantification	**R34**	Measurable Performance			X	X			X					X			
	R35	Temporal Dependencies			X	X			X								
Uniformity of Language	**R36**	Consistent Terms and Units			X					X	X		X		X	X	X
	R37	Acronyms			X						X		X		X	X	X
	R38	Abbreviations									X		X		X	X	X
	R39	Style Guide				X	X				X		X		X	X	X
	R40	Decimal Format			X	X					X		X				
Modularity	**R41**	Related Needs and Requirements				X					X	X	X		X		X
	R42	Structured Sets										X	X		X	X	X

NRM Concepts and Activities to Characteristics Cross Reference Matrix Part 1

		Characteristics for Individual needs and requirements									Characteristics for Sets of needs requirements					
		Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be validated	Correct
NRM Concepts and Activities		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
SECTION 3: INFORMATION-BASED NEEDS AND REQUIREMENT DEVELOPMENT AND MANAGEMENT																
3.2.1.1	Communication			X				X							X	
3.2.1.2	Power of Expression			X	X			X						X	X	
3.2.1.3	Managing Sets of Needs And Requirements				X									X		
3.2.1.5	Attributes	X												X		
3.2.1.6	Formal, Binding Agreement	X		X	X		X	X						X	X	
3.2.1.7	System Verification and System Validation							X							X	
3.2.2.1	Analysis from Which Needs and Requirements are Derived	X					X		X		X	X	X	X	X	X
3.2.2.2	Completeness										X			X	X	
3.2.2.3	Consistency											X		X	X	
3.2.2.4	Identity and Manage Interdependencies								X			X		X	X	X
3.2.2.5	Support Simulations							X						X	X	
3.2.2.6	Key to Understanding													X	X	
SECTION 4: LIFECYCLE CONCEPTS AND NEEDS DEFINITION																
4.3.3	Identify External and Internal Stakeholders										X					
4.3.6.2	Technology Maturity						X						X			
4.3.7.1	Classes of Risk - Development Risk						X						X			
4.4.3	Get Stakeholder Agreement	X		X	X			X	X		X	X		X	X	X
4.4.4	Completeness										X					
4.5	Lifecycle Concepts Analysis and Maturation	X			X		X	X	X			X				X
4.5.1	Feasibility						X						X			
4.5.3	User of Diagrams and Models for Analysis	X							X		X	X				X
4.5.4	Levels of Detail and Abstraction		X													
4.5.7.1	Model Development, Analysis, and Maturation	X							X		X	X				X
4.5.7.4	Zeroing in on a Feasible Architecture and Design						X						X			
4.6.2.3	Organizing the Integrated Set of Needs									X	X					
4.6.3.1	Managing Unknowns			X	X		X	X	X							X
4.6.3.2	Appropriate to Level		X													
4.6.3.3	Completeness of the Integrated Set of Needs										X					
4.6.3.4	Needs Feasibility and Risk	X	X				X						X			
4.7	Plan for System Validation														X	
4.8	Baseline & Manage Lifecycle Concepts & Needs Definition Outputs	X		X	X		X		X		X	X	X	X	X	X
SECTION 5: NEEDS VERIFICATION AND NEEDS VALIDATION																
5.1.2	Perform Needs Verification	X		X	X					X	X	X			X	
5.2	Needs Validation														X	
5.2.2	Perform Needs Validation			X			X		X		X		X	X	X	X

NRM Concepts and Activities to Characteristics Cross Reference Matrix Part 2

		Characteristics for Individual needs and requirements									Characteristics for Sets of needs requirements					
		Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be validated	Correct
NRM Concepts and Activities		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15
SECTION 6: DESIGN INPUT REQUIREMENTS DEFINITION																
6.2	Perform Design Input Requirements Definition	X	X					X	X		X	X	X	X	X	X
6.2.1	Transforming Needs into Design Input Requirements	X			X						X					
6.2.1.1	Organizing Sets of Design Input Requirements		X							X	X					
6.2.1.2	Considerations For Each Type Of Requirement				X			X	X		X					X
6.2.1.4	Appropriate to Level		X													
6.2.1.5	Managing Unknowns			X	X		X	X	X							X
6.2.2	Establish Traceability	X									X	X				
6.2.2.1	Establishing Traceability Between Dependent Peer Requirements											X				
6.2.3.6	Interface Requirements Audit	X			X			X	X		X	X			X	X
6.2.5	Plan for System Verification							X								
6.2.6.2	Completeness, Correctness, and Consistency								X		X	X				X
6.2.6.3	Requirements Feasibility and Risk	X	X				X						X			
6.3	Baseline and Manage Design Input Requirements	X		X	X		X		X		X	X	X	X	X	X
6.4.3	Allocation – Flow Down of Requirements		X								X	X				
6.4.4	Defining Child Requirements that Meet the Intent of the Allocated Parents										X					
6.4.5	Budgeting of Performance, Resource, and Quality Requirements										X	X				
6.4.7 .	Use of Traceability and Allocation to Manage Requirements	X							X		X	X			X	X
SECTION 7: DESIGN INPUT REQUIREMENTS VERIFICATION & VALIDATION																
7.1.2	Perform Design Input Requirements Verification	X		X	X			X		X	X	X			X	
7.2	Design Input Requirements Validation														X	
7.2.2	Perform Design Input Requirements Validation	X		X	X		X		X		X	X	X	X	X	X
SECTION 8: DESIGN VERIFICATION AND DESIGN VALIDATION																
8.1	Design Definition Process Overview			X	X		X	X	X		X	X	X	X	X	X
8.2	Early System Verification and System Validation			X	X		X	X	X		X	X	X	X	X	X
8.4	Design Verification			X	X		X	X	X			X				X
8.5	Design Validation										X	X	X	X	X	
SECTION 14: NEEDS, REQUIREMENTS, VERIFICATION, & VALIDATION MANAGEMENT																
14.2.1	Baseline Needs, Requirements, and Specifications	X		X	X		X		X		X	X	X	X	X	X
14.2.4	Managing Unknowns			X	X		X	X	X			X				X
14.2.7	Combine Allocation and Traceability to Manage Requirements	X							X		X				X	X
14.2.8	Managing Interfaces										X	X			X	
14.2.9	Managing System Verification and System Validation							X							X	

Attributes to Characteristics Cross Reference Matrix

		Characteristics for Individual needs and requirements									Characteristics for Sets of needs requirements					
		Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable/Validatiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be Validated	Correct
Attribute	**Subject**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**
A1	Rationale	X		X										X		
A2	Trace to Parent	X									X					
A3	Trace to Source	X														
A4	States and Modes										X					
A5	Allocation/Budgeting															
A6	System Verification or Validation Success Criteria			X	X		X	X	X							
A7	System Verification or Validation Strategy															
A8	System Verification or Validation Method			X	X		X	X	X							
A9	System Verification or Validation Responsible Organization															
A10	System Verification or Validation Level															
A11	System Verification or Validation Phase															
A12	Condition of Use				X				X							
A13	System Verification or Validation Results															
A14	System Verification or Validation Status															
A15	Unique Identifier															
A16	Unique Name															
A17	Originator/Author															
A18	Date Requirement Entered															
A19	Owner															
A20	Stakeholders															
A21	Change Board															
A22	Change Proposed															
A23	Version Number															
A24	Approval Date															
A25	Date of Last Change															
A26	Stability/Volatility						X						X			
A27	Responsible Person															
A28	Need or Requirement Verification Status															
A29	Need or Requirement Validation Status															
A30	Status of the Need or Requirement															
A31	Status (of Implementation)												X			
A32	Trace to Interface Definition	X			X			X				X				
A33	Trace to Dependent Peer Requirements	X										X				
A34	Priority	X														
A35	Criticality or Essentiality	X														
A36	Risk (of Implementation)						X						X			
A37	Risk (Mitigation)															
A38	Key Driving Need or Requirement (KDN/KDR)						X						X			
A39	Additional Comments			X												
A40	Type/Category										X					
A41	Applicability															
A42	Region															
A43	Country															
A44	State/Province															
A45	Market Segment															
A46	Business Unit															
A47	Product Line															
A48	Product Line Common Needs and Requirements															
A49	Product Line Variant Needs and Requirements															

Refer to the Needs and Requirements Manual Section 15 for a detailed description of each attribute.

Appendix F: Comment Form

Reviewed Document:	
Name of submitter (first name & last name):	
Date Submitted:	
Contact information (email address):	
Type of submission (individual/group):	
Group name and number of contributors (if applicable)	***Please read examples carefully before providing your comments (and delete the examples provided.)***

Comment sequence number	Commenter name	Section number (for example, 2.1.1, no alpha)	Specific reference (for example, paragraph, line, Figure no., Table no.)	Issue, comment and rationale *(Rationale must make comment clearly evident and supportable)*	Proposed Changed/New Text -- MANDATORY *ENTRY* -- *(Must be substantial to increase the odds of acceptance)*	Importance Rating (R = Required, I = Important, T = Think About for future version)

Submit comments to Working Group chair. Current WG chair will be listed at:

http://www.incose.org/techcomm.html

If this fails, comments may be sent to info@incose.org (the INCOSE main office), which can relay to the appropriate WG, if so, requested in the comment cover page.

(INTENTIONALLY BLANK)

(BACK COVER)

INCOSE Publications Office
7670 Opportunity Road, Suite 220
San Diego, CA 92111-2222 USA

ISBN: 978-1-93707-05-4