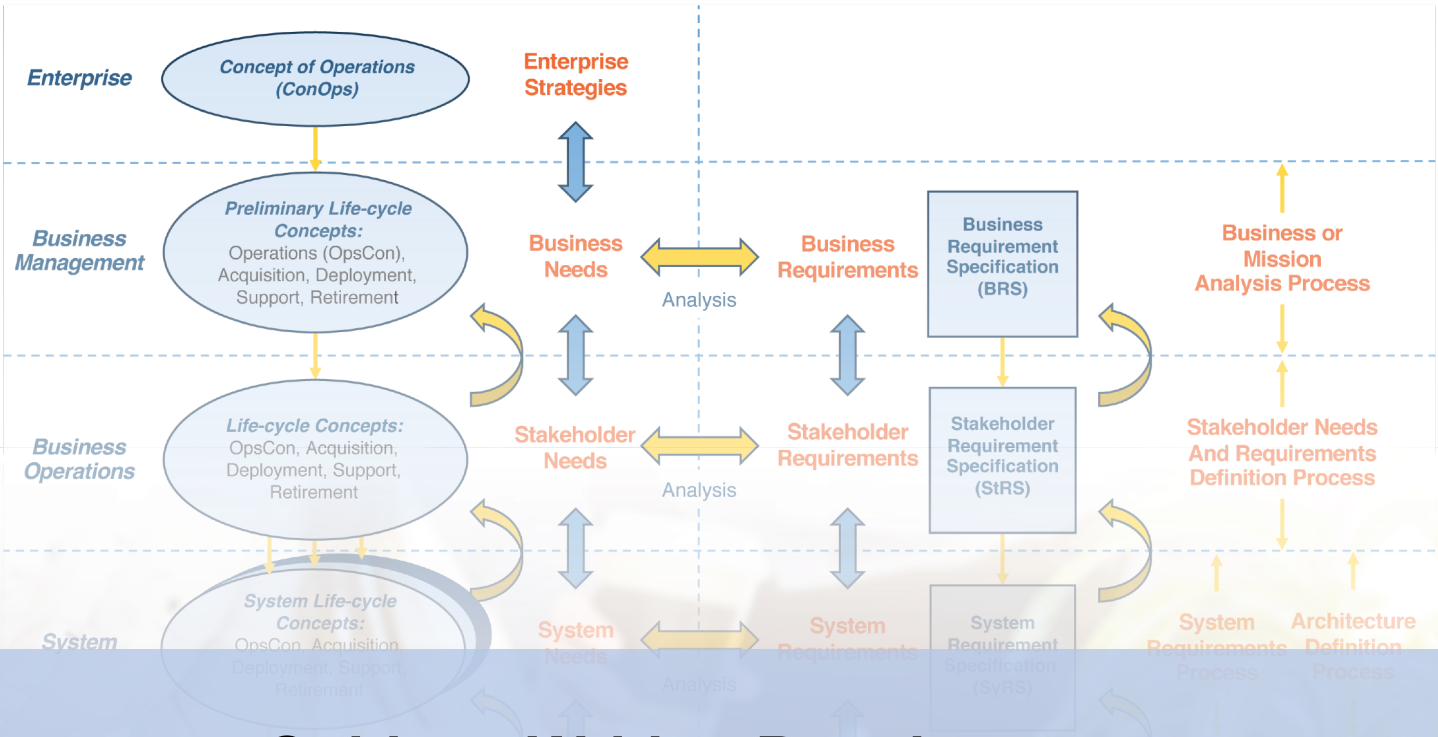
INCOSE



Guide to Writing Requirements

May 2022

INCOSE-TP-2010-006-04 | VERS/REV: 4 | 1 April 2022

INCOSE

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-2010-006-03.1
Version/Revision: Rev 3.1
Date: April 2022
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 after a number of revisions was published as an INCOSE Technical Product in July 2015. Subsequently, the document has been revised extensively following reviews by the RWG led by Michael Ryan, Lou Wheatcraft, Kathy Baksa, and Rick Zinni at annual International Workshops. The latest revision accommodates movement of a substantial portion of introductory material to the Needs and Requirements Manual (NRM) recently prepared by the RWG as well as an alignment between this Guide and the NRM and Guide to Needs and Requirements (GtNR).

Authors

The principal authors of this Guide are:

Michael Ryan, Capability Associates Pty Ltd, Canberra, Australia
Lou Wheatcraft, Wheatland Consulting, LLC, USA
Rick Zinni, Harris Corporation, USA
Jeremy Dick, Retired, UK
Kathy Baksa, Pratt & Whitney, USA

Major Contributors

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

Ronald S. Carson, Seattle Pacific University
Jose L. Fernandez, Madrid Technical University, Spain
José Fuentes, The Reuse Company, Spain
Juan Llorens, Universidad Carlos III de Madrid, Spain
Robert A. Noel, The Boeing Company
Gina R. Smith, Systems Engineering Global Inc, USA
Christopher Unger, GE Healthcare, USA

Reviewers

The following reviewers submitted comments that were included in the development of this Guide:

Sascha Ackva, Continental AG, Germany
Luis Alonso, The Reuse Company, Spain
David Bar-On, Motorola Solutions, USA
Paul DeSantis, GHSP
Charles Dickerson, Loughborough University, UK
Jim van Gaasbeek, Retired, USA
Elena Gallego, The Reuse Company, Spain
Cecilia Haskins, Norwegian University of Science and Technology, Norway
Tim Kerby, Edinburgh Systems Ltd
Joel Knapp, National Aeronautics and Space Administration, USA
Tami Katz, Ball Aerospace, USA
Ron Kohl, R. J. Kohl & Assoc, USA
Mario Kossmann, Airbus, United Kingdom
Avishek Kumar
Phyllis Larson, Bechman Coulter, USA
Robert Luff, Loughborough University, UK
Glenn Thibodeau, Autoliv-Nissin Brake Systems
Andreas Vollerthun, KAIAO-Consulting, Germany
Arie Wessels, INCOSE South Africa

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 characterizes 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 the reviews by RWG members
2	1 Jul 2015	Release by INCOSE Tech Ops
2.1	30 Jun 2017	Updates resulting from the reviews by RWG members at INCOSE IW2016 & IW2017
2.2	10 Feb 2019	Updates resulting from the reviews 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

TABLE OF CONTENTS

LIST OF FIGURES

Section 1: Introduction

1.1 Purpose and Scope

This guide is specifically about how to express need and requirement statements in textual form in the context of systems engineering. The aim is to draw together advice from existing standards such as ISO/IEC/IEEE 29148 and 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 structure, natural language.

The goals and objectives of this guide are to:

- Provide practical, cross domain guidance, with examples that will enable project teams to define well-formed need and requirement statements.
- Provide a definitive source on writing well-formed needs and requirements that organizations can reference when developing and documenting a set of rules, guidelines, processes, and templates that are tailored to their specific organizational needs, unique culture, domain, and product line.
- Define a set of characteristics that well-formed needs, requirements, sets of needs, and sets of requirements should have.
- Provide a set of rules for writing needs and requirements that, if followed, will result in the relevant statements having the desired characteristics.
- Addresses the concept of a boilerplate, template, or pattern for a requirement statement.

This guide is not about the discovery, capture, or elicitation of requirements; nor is it about requirements analysis, requirements management, and the development of models or design outputs, even though requirements play a key role in all those activities and processes—those are discussed in the Needs, Requirements, Manual (NRM) and the Guide to Needs and Requirements (GtNR). The focus of this guide is on how to express need statements (needs) and requirement statements (requirements) clearly and precisely as text using a structured, natural language in a form that supports further analysis and implementation, independent of any systems engineering (SE) tool used to capture and manage those needs and requirements throughout the system lifecycle.

Although the focus of the guide is on writing textual needs and requirements, it does not, and should not, incorporate all the rules of good grammar and writing style. Rather, the guidance given herein assumes the writer has applied proper grammar and style—for example, applied proper punctuation, sentence structure, avoided double negatives, and appropriately located modifying clauses.

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. Fundamental to forming shareable sets of data, the project must define and document a project ontology. The ontology includes the formal naming and definition of a set of terms, entities, data types, and properties as well as defining the relationships between these terms, entities, and data types that are fundamental to the project and organization of which the project is part. Having a documented project ontology 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 project ontology is key to interoperability and the ability to share sets of data and information, including needs and requirements.

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, the apply to the business requirements result 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*.

1.2 Audience

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

Major user groups include systems engineers, requirements engineers, business analysts, product developers. system architects, configuration managers, designers, testers, verifiers, validators, manufactures, coders, operators, users, disposers, course developers, trainers, tool suppliers, and project managers.

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

Although reading, reviewing, and commenting on need and requirement statements requires less skill than writing them, it is helpful to understand why needs and requirements are expressed in the way they are. Understanding the characteristics of well-formed needs and sets of needs, requirements and sets of requirements, and the rules that need to be followed that result in having these characteristics will help both the authors as well as reviewers, developers, verifiers, and validators to identify defects, that if not corrected, can result in costly rework, schedule slips, and needs not being realized.

This Guide is also addressed to tool vendors that are applying Artificial Intelligence (AI) and Natural Language Processing (NLP) to the evaluation of needs and requirement statements with the goal of improving their quality using the rules defined in this Guide.

Given the number of rules defined, it can be overwhelming to apply them all. These tools serve as a "digital assistant" that will help systems engineers to craft higher quality, well-formed requirements which is the ultimate goal of this Guide.

1.3 Approach

This guide presents underlying characteristics of need and requirement statements, practical rules for writing need and requirement statements, and patterns that can be followed when writing need and requirement statements.

All need statements, sets of needs, requirement statements, and requirement sets must have the characteristics expressed in this guide. The rules included herein provide guidance on how to achieve these characteristics. Focusing on characteristics is important because the set of rules alone can never be a perfect and complete expression of how to achieve the goal of well-formed needs and requirements. The diverse nature of needs and requirements means that the rules need to be adapted to specific cases. Given this reality, an understanding of the underlying characteristics and reasons for those characteristics informs the adaptation of the rules to meet the needs of an organization.

Pre-defined patterns for needs and requirement statements provide a template or boilerplate for various types of needs and requirements helping to ensure the need and requirement statements have the defined characteristics.

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

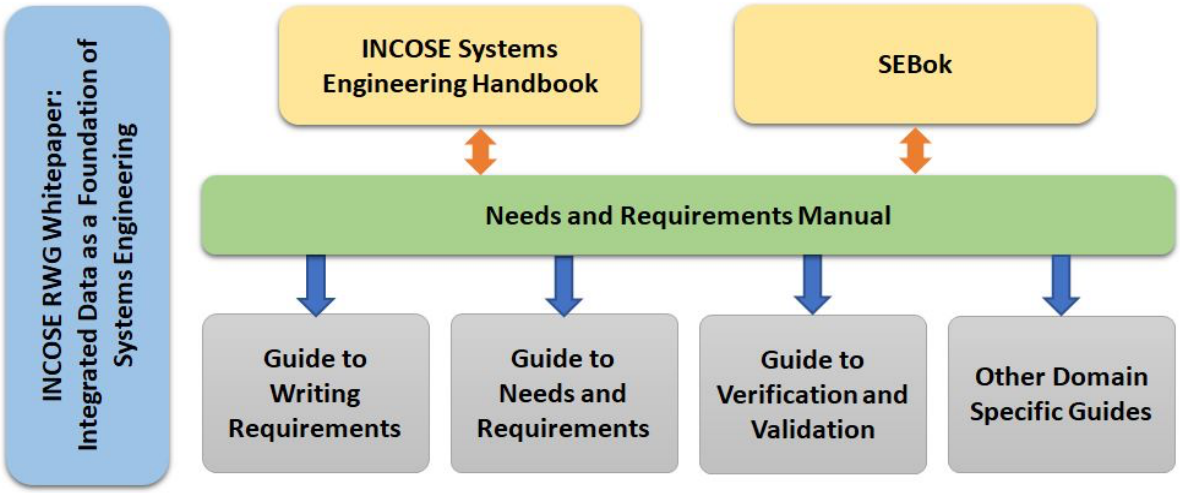


Figure 1: Relationships Among RWG Products.

It is recommended the reader have familiarity with the underlining concepts within the NRM as well as the related guides: Guide to Needs and Requirements (GtNR) and Guide to Verification and Validation (GtVV). The quality of need and requirement expressions is highly dependent on the successful implementation of the Needs and

Requirements Definition and Management (NRDM) activities. Successful implementation of the system verification and validation activities described in the NRM and GtVV is highly dependent on the quality of the needs and requirements resulting from following the guidance contained in this Guide as well as the activities defined within the NRM and GtNR.

Figure 2 shows the context of this Guide where the primary focus is on 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 a set of design input requirements for the SOI. Together, the integrated set of needs and resulting set of design input requirements are considered inputs into the INCOSE SE HB *Architecture Definition* and *Design Definition* processes which transform the design input requirements into sets of design output specifications to which the SOI is realized.

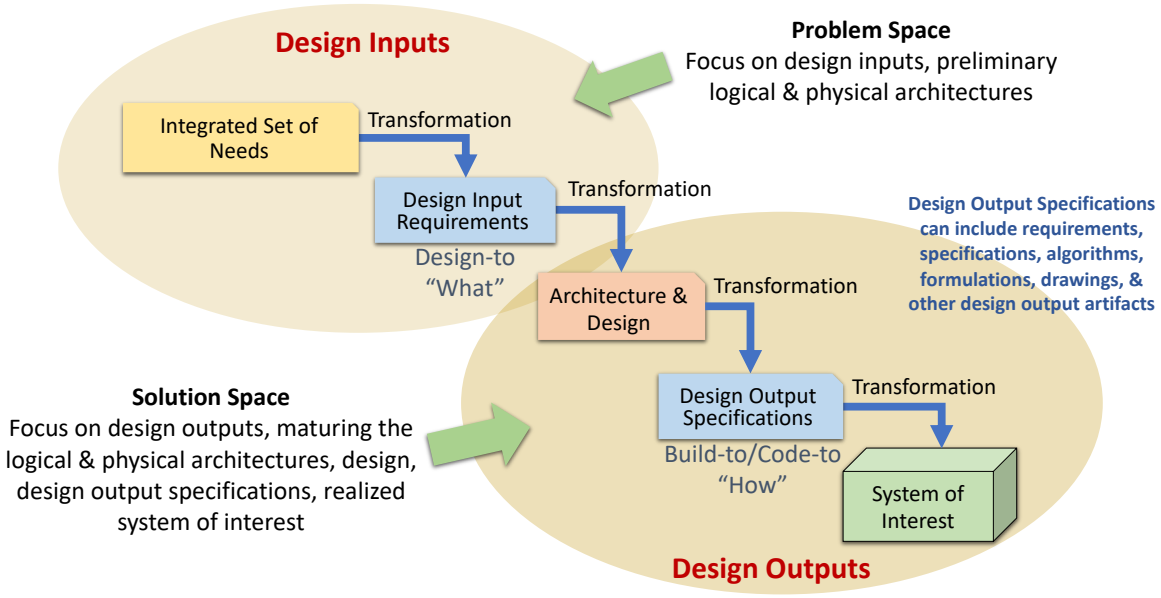


Figure 2: Needs and Requirements in Context.

1.4 Why use the Textual Form of Communication?

For many ideas and concepts that need to be communicated; well-formed, textual need and requirement statements have been proven to be the most effective form of communication that covers the wide variety of concepts that must be communicated throughout a system lifecycle.

A major critique of textual needs and requirement statements is the inherent ambiguity in the use of an unstructured, natural language. English has many synonyms and words with slightly different shades of meaning based on context. Because of this, it can be difficult to be clear, precise, and to avoid ambiguity.

To help avoid this ambiguity, the language advocated in this Guide is a structured, natural language with a set of rules to reduce ambiguity. When the phrase "well-formed" is applied to textual need and requirement statements, it refers to need and

requirement statements that are written using this structured, natural language following the rules stated 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 Scaled Agile Framework (SAFe®));
- Prototypes, such as used in production-driven and rapid application development methodologies; 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, Sections 3 and 4 for a detailed discussion concerning the complementary use of these methods to discover, analyze, and express needs and requirements.

Just as there are issues with any form of technical communication, these other approaches can also be imperfect as they do not yet cover the wide range of concepts needed and have their own presentational, traceability, and management challenges.

- Problem statements, operational scenarios, use cases and user stories are written from the perspective of the user's (actor's) interaction with other actors and the system under development rather than 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, as defined by the use cases. While use cases are an excellent conceptual tool for stakeholder expectation analysis to help understand the features and associated functionality and performance expected by the stakeholders of the SOI, they do not always effectively replace well-formed, text-based stakeholder needs and requirements for all the various ideas and concepts that must be communicated, especially non-functional needs and requirements.
- As an alternate form to communicate stakeholder needs and requirements, use cases, diagrams, and other model forms do not have the characteristics of well-formed need and requirement statements defined in this Guide that are necessary to clearly communicate the broad spectrum of needs and requirements into a language that can be clearly understood by all parties (stakeholders, developers, testers) over time.

The reality is that the textual form of needs and requirements are not only useful, but they are also necessary. Operational scenarios, use cases, diagrams, and other types of models are also useful and necessary. Each of these forms represents a specific visualization and need to be represented in an integrated data and information model of the SOI. With this approach, the underlying data and information model captures not only the needs and requirements but also their relationships to each other and to other artifacts across the lifecycle represented within the data and information model. This enables stakeholders to view the data and information in whichever form is best for

what they are trying to communicate or achieve. Therefore, it is important to understand which form and media is best to communicate specific ideas and concepts to a given audience over time.

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. Advantages include:

- ***Communication***. There is still (arguably, there will always be) a sizable audience who cannot interpret, 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 words used (technical jargon) are not intuitively obvious to the reader. These people, particularly management, customers, and users of the system or other non-technical personnel, may not have been trained in language-based models or find the terminology used in some diagrams and models confusing and not intuitive.

  When that is the case, effective communication (See NRM, Section 2.10) has not taken place since the receiver of the message may not interpret and understand the message as intended by the sender and may well lose interest in the needs and requirements definition activities. On the other hand, text is universal. Of course, the textual statements must be well-written in such a way as to be clear, correct, and unambiguous; but then diagrams and models have even more potential to be unclear, incorrect, and ambiguous if they are poorly formed, defective, or the wrong meaning is assigned to them. Diagrammatic or model representation will have to be supported by well-written, detailed textual statements and descriptions for the representations to be understood unambiguously by all stakeholders. *This capability supports the characteristics: C3 - Unambiguous, C7 - Verifiable, C13 - Comprehensible, C14 - Able to be Validated.*

- ***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 may be capable of expressing functional, performance, and interface needs and requirements, but 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 carry the universal power of expression using a structured, natural language for all types of needs and requirements. *This capability supports the characteristics: C3 - Unambiguous, C4 - Complete, C7 - Verifiable, C13 -Comprehensible, C14 - Able to be Validated*.

- **Managing Sets of Needs and Requirements.** There is a significant barrier to defining and managing needs and requirements as part of language-based models: they do not lend themselves to the presentation of large numbers of needs and requirements. SysML requirement diagrams can present individual requirement expressions but are not well suited to representing multiple or large sets of requirements. This limitation of the SysML requirement diagram as a 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 "needs", nor corresponding needs diagrams. This capability supports the characteristics: C4 - Complete, C13 - Comprehensible.

- ***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, the SE tools (including RMTs) and modeling tools used to create and view the data and information represented by the model's dataset are not readily available and assessable to all stakeholders. In many cases, access is restricted by the number of "seats" or "licenses" purchased. Being able to provide needs and requirements in an electronic document format (pdf, or common office application formats) allows the stakeholders to view the needs and requirements in applications that have been installed on their computers. In addition, there are still managers who still prefer, and demand, printed, text-based documents, and will continue to do so for the foreseeable future.
- ***Attributes***. Both the need and requirement expressions include attributes that can be used to manage them as well as the system under development across all lifecycle process activities. While modeling languages allow users to define an entity having the name "attribute" and link that entity to a need or requirement statements, few practitioners do so, especially when there are multiple attributes that the project team has decided to use. Operational scenarios, use cases, user stories, and other diagrams used to represent needs or requirements are not conducive to appending a large set of attributes nor using those attributes to produce meaningful reports and dashboards for use by management stakeholders. *Refer to the NRM, Section 15 for a more detailed discussion concerning the use of attributes. This capability supports the characteristics: C1 - Necessary and C13 - Comprehensible*.
- ***Formal, binding agreement***. Textual needs and requirement statements are more easily understood in a formal agreement or contract-based system development effort by a wider, and often, non-technical set of stakeholders including business management, project management (PM), configuration management (CM), contract administrators, and legal support. 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 will be verified to meet the requirements or validated to have met the needs. 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. *This capability supports the characteristics: C1 - Necessary, C3 - Unambiguous, C4 - Complete, C6 - Feasible, C7 - Verifiable, C13 - Comprehensible, C14 - Able to be Validated*.
- ***System verification and system validation***. Most formal agreement (contract-based) product development and management processes as well as highly regulated products include system verification and system validation as formal

processes that must occur prior to product acceptance, qualification, certification, and approval for use. In highly regulated, safety critical industries such as the medical device industry, aviation, and consumer products, formal evidence that the design outputs, including the product, meet the design inputs (needs and design input requirements) is needed prior to the product being approved for release into the market. Currently, no other form, other than textual requirements, has been able to meet these characteristics. For a detailed discussion on system verification and system validation refer to the NRM and GtVV. *This capability supports the characteristics: C7 - Verifiable, C14 - Able to be Validated*.

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. 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.

This guide refers solely to the expression of textual needs and requirements. If an organization chooses to use alternate forms to define needs and requirements, the characteristics defined in this guide are still applicable. If the alternate form does not have these characteristics, there is a risk that the needs and requirements of those for whom the system is being developed will not be met.

1.5 Definitions

This section defines several fundamental terms, based on definitions provided by ISO/IEC/IEEE 29148 and ISO/IEC/IEEE 15288, in “An Improved Taxonomy for Definitions Associated with a Requirement Expression” (Ryan, Wheatcraft, Dick and Zinni, (2014)) and the NRM.

When describing system development, some form of distinction is commonly made between lifecycle concepts, needs, and requirements as shown in Figure 3.

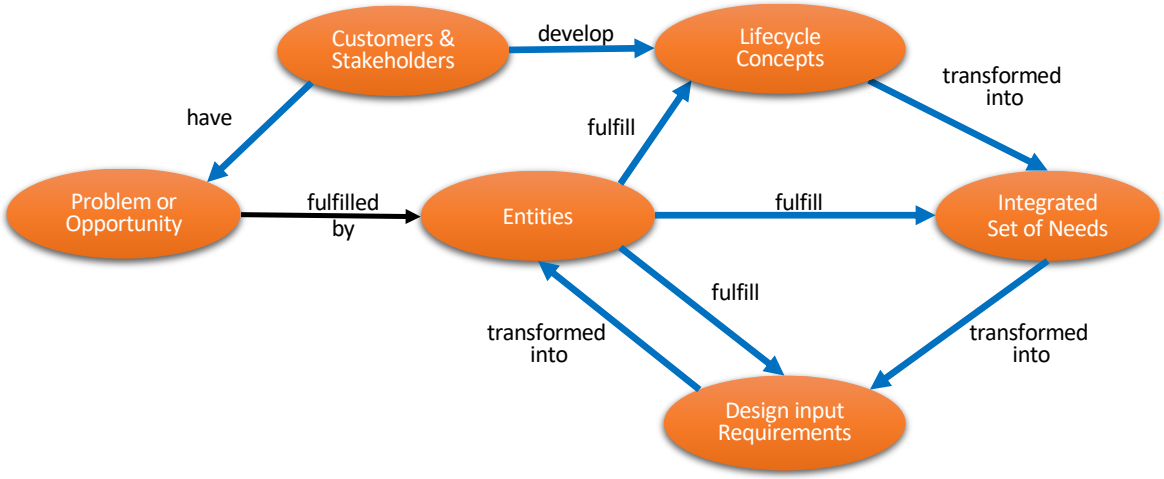


Figure 3: Entity-relationship Diagram for Customers, Concepts, Entities, Needs and Requirements Terms (based on Ryan and Wheatcraft, 2017).

Note: In these definitions the term “customer” is used to refer to the organizations or persons (internal or external) 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 are multiple customers.

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.)*.

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 capability in terms of people, process, and products.

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

The information used to define the lifecycle concepts includes a problem statement, a mission statement, goals, objectives, measures, stakeholder needs and stakeholder-owned system requirements, use cases, user scenarios, user stories, operational scenarios, drivers and constraints, and risks. Using this information, through formal lifecycle analysis and maturation, a set of lifecycle concepts is defined for an entity.

Lifecycle concepts can be communicated in various forms including textual (e.g., Operational Concept (OpsCon) or Concept of Operations (ConOps)), graphical representations (e.g., diagrams and models), and/or electronic (data bases). There are multiple lifecycle concepts that apply to each entity, so it is useful to develop a necessary and sufficient set of lifecycle concepts which define the needs and requirements for that entity.

Based on the set of lifecycle concepts, through formal needs analysis, an integrated set of needs is defined, which will result in the set of lifecycle concepts for the entity to be realized.

Needs are formal textual statements of expectations for an entity stated in a structured, natural language from the perspective of what the stakeholders need a SOI to do, 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 lifecycle concepts into an agreed-to expectation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.

As with defining needs, defining requirements is an activity which, through formal requirements analysis, determines specifically what the entity must do to meet the needs they are being transformed from using a formal transformation process involving decomposition, derivation, diagrams, and architectural and analytical/behavioral models. A deeper exploration and elaboration of the lifecycle concepts, including a thorough examination of interactions between the entity and other entities, are part of the transformation from needs into requirements. As a result of the requirements analysis, there may be more than one requirement defined for each need.

Requirements are formal 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 needs or parent requirements into an agreed-to obligation for an entity to perform some function or possess some quality within specified constraints with acceptable risk.

The analysis used to transform lifecycle concepts into needs and to transform needs into requirements is frequently referred to as business analysis or mission analysis at the enterprise and strategic levels of the organization and needs analysis and requirements analysis at the business operations, system, subsystem, and system element levels. Diagrams and models are important tools used as part of this analysis to help achieve correctness, consistency, completeness, and feasibility of the transformations.

Lifecycle concepts, needs, and requirements should be developed for entities at all levels.

The reader is urged to be diligent with the lifecycle concepts-needs-requirements (CNR) approach advocated in the NRM and GtNR and resist the temptation to jump from the lifecycle concepts directly to writing requirements or even worse, going directly from lifecycle concepts to candidate design solutions, skipping needs and requirements definition entirely.

Both need and requirement statements are more than the well-formed textual statements which are written succinctly in a standard format using a structured, natural language having the characteristics defined in this Guide. As illustrated in Figure 4, the full need or requirement expression includes associated attributes that aid in the definition and management of a need, sets of needs, requirements, and sets of requirements. These attributes aid in achieving many of the characteristics defined in this Guide. *Refer to the NRM, Section 15 for a more detailed discussion on attributes as applied to needs and requirements.*

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

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.

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

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

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

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

Figure 4 shows an Entity-Relationship Diagram that shows how the above terms related to each other.

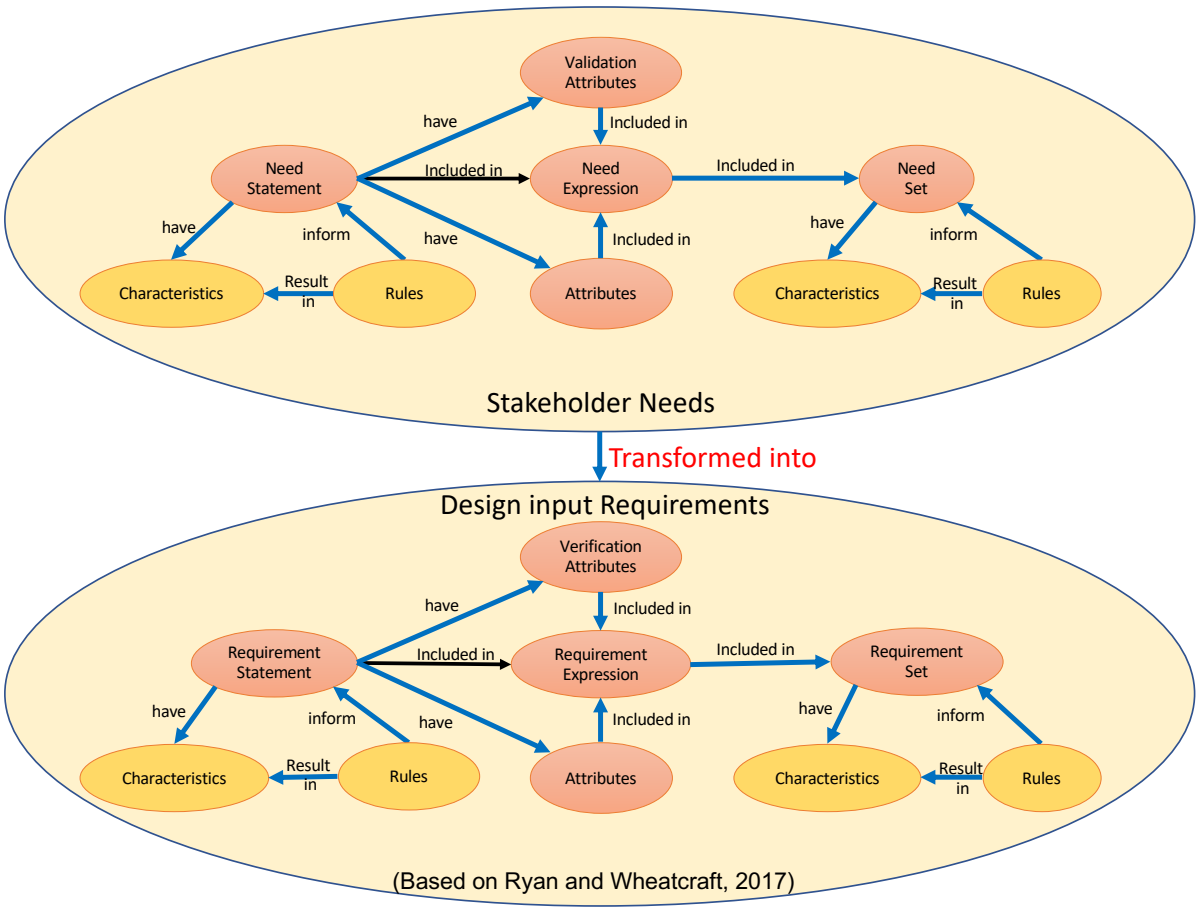


Figure 4: Entity-Relationship Diagram for Needs and Requirements Terms.

1.6 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 the above definitions.

In terms of need and requirement statements, the entity that is the object of a need statement (“The <stakeholders> need the <entity> to .....”) will be the subject of a subsequent requirement statement (“The <entity> shall .....”) transformed from that need. (Stakeholders are entities as well.)

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

The project shall …….
The [xxxxx team] shall…….

There will be *supplier 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 supplier shall …….
The contractor shall …….

As discussed in the NRM, Section 10, there will be *procedural requirements* concerning actions the person or organization responsible for conducting steps within a procedure resulting in those actions, e.g., a system verification or system validation procedure or a test procedure, such as the following:

> 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 *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 provides expectations concerning the production of a SOI:

> 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).

Systems engineers must make clear the applicability of needs and requirements based on the entity they apply and not mix them together within a single set. Each need and requirement must be recorded in separate sets for the entity to which they apply. This is important because for each entity there is an expectation that objective evidence can be obtained which can be used to prove, with some level of confidence, that the entity has met the requirements (system verification) or needs (system validation) of that entity. See also Rule R3.

1.7 Needs vs Requirements

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, e.g., Users Requirements Document, Program Requirements Document or Stakeholder Requirements Document, communicating the needs and requirements in the form of a requirement using a “shall” statement. This can result in confusion as to their form, in terms of what an is “expectation” versus what is a “need” as opposed to what is a “requirement”.

For some practitioners, “stakeholder expectations” are communicated in terms of stakeholder needs and requirements; ISO/IEC/IEEE 29148 [25] defines “stakeholder requirements” in terms of “stakeholder needs”. This Guide defines “needs” in terms of “expectations”. For others, stakeholder needs and expectations are communicated within a use case, user story, or an operational concept. 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 requirements” designation.

As well as stakeholder requirements”, ISO/IEC/IEEE 29148 also includes the phrase “stakeholder-owned system requirements”, although it is not clear whether what is meant by “stakeholder requirements” are really “stakeholder-owned system

requirements", i.e., stakeholder requirements for a system that result in the stakeholder needs being met. In this context "customer requirements" could also be interpreted as "customer-owned system requirements".

To avoid ambiguity, when writing needs and requirements the reader is cautioned to be aware of the differences and the role each has in the lifecycle as discussed below. Whenever the phrase "stakeholder or customer requirements" is used, the reader should assume what is meant is "stakeholder-owned or customer-owned system requirements".

This distinction is important in that "stakeholder, user, or customer needs" represent a stakeholder, user, or customer perspective of what they need the SOI to do as viewed externally, while the "stakeholder-owned", "user-owned", or "customer-owned" system requirements are a technical perspective that communicates what the stakeholders, users, or customers require the SOI to do to meet their needs. In this context, the stakeholder-owned, user-owned, or customer-owned system requirements are transformed from the stakeholder, user, and customer needs. Often, the requirements from the customer included in a contract with a supplier are "customer-owned system requirements". Below is further elaboration on the differences between needs and requirements.

Needs are NOT Requirements

Needs are written from the perspective of the what the stakeholders need the SOI to do, while the requirements 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.

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

One approach to avoiding this confusion is to use the format:

> "The stakeholders need the system to ……." or for a goal "The stakeholders would like the system to ……".

These statements are in contrast to the form of a requirements statement

> "The SOI "shall" …" or for a goal, "The SOI "should" ……"

Using these distinct formats helps make clear what is a need statement and what is a requirement statement. For example, a stakeholder need:

> "The stakeholders need the system to comply with OSHA safety standards".

This need states the stakeholder expectation for the system concerning safety. The requirements for the system transformed from this need statement would focus on the specific requirements the system must meet for the intent of this need to be realized. These requirement statements use the word "shall" to make clear they are

requirements. (Refer to the GtNR and NRM for guidance on writing well-formed requirements that address requirements within standards and regulations.)

A common issue when recording needs, is when the need statements are written as "shall" statements and calling these "stakeholder requirements" rather than needs. For example:

"The system shall meet government safety standards and regulations."

While the level of abstraction may be appropriate for a need, it is not appropriate for a requirement because it is ambiguous as to which standards and regulations. The result is often "stakeholder requirements" that do not have the characteristics of well-formed requirements as defined in this Guide.

Another example 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 system to allow users to [do something]".

A common error is to rewrite the need statement as a requirement:

"The user shall be able to [do something]".

This is not an appropriate form for a requirement. The project team would need to do an engineering analysis to determine what the system must to do such that users will have the capability to do that action. Based on that analysis, the project team would then write one or more requirements that when implemented by the design would result in the user being able to do that action as stated in the need statement.

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 set of design input requirements.

1.8 Quality of Needs and Requirements

When defining needs and requirements, it is important that they have the characteristics of well-formed needs and requirements as defined in this Guide. These characteristics are a result of following the rules defined in this Guide as well as performing the activities associated with the definition of the needs and requirements 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. "Writing needs and requirements is not an exercise in writing, but an exercise in engineering."

To help ensure needs and requirements and sets of needs and requirements have the characteristics defined in this Guide, for each characteristic there is traceability between both the rules in this Guide and the concepts and activities discussed in the NRM.

Verification of the needs and requirements. Needs verification and requirements verification involves verifying that the need and requirement statements and sets of

needs and requirements have the characteristics of well-formed need and requirement statements and sets of needs and sets of requirements resulting from following the rules such as those in this Guide or similar organizational guide or standard.

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.

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 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. (Note: Currently many of these tools focus on the quality of requirement statements and not need statements. Hopefully, in the future, the tools will address both need and requirement statements and understand the differences between the two.).

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. The project team will determine how this score will be used in the definition and management of the requirements and requirements verification activities.

While these applications can identify some defects, the project team still must do the analysis to address and correct these defects. Given these applications do not address all the 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, e.g., analysis, traceability, allocation, etc., and thus must be done by the project team to address defects associated with these activities.

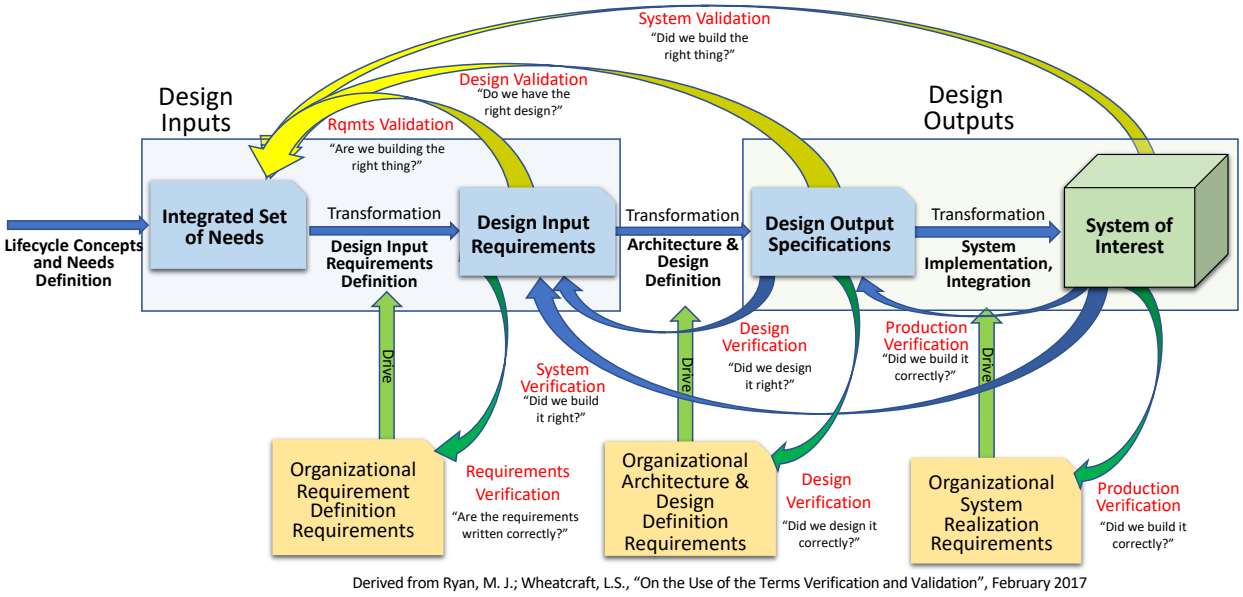
Validation of the needs and requirements. Validation of a need statement determines whether a need statement clearly communicates the intent of the lifecycle concepts or source from which it was derived or transformed. Validation of a requirement statement determines whether the requirement statement clearly

communicates the intent of the need, parent requirement, or source from which it was derived or transformed.

A need or requirement statement can be well-formed in accordance with the rules in this Guide (that is, the statement can be verified in terms of its quality in terms of its adherence to the rules) - but can speak to the wrong need or requirement or it may not communicate the true intent of the lifecycle concept, need, or parent requirement from which it was derived or transformed. Again, the focus is on 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 validation and the NRM Section 7 for a more detailed discussion on requirements validation.

Unlike needs and requirements verification, needs and requirements validation confirming that the intent of the source is effectively communicated 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.

1.9 Needs and Requirements in the Context of Verification and Validation,



Derived from Ryan, M. J.; Wheatcraft, L.S., "On the Use of the Terms Verification and Validation", February 2017

Figure 5: Needs and Requirements in the Context of Verification and Validation

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

Once the needs are verified and validated as discussed in the previous section, all subsequent artifacts are validated against the needs as shown in Figure 5. Once the resulting design input requirements are verified and validated, all subsequent artifacts are verified against those design input requirements, as shown in Figure 5.

Refer to the NRM Section 2.4 for a detailed discussion concerning verification and validation across the lifecycle and the context in which the terms are used.

1.10 Organizing and Managing Needs and Requirements

Today's systems are becoming increasingly complex and software intensive. This presents major challenges for organizations being able 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, and consistency. Because of the complexity, there are more people involved in the development of these systems spread over different geographical locations. This results in many of the documents being developed and managed within silos with limited collaboration.

Because of these issues, it is nearly impossible to keep all the needs and requirements contained within 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 resulting in no real single source of truth (SSoT). 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.

To address these issues the trend is to move from a document-centric to a data-centric practice of SE. As such, 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. In a data-centric practices of SE, needs and requirements are defined, recorded, and managed electronically in sets within a SE toolset.

As shown in Figure 5, for each SOI there is an integrated set of needs, a set of design input requirements, and a set of design output specifications for the SOI. As discussed in the NRM, there is a set of lifecycle concepts, 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 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, R40, and R41) defined in this Guide, within the sets it is a best practice for organizations to organize their needs and requirements 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.

1.11 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.
- Attributes of individual need and requirement statements from the NRM that help achieve the characteristics.
- Activities and concepts from the NRM that aid in the definition of need and requirement statements that have these characteristics.
- 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, provides guidance for helping understand the characteristics, and provides traceability for rules, attributes, and activities that help achieve these characteristics.
- Section 3 defines the characteristics of well-formed sets of needs and requirements, provides rationale for the characteristics, provides guidance for helping understand the characteristics, and provides traceability for rules, attributes, and activities that help achieve these 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. Included with each rule is an explanation of the rule and examples of the application of the rule. The intent of these rules is to define a more structured natural language in which needs and requirements are written to help clearly communicate the need and requirement statements and help avoid ambiguity often found when using an unstructured natural language to formulate need and requirement statements.
- 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 cross reference matrices mapping the rules to the characterizes and mapping concepts and activities discussed in the NRM to the characteristics.
- Appendix E includes a comment form.

Section 2: Characteristics of Need and Requirement Statements

This section defines the characteristics of individual need and requirement statements, provides rationale for the characteristics, and provides guidance for helping to understand and achieve the characteristics.

Also provided for each characteristic are traces to the rules defined in this Guide and the attributes, activities and concepts contained in the NRM associated with these 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 doing the activities and analysis discussed in the NRM and GtNR and through the use of attributes defined in the NRM.

Appendix D contains cross reference matrices mapping the rules to the characterizes and mapping concepts and activities discussed in the NRM to the characteristics.

The following characteristics are currently in harmony with those listed in the SE Handbook, the SEBoK, and ISO/IEC/IEEE 29148 and only slightly different to those in ISO/IEC/IEEE 15288. The authors acknowledge that other sources may have slightly different lists of characteristics and that there are dependencies and apparent overlap between several of the characteristics (Carson 2018).

The list of characteristics includes individual characteristics that provide an important perspective that is needed in order to craft well-formed requirement statements. Dependencies are addressed within either the rationale or guidance provided for each characteristic.

In defining needs and requirements, care should be exercised to ensure the need or requirement statement is appropriately crafted. The following characteristics of needs and requirements are elaborated in this Guide.

2.1 C1 - Necessary

Definition:
The need or requirement statement defines an essential capability, characteristic, constraint, or quality factor needed to satisfy a lifecycle concept, need, source, or parent requirement. If it is not included in the set of needs and requirements, a deficiency in capability or characteristic will exist which cannot be fulfilled by implementing other needs or requirements in the set.

Rationale:
The formal transformation of a need from a lifecycle concept must result in a need addressing a specific aspect of the lifecycle concept that is necessary in order to meet the goals, objectives, stakeholder expectations, drivers, constraints, risks that are included 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” both encompasses the characteristic “Correct” (C8) and enhances it to

ensure it is not only an "accurate representation of the specific aspect of the of the lifecycle concept or other source but is sufficient to ensure the specific aspect of the of the lifecycle concept or other source will be satisfied when the SOI realizes the needs.

The transformation of a need into a one or more requirements must result in a set of requirements that where each is necessary, and the set is sufficient in order to meet a need or set of needs for the entity from which it was transformed.

Each requirement, individually or in combination with other requirements in the set, must be sufficient to satisfy a specific stakeholder need, source, or parent requirement.

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 enhances it to ensure it is not only an "accurate representation of the entity need but is sufficient to ensure the need is satisfied by the child requirement when the SOI fulfills the requirements.

The development of child requirements via either decomposition or derivation must result in a set of child requirements that are necessary and sufficient in order to meet the intent of the need or source from which it is transformed or the allocated parent requirement the child is being developed in response to. Members of the set of child requirements can exist in different system element sets of requirements which the parent requirement was allocated to. 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. Refer to the NRM Section 6 concerning allocation/budgeting and interface requirements.

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, and maintenance. Unnecessary needs and requirements can lead to non-value-added work, additional cost, and unnecessary risk. Only necessary needs and the resulting requirements should therefore be included in the need and requirement sets. Once each need and requirement is proven to be necessary, the set of needs must then be a sufficient solution for the agreed-to lifecycle concepts and requirements must then be a sufficient solution to the set of needs. Together, the sets of needs and requirements form the design inputs to which the design outputs will be verified and validated against.

Guidance:

There is no such thing as a "self-derived" need or requirement! A need must be traced to 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.

A requirement must be traced to a need, parent, or source which could be one or more need(s) or higher-level allocated parent requirement(s)), lifecycle concept, driver, constraint, or risk.

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

- the need or requirement can be removed, and the remaining set will still result in the entity lifecycle concept or needs being satisfied;

- the intent of the need or requirement will be met by the implementation of other needs or requirements;
- the need or requirement cannot be traced back to a source, need, or parent requirement; or
- the author cannot communicate a valid reason (rationale) for the need or requirement.

One approach used by some organizations to limit the number of design input requirements is to take a "zero based" or "minimum viable product (MVP)" approach. Start by only including needs and requirements that are high priority and are critical or essential for an MVP. Then only add additional requirements that add value by reviewing each proposed need and requirement against the mission, goals, and objectives as well as drivers, constraints, risks, lifecycle concepts, and scenarios defined for the entity. If the need or requirement cannot be traced to one or more needs, parent requirement, or one of these sources, it is not necessary. The inclusion of rationale and other attributes defined in the NRM, Section 15, such as trace to source or parent, for each need and requirement also aids in communicating the necessity and intent of the need or requirement.

Caution: When developing child requirements in response to a need or an allocated parent or source, the set of child requirement must be limited to only those that can be proven to be necessary and sufficient to meet the intent of the need, parent, or source to which they are traced and not include any unnecessary requirements. Avoid gold plating and needs 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.

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 needs and requirements analysis is used where they are considered based on their 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.

Rules that help establish this characteristic:
R20 - /Singularity/AvoidPurpose
R30 - /Uniqueness/ExpressOnce

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A1 - Rationale
A2 - Trace to Parent
A3 - Trace to Source
A32 – Trace to Interface Definition
A33 - Trace to Dependent Peer Requirements
A34 – Priority
A35 – Criticality or Essentiality

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.1.5 – Attributes; 3.2.1.6 – Attributes; 3.2.2.1 – Analysis From Which Needs and Requirements Are Derived; 4.4.3 – Get Stakeholder Agreement; 4.5 – Lifecycle Concepts Analysis and Maturation; 4.5.3 – User of Diagrams and Models for Analysis; 4.5.7.1 – Model Development, Analysis, and Maturation; 4.6.3.4 – Needs Feasibility and Risk; 4.8 – Baseline and Manage

Lifecycle Concepts and Needs Definition Outputs; 5.1.2 – Perform Needs Verification; 6.2 – Perform Design Input Requirements Definition; 6.2.1 – Transforming Needs into Design Input Requirements; 6.2.2 – Establish Traceability; 6.2.3.6 – Interface Requirements Audit; 6.2.6.3 – Requirements Feasibility And Risk; 6.3 – Baseline and Manage Design Input Requirements; 6.4.7 – Use of Traceability and Allocation to Manage Requirements; 7.1.2 – Perform Design Input Requirement Verification; 7.2.2 – Perform Design Inputs Requirement Validation; 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.7 – Combine Allocation and Traceability to Manage Requirements;

2.2 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:

Level of abstraction refers to the level of detail and specificity of the information being communicated within a need or requirement statement.

The wording of needs is often stated at a higher level of abstraction than is appropriate for a requirement. *See the example under R31, Abstraction/SolutionFree*.

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. Design input, design-to "what" requirements should state what needs to be stated for that level of the entity, 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 4***.

A need or requirement stated at the wrong level for an entity is either not correct or may not be verifiable nor able to be validated at that level.

Refer to the NRM for an in-depth discussion on levels.

Guidance:

Refer to Section 1.6 for a discussion concerning needs and requirements and the entity to which they apply. Refer to Section 1.7 concerning the differences between needs and requirements.

Design input requirements must not be any more detailed or specific than is necessary for the level of the entity at which they are stated. In particular, the entity for which the requirement applies needs to be appropriate to the level of organization or architecture in which the entity lives. Unless there is a good reason, a requirement subject/noun should refer to the entity at the level the requirement is being expressed (not higher or lower).

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).

Needs and requirements should be stated for an entity at the level of integration at which system verification and system validation will be performed to verify the entity meets that requirement.

Requirements for lower-level entities stated within the requirements set for a higher-level entity may seem like implementation. I n 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 can be problematic because they may not have been allocated properly to the other entities at the next level of the architecture, resulting in missing requirements for those entities.

If the requirement is valid but stated at a lower or higher-level entity, determine what the appropriate level entity is and document the requirement for that level entity.

It is good practice, once the next level entity requirements are written, for the project team to do a "leveling" exercise, where they look at each requirement for the entity and determine whether or not 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.

Rules that help establish this characteristic:
R2 - /Accuracy/UseActiveVoice
R3 - /Accuracy/SubjectVerb
R31 - /Abstraction/SolutionFree

Activities and concepts associated with this characteristic: (Sections within the NRM)
4.5.4 – Levels of Detail and Abstraction; 4.6.3.2 – Appropriate to Level; 4.6.3.4 – Needs Feasibility and Risk; 6.2 – Perform Design Input Requirements Definition; 6.2.1 – Transforming Needs into Design Input Requirements; 6.2.1.4 – Appropriate to Level; 6.2.6.3 – Requirements Feasibility and Risk; 6.4.3 – Allocation – Flow Down of Requirements.

2.3 C3 - Unambiguous

Definition:
Need statements must be written such that the stakeholder intent is clear. Requirement statements must be stated such that the requirement can be interpreted in only one way by all the intended stakeholders.

Rationale:
A need or requirement statement must lend itself to a single interpretation of intent. An agreement is difficult to enact unless both parties are clear on the exact obligation. Ambiguity leads to multiple interpretations such that the stakeholder expectations may not be met. The intent of a need or requirement must be understood in the same way by the writer, the designer, and those doing verification and validation activities across the lifecycle following the "reasonable person" guideline. 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 is not correct nor able to be validated. An ambiguous requirement is not Verifiable (C7) nor Correct (C8).

Guidance:
When writing a need or requirement statement, ask whether it could be interpreted more than one way. For needs, ask whether, it can be validated, i.e., whether it is stated in such a way that evidence can be obtained that the stakeholder 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, i.e., whether it is stated in such a way that evidence can be obtained that the requirement has been met based on the wording of the requirement without having to interpret the meaning or make assumptions as to the meaning. The possibility of ambiguity is reduced by addressing these questions and applying the rules in this Guide. Additionally, it is useful for the parties (stakeholders) who are involved in the implementation of the needs and resulting requirements or system verification and system validation to be involved in the development, review, and baseline of the needs and resulting requirements. When they see needs or requirements that are ambiguous and their intent not clear, 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) take the development team and those involved in system verification and system validation on a walkthrough of the need or requirement set to ensure that needs and requirements are understood, individually and as a set. As discussed in Section 1.8, this activity is referred to as need or design input requirement validation. 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, to include contextual information to better understand the reason, and source of the requirement may provide additional insight of the intent, helping to reduce ambiguity. This may include supporting information or commentary on how the requirement was formed. When text only makes it difficult to communicate the intent of complex requirement, the inclusion of a diagram 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.

Rules that help establish this characteristic:
R1 - /Accuracy/SentenceStructure
R2 - /Accuracy/UseActiveVoice
R3 - /Accuracy/SubjectVerb
R4 - /Accuracy/UseDefinedTerms
R5 - /Accuracy/UseDefiniteArticles
R6 - /Accuracy/Units
R7 - /Accuracy/AvoidVagueTerms
R8 - /Accuracy/NoEscapeClauses
R9 - /Accuracy/NoOpenEnded
R10 - /Concision/SuperfluousInfinitives
R11 - /Concision/SeparateClauses
R12 - /NonAmbiguity/CorrectGrammar
R13 - /NonAmbiguity/CorrectSpelling
R14 - /NonAmbiguity/CorrectPunctuation
R15 - /NonAmbiguity/LogicalCondition
R16 - /NonAmbiguity/AvoidNot
R17 - /NonAmbiguity/Oblique
R18 - /Singularity/SingleSentence
R19 - /Singularity/AvoidCombinators
R22 - /Singularity/Enumeration
R23 - /Singularity/Context
R24 - /Completeness/AvoidPronouns
R28 - /Conditions/ExplicitLists
R32 - /Quantifiers/Universals
R33 - /Tolerance/ValueRange
R34 - /Quantification/Measurable
R35 - /Quantification/TemporalIndefinite
R36 - /UniformLanguage/UseConsistentTerms
R37 - /UniformLanguage/DefineAcronyms

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A1 - Rationale
A6 - System Verification or System Validation Success Criteria
A8 – System Verification or System Validation Method

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.1.1 – Communication; 3.2.1.2 – Power of Expression; 3.2.1.6 – Formal, Binding Agreement; 4.4.3 – Get Stakeholder Agreement; 4.6.3.1 – Managing Unknowns; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.1.2 – Perform Needs Verification; 5.2.2 -

Perform Needs Validation; 6.2.1.5 – Managing Unknowns; 6.3 – Baseline and Manage Design Input Requirements; 7.1.2 – Perform Design Input Requirements Verification; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.4 – Design Verification, 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.4 – Managing Unknowns

2.4 C4 - Complete

Definition:

The requirement statement sufficiently describes the necessary capability, characteristic, constraint, or quality factor to meet the need, source, or parent requirement from which it was transformed without needing other information to understand the requirement.

Rationale:

An agreement is not useful unless the obligation is complete and does not need further explanation. A well-formed requirement needs no further amplification to implement its intent. As an example, 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) or Data Dictionary). Additionally, requirements based on a standard or regulation need to include a reference to the specific location within the standard or regulation from which it was derived.

Each requirement should be understood in its own right without the overhead of having to understand a number of other requirements.

Baselined requirement statements should not contain To Be Defined (TBD), To Be Specified (TBS), or To Be Resolved (TBR) clauses. TBx can be used during the analysis and definition process to indicate ongoing work but should not be in the final requirement set. Resolution of the TBx designation may be iterative, in which case there should be an acceptable timeframe 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 managing unknows.

An incomplete requirement is not Verifiable (C7) nor Correct (C8) due to missing information (the 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 parent requirement because of these errors or omissions.

While fully appreciating that a requirement may require some context, the requirement statement itself should be a complete sentence that does not require reference to other statements to be understood in its basic form. Note, however, that a requirement can refer to other documents (for example, ICDs, standards, and regulations). When making these referrals, be specific to the sections of the documents that apply. Refer to a complete standard or regulation only if all the requirements within the standard or regulation apply to your specific SOI (which is a rare occurrence).

Caution: When referring to a standard or regulation, remember that standards and regulations are often written at a higher level of abstraction for a class of products similar to your SOI, but not necessarily your SOI specifically. It is a mistake to just copy and paste a requirement from a standard or regulation. Instead, you will need to derive requirements specific to your SOI at the appropriate level and then trace the resulting requirement(s) to the standard or regulation from which it was derived. Also refer to the NRM Section 6.2,1,2 for a more detailed discussion concerning complying with requirements within standards and regulations.

Requirements should not refer to one another through use of pronouns, nor should the understanding of the requirement assume the existence of a previous or subsequent requirement. This is especially important when requirements are managed within an application or database.

In a set of requirements contained in some form of "document", each requirement should be understood in its own right without having to understand the context as communicated in the requirements or headings surrounding it.

By following a specific pattern for a requirement statement, requirements tend to be "more complete", since these patterns provide guidance for specific information the requirement statement should include. See Appendix C for more information on patterns.

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

Rules that help establish this characteristic:
R1 - /Accuracy/SentenceStructure
R2 - /Accuracy/UseActiveVoice
R6 - /Accuracy/Units R7 - /Accuracy/AvoidVagueTerms R9 - /Accuracy/NoOpenEnded R11 - /Concision/SeparateClauses R23 - /Singularity/Context R24 - /Completeness/AvoidPronouns R25 - /Completeness/UseOfHeadings R27 - /CONDITIONS/EXPLICIT R33 - /Tolerance/ValueRange R34 - /Quantification/Measurable R35 - /Quantification/TemporalIndefinite R39 - /UniformLanguage/StyleGuide R40 - /MODULARITY/RELATEDREQUIREMENTS

Attributes that help establish this characteristic: *(Refer to the NRM Section 15.)*
A12 – Condition of Use
A32 - Trace to Interface Definition

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.1.2 – Power Of Expression; 3.2.1.3 – Managing Sets of Needs And Requirements; 3.2.1.6 – Formal, Binding Agreement; 4.4.3 – Get Stakeholder Agreement; 4.5 – Lifecycle Concepts

Analysis and Maturation; 4.6.3.1 – Managing Unknowns; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.1.2 – Perform Needs Verification; 6.2.1 – Transforming Needs into Design Input Requirements; 6.2.1.2 – Considerations For Each Type Of Requirement, Functional/Performance; 6.2.1.5 – Managing Unknowns; 6.2.3.6 – Interface Requirements Audit; 6.3 – Baseline and Manage Design Input Requirements; 7.1.2 – Perform Design Input Requirements Verification; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.4 – Design Verification, 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.4 – Managing Unknowns

2.5 C5 - Singular

Definition:

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

Rationale:

The formal transformation from a lifecycle concept to 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 from a need, source, or allocated parent requirement to 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, depends 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. A need or requirement with multiple thoughts is difficult to allocate and to trace to a parent or source.

A nonsingular need or requirement is neither Verifiable (C7) nor Correct (C8).

Requirements patterns are useful ways to ensure singularity (see Appendix C).

Guidance:

Keep the need or requirement statement limited to one quality, characteristic, capability, function, or constraint. Understand how the statements fit into the allocation and traceability philosophy for the project.

Use the project standard patterns for writing needs and requirements.

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 verified differently. The presence of the conjunction “and” in a need or requirement statement should always prompt the writer to consider whether or not the statement is singular.

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 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 (you would never do one action 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.

Rules that help establish this characteristic:

R9 - /Accuracy/NoOpenEnded

R18 - /Singularity/SingleSentence

R19 - /Singularity/AvoidCombinators

R20 - /Singularity/AvoidPurpose

R21 - /Singularity/AvoidParentheses

R22 - /Singularity/Enumeration

R23 - /Singularity/Context

R39 - /UniformLanguage/StyleGuide

2.6 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.

An infeasible need or requirement cannot be satisfied because

a. it breaks the laws of physics,
b. it violates laws or regulations in an applicable jurisdiction,
c. it conflicts with another requirement and cannot be concurrently satisfied, or
d. it 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:

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

Feasible implies the existence of a possible feasible solution to satisfying a 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 be the requirements transformed from those needs.

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. Before allowing a need into your 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 will not be met and verified.

A useful tool for assessing risk is the use of Technology Readiness Levels (TRLs) to determine and compute the maturity of a critical 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 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.

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). The need or requirement is inherently infeasible where the requirement is either internally contradictory or it violates the laws of physics. More often, feasibility must be examined for sets of needs or requirements associated with a single entity to ensure there is no conflict in the set of requirements (C12).

The measurement of feasibility is not always easy to assess. Measurement of feasibility is based on the degree of risk in successfully implementing the requirement within program constraints, unless precluded by physics (cannot be done, period) or requirements conflict. As stated above, TRLs can be a useful measure of risk for requirements dependent on the maturity of an essential technology.

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

Rules that help establish this characteristic:
R26 - /Realism/AvoidAbsolutes
R33 - /Tolerance/ValueRange

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A6 - System Verification or System Success Criteria
A8 - System Verification or System Validation Method
A26 – Stability/Volatility
A36 - Risk (of implementation)
A38- Key Driving Need or Requirement (KDN/KDR)

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.1.6 – Formal, Binding Agreement; 3.2.2.1 – Analysis From Which Needs And Requirements Are Derived; 4.3.6.2 – Technology Maturity; 4.3.7.1 – Classes of Risk - Development Risk; 4.5 – Lifecycle Concepts Analysis And Maturation; 4.5.1 – Feasibility; 4.5.7.4 – Zeroing in on a Feasible Architecture and Design; 4.6.3.1 – Managing Unknowns; 4.6.3.4 – Needs Feasibility and Risk; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.2.2 – Perform Needs Validation; 6.2.1.5 – Managing Unknowns; 6.2.6.3 – Requirements Feasibility and Risk; 6.3 – Baseline and Manage Design Input Requirements; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.4 – Design Verification, 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.4 – Managing Unknowns

2.7 C7 - Verifiable

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

Rationale:
Unless a requirement is written in a way that allows design verification or 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 success criteria can be defined, and the SOI can be verified such that that success criteria has been met, i.e., there is no ambiguity regarding what the requirement statement communicates and there are no missing characteristics within the requirement, i.e., the requirement is complete. An unverifiable requirement can result in multiple, objective observers (for example, designers or testers) interpreting the requirement differently making it difficult to verify the SOI meets the requirement. Verifiability is a necessary condition for establishing the characteristics: Appropriate (C2), Unambiguous (C3), Complete (C4), Singular (C5), Feasible (C6), Conforming (C9), Consistent (C11), and Comprehensible (C13). Therefore, verifiability should be addressed as the initial criterion and a basis for examining these other characteristics.

Guidance:
It is important not to confuse the phrase "requirement verification" vs. "design verification" or "system verification" as shown in Figure 5 and discussed in Section 1.7 and 1.8. This characteristic, verifiable, is about design and system verification – showing that the design can be verified such that, when realized, will result in a SOI that meets the requirement and verifying that the realized SOI meets the requirement. Write each requirement statement in a way that allows the design or system to be verified that the requirement has been met by one of the four standard verification methods (inspection, analysis, demonstration, or test). Various kinds of requirements are verifiable in different ways, and this will influence the way the requirement is written. In general, to be verifiable, a requirement should be measurable. (Refer to the NRM and GtVV for a detailed discussion on verification and validation across the lifecycle.)

A requirement is considered to be verifiable if:

1) a verification case can be determined from the requirement statement that allows for complete examination of all aspects of the requirement and precise determination of all values, including tolerances, such that success or failure can be determined, and
2) the requirement content is adequate to completely define the expected behavior, characteristics, conditions, and success criteria for the actual verification activity.

The measure of verifiability is the completeness and quality of the requirement: does it contain all the necessary information to establish what, how well, and under what conditions, i.e., is it Complete (C4)?

Each requirement must have all the necessary information to be verifiable – the wording of the requirement is unambiguous (C3) and will have the same meaning to all observers or readers.

This can be facilitated by use of a standard template as described in Appendix C – see also characteristic Conforming (C9).

A customer may specify, “The aircraft’s range shall be as long as possible.” This statement is ambiguous and unverifiable. This type of requirement is a signal that a trade study is needed to establish a verifiable maximum range requirement. [Note: This is a good example of a need written as a requirement. While “The stakeholders need the aircraft’s range to be as long as possible” is an acceptable level of abstraction for a need statement, it is not acceptable as a requirement.]

The most usual causes for a requirement not to be verifiable are:

- no clear definition of the correct functional behavior, conditions, and states.
- lack of accuracy or feasibility in the ranges of acceptable performance.
- use of ambiguous terms.
- failure to define a feasible lifecycle concept and associated stakeholder need from which the requirement was transformed.
- the requirement is not feasible.

When writing requirement statements, use a verification point-of-view to imagine yourself performing the verification activity and define what evidence is needed to show that the requirement’s intent has been achieved as defined by the success criteria.

It is a recommended best practice to define the verification success criteria, strategy, and method attributes when forming the requirement statement. When this is done, the quality of the requirement statement will improve resulting in this characteristic (verifiable).

Whether the SOI will be able to be verified against a requirement can be assessed during early system verification and design verification activities discussed in the NRM and GtVV.

Useful questions to ask of a requirement statement are:

- How will I know if the requirement has been met? If the requirement has been properly quantified, it will provide a precise answer to this question.
- What are the mandatory and desirable levels of performance required? The result of this may be that several values are provided describing the tolerance and trade-space allowed for this requirement.
- Is what the requirement states what I want to verify the SOI against? If not, then rewrite the requirement to state what is intended.

For example, it is best to verify you obtain specific performance data rather than verifying the system has sensors to provide that data. It may have the sensors, but do they result in the actual quantity and quality of data needed? This is also a good example of design input versus design

output. The need for the data is a design input, the sensors used to obtain this data are design outputs. All too often, when a design output requirement is stated there was a failure to state why the design output was needed – the proper design input. Because of this it could seem the SOI passed system verification, but in reality, it did not –the SOI was verified against the wrong requirement!

Rules that help establish this characteristic:
R1 - /Accuracy/SentenceStructure
R2 - /Accuracy/UseActiveVoice
R3 - /Accuracy/SubjectVerb
R4 - /Accuracy/UseDefinedTerms
R5 - /Accuracy/UseDefiniteArticles
R6 - /Accuracy/Units
R7 - /Accuracy/AvoidVagueTerms
R8 - /Accuracy/NoEscapeClauses
R9 - /Accuracy/NoOpenEnded
R10 - /Concision/SuperfluousInfinitives
R11 - /Concision/SeparateClauses
R12 - /NonAmbiguity/CorrectGrammar
R13 - /NonAmbiguity/CorrectSpelling
R15 - /NonAmbiguity/AvoidNot
R16 - /NonAmbiguity/AvoidNot
R17 - /NonAmbiguity/Oblique
R18 - /Singularity/SingleSentence
R24 - /Completeness/AvoidPronouns
R26 - /Realism/AvoidAbsolutes
R27 - /Conditions/Explicit
R28 - /Conditions/ExplicitLists
R32 - /Quantifiers/Universals
R33 - /Tolerance/ValueRange
R34 - /Quantification/Measurable
R35 - /Quantification/TemporalIndefinite

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A6 – System Verification or System Validation Success Criteria
A8 - System Verification or System Validation Method
A32 - Trace to Interface Definition

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.1.1 – Communication; 3.2.1.2 – Power of Expression; 3.2.1.6 – Formal Binding Agreement; 3.2.1.7 – System Verification and System Validation; 3.2.2.5 – Support Simulations; 4.4.3 – Get

Stakeholder Agreement; 4.5 – Lifecycle Concepts Analysis and Maturation; 4.6.3.1 – Managing Unknowns; 6.2 – Perform Design Input Requirements Definition; 6.2.1.2 – Considerations For Each Type Of Requirement, 6.2.1.5 – Managing Unknowns; 6.2.3.6 – Interface Requirements Audit; 6.2.5 – Plan for System Verification; 7.1.2 – Perform Design Input Requirements Verification; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.4 – Design Verification, 14.2.4 – Managing Unknowns 14.2.9 – Managing System Verification and System Validation.

2.8 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 parent requirement from which it was transformed.

Rationale:

Needs are transformed from a source: lifecycle concept, stakeholder expectation, drivers and constraints, goals, objectives, risks, etc. defined as part of the lifecycle concept and needs definition activities discussed in the NRM and GtNR. The resulting needs transformed from these sources must be validated to determine the accuracy of the transformation.

An incorrect need can result in a need that does not reflect the lifecycle concept or sources from which it was transformed or a requirement that is not compliance with a need, source, or parent requirement from which it was derived.

The requirement must be able to be validated to ensure that the requirement communicates the right thing (for example, value, tolerance, and units) such that the need(s) from which the requirement was transformed will be met. It must be able to be shown (validated) that achievement of the requirement, as written, will result in meeting the intent of the need(s) from which it was transformed.

Correct implies “no errors” both from the perspective of the inclusion of incorrect information, the omission of required information, and avoidance of ambiguous wording.

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

Guidance:

Ensure the need statement reflects:

- the accurate interpretation of the concept, stakeholder expectation, drivers and constraints, goals, objectives, risks, etc. from which it was transformed;
- an accurate understanding of the problem or opportunity and underlying goals and objectives;
- the model or diagram from which the need was extracted so the need traces to the model; and
- an accurate representation of the 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 integrated set of needs.

Ensure the requirement statement reflects:

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

Incorrect information can mean having the wrong:

- values,
- functions,
- conditions, or
- other characteristics identified in the need or requirement.

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.

Rules that help establish this characteristic:
R1 - /Accuracy/SentenceStructure
R6 - /Accuracy/Units
R11 - /Concision/SeparateClauses
R12 - /NonAmbiguity/CorrectGrammar
R14 - /NonAmbiguity/CorrectPunctuation
R16 - /NonAmbiguity/AvoidNot
R26 - /Realism/AvoidAbsolutes
R27 - /CONDITIONS/EXPLICIT
R32 - /Quantifiers/Universals
R33 - /Tolerance/ValueRange
R36 - /UniformLanguage/UseConsistentTerms

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A6 – System Verification or System Validation Success Criteria
A8 - System Verification or System Validation Method
A12 - Condition of Use

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.2.1 - Analysis from Which Needs and Requirements are derived; 3.2.2.4 – Identify and Manage Interdependencies; 4.4.3 – Get Stakeholder Agreement; 4.5 – Lifecycle Concepts Analysis and Maturation; 4.5.3 – User of Diagrams and Models for Analysis; 4.5.7.1 – Model Development, Analysis, and Maturation; 4.6.3.1 – Managing Unknowns; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.2.2 – Perform Needs Validation; 6.2 - Perform Design Input Requirements Definition; 6.2.1.2 – Considerations For Each Type Of Requirement, 6.2.1.5 – Managing Unknowns; 6.2.3.6 – Interface Requirements Audit; 6.2.6.2 –

Completeness, Correctness, and Consistency; 6.3 – Baseline and Manage Design Input Requirements; 6.4.7 – Use of Traceability and Allocation to Manage Requirements; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.4 – Design Verification, 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.4 – Manage Unknowns; 14.2.7 – Combine Allocation and Traceability to Manage Requirements.

2.9 C9 - Conforming

Definition:

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 and requirement statement is easier to write, understand, and review. Also, when conforming to an approved standard, the quality of the individual need and requirement statement will improve.

Conforming to standards or templates for needs and requirements will help to identify missing information resulting in the characteristics of complete (C10), unambiguous (C3) and verifiable (C7).

Guidance:

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>.” If goals (non-mandatory) are included in the set of needs then the following pattern could be used: “The stakeholders would like the <subject clause> to <action verb clause> <object clause>, <optional qualifying clause>.”

For the transformation from a need to a requirement to be formal, the structure of the resultant requirement statement must also be formal. For example, all requirements 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>.” See R1 for more detail on requirement formats and Appendix C for more detailed information on the use of templates and patterns.

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

In many instances, there are applicable government, industry, and product standards, specifications, and interfaces 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 people responsible for writing the needs and requirements may need to conform to the customer's processes and standards.

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

Rules that help establish this characteristic:
R1 - /Accuracy/SentenceStructure
R12 - /NonAmbiguity/CorrectGrammar
R18 - /Singularity/SingleSentence
R30 - /Uniqueness/ExpressOnce
R36 - /UniformLanguage/UseConsistentTerms
R37 - /UniformLanguage/DefineAcronyms
R38 - /UniformLanguage/AvoidAbbreviations
R39 - /UniformLanguage/StyleGuide
R40 - /Modularity/RelatedRequirements

Activities and concepts associated with this characteristic: (Sections within the NRM)
4.6.2.3 – Organizing the Integrated Set of Needs; 5.1.2 – Perform Needs Verification; 6.2.1.1 – Organizing Sets of Design Input Requirements; 7.1.2 – Perform Design Input Requirements Verification.

Section 3: Characteristics of Sets of Needs and Requirements

This section defines the characteristics of sets of needs and requirements, provides rationale for the characteristics, and provides guidance for helping understand the characteristics. Also provided for each characteristic are traces to the rules defined in this Guide and the attributes, activities and concepts contained in the NRM associated with these characteristics. In most cases 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-written set of needs and requirements must also be considered.

3.1 C10 - Complete

Definition:
The need or requirement set for a given SOI should stand alone such that it sufficiently describes the necessary capabilities, characteristics, functionality, performance, drivers, constraints, interactions, standards, regulations, and/or quality factors without requiring other sets of needs or requirements at the appropriate level of abstraction.

Rationale:
If the formal derivation of needs from agreed-to lifecycle concepts results in individual needs that are necessary, the set of needs must be a sufficient representation of the lifecycle concepts and other sources from which they were derived—that is, all necessary needs have been included to implement the lifecycle concepts and other sources, and that all unnecessary needs have been excluded. If the formal transformation of needs, parent requirements, and other sources into requirements results in individual requirements that are necessary, the set of requirements must be a sufficient representation of the set of needs, parent requirements, and other sources for the entity from which it was transformed — that is, all necessary requirements have been included and all unnecessary requirements have been excluded. 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 set of parent requirements allocated to the entity. Complete must also address the possible conditions a system may experience, including anomalies. Completeness therefore requires an additional analysis of all possible conditions to ensure that the required behavior for specific conditions is consistent with stakeholder needs. These are the kinds of requirements (“what if?”) that might not be addressed as part of initial analysis because the focus is on desired behavior under expected conditions. These additional requirements derived from anomalous conditions may not arise until the system development is underway and represent risk in system development and difficulty in ensuring completeness.

Unspecified anomalous conditions can be handled using an “else” statement in the requirements. However, the required behavior still needs to be validated by the stakeholders

A set of needs or 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 SOI meets its intended use in the operational environment when operated by the intended users.

A set of needs or requirements cannot be complete if any individual need or requirement is not complete (C4) or contains any unnecessary needs or requirements (C1).

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, such 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 in the set may also result in a set of needs that is not Feasible (C12).

Guidance:

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

For each SOI, the integrated set of needs represents a complete definition of the stakeholder expectations, goals, objectives, lifecycle concepts, drivers, constraints, and risks to be mitigated whether explicitly stated or undocumented implicit expectations. Addressing implicit stakeholder expectations is a key part of lifecycle concept and needs definition activities and transforming the resulting integrated set of needs into a set of design input requirements that when realized will result in a SOI that meets those needs, even if not originally stated. *Any implicit needs should be captured and included explicitly in the integrated set of needs. It is the responsibility of those defining the integrated set of needs, to establish bi-directional traceability between each need and its source.*

A complete requirements set represents a complete transformation of the needs for an SOI, both explicitly stated as needs and the implicit needs not documented (part of the transformation process). Addressing implicit needs (and making them explicit) is a key part of requirements analysis defining the requirement set and deriving requirements that result in those needs being met, even if not originally stated as needs. *Note: This should not be an issue if the organization defined a complete integrated set of needs as described in the previous paragraph. However, if this set of needs was not defined, then implicit needs could be an issue and will need to be considered when writing the requirement set. It is the responsibility of those defining the set of requirements to establish bi-directional traceability between each design input requirement and the need or source the requirement is being transformed.*

A complete requirements set also represents a complete transformation of higher-level requirements that were allocated to the SOI into a necessary and sufficient set of child requirements, that when implemented, will result in the intent of the allocated parent 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 SOI to 1) validate the allocations were correct, 2) develop the 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 SOI 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 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 today's 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 SOI. 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 SOI and other system is defined. In the case of standards and regulations, reference should be made to the specific requirements that apply to the SOI, rather than the whole document.

Completeness can be facilitated through traceability, allocation, and budgeting.

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 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 R41 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 requirements.

Rules that help establish this characteristic:
R3 - /Accuracy/SubjectVerb
R29 - /Uniqueness/Classify
R40 – /Modularity/Related
R41 - /Modularity/Structured

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A2- Trace to Parent
A4 - States and Modes
A40 - Type/Category

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.2.1 – Analysis from Which Needs and Requirements are Derived; 3.2.2.2 – Completeness; Identify and Manage Interdependencies; 4.3.3 – Identify External and Internal Stakeholders; 4.4.3 – Get Stakeholder Agreement; 4.4.4 – Completeness; 4.5.3 – User of Diagrams and Models for Analysis; 4.5.7.1 – Model Development, Analysis, and Maturation; 4.6.2.3 – Organizing the Intergrade Set of Needs; 4.6.3.3 – Completeness of the Integrated Set of Needs; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.1.2 – Perform Needs Verification; 5.2.2 – Perform Needs Validation; 6.2 – Perform Design Input Requirements Definition; 6.2.1 – Transforming Needs into Design Input Requirements; 6.2.1.1 – Organizing the Sets of Design Input Requirements; 6.2.1.2 – Considerations for Each Type of Requirement; 6.2.2 – Establish Traceability; 6.2.3.6 – Interface Requirements Audit; 6.2.6.2 – Completeness, Correctness, and Consistency; 6.3 – Baseline and Manage Design Input Requirements; 6.4.3 – Allocation – Flow Down of Requirements; 6.4.4 - Defining Child Requirements that Meet the Intent of the Allocated Parents; 6.4.5 – Budgeting of Performance, Resource, and Quality Requirements; 6.4.7 – Use of Traceability and Allocation to Manage Requirements; 7.1.2 – Perform Design Input Requirements Verification; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.5 – Design Validation, 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.7 – Combine Allocation and Traceability to Manage Requirements; 14.2.8 – Managing Interfaces.

3.2 C11 - Consistent

Definition:
The set of needs contains individual needs that are unique, do not conflict with or overlap with other needs in the set, and the units and measurement systems they use are homogeneous. The language used within the set of needs is consistent (i.e., the same words are used throughout the set to mean the same thing). The set of requirements contains individual requirements that are unique, do not conflict with or overlap with other requirements in the set, and the units and measurement systems they use are homogeneous. The language used within the set of requirements is consistent (i.e., the same words are used throughout the set to mean the same thing) All terms used within the need and requirement statements are consistent with the architectural model, project glossary, and project data dictionary.

Rationale:
Conflicting needs and requirements lead to an incomplete 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. Frequently, customer and other relevant stakeholder expectations or design constraints conflict with one another and need to be reconciled. Additionally, 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 requirement set.

Needs and requirements that are inconsistent and conflicting with other requirements are not Correct (C8).

Needs and requirements that are inconsistent and conflicting with other 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).

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

Guidance:

It is a challenge spotting conflicts between needs and requirements when the set of needs and requirements is large, as is often the case in today's increasingly complex, software intensive systems.

As discussed in the NRL and GtNR, it is important to identify 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 is 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. Buy grouping like needs and requirements together, it will be easier to identify inconsistencies.

See R41 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 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 requirements as well as between needs and the resulting requirements. Using a software tool, e.g., diagraming and modeling, to manage relationships and dependencies can help in identifying conflicts and manage these relationships and dependencies.

Define an ontology for the project and use a glossary or data dictionary to ensure consistent use of terms and abbreviations throughout the need and requirement sets.

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 user of terminology.

Pay special attention to interface requirements to make sure they are consistent with the interface requirements for other systems your SOI interacts with. Ideally, both systems 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, e.g., ICD or 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.

Rules that help establish this characteristic:
R4 - /Accuracy/UseDefinedTerms
R29 - /Uniqueness/Classify
R30 - /Uniqueness/ExpressOnce
R36 - /UniformLanguage/UseConsistentTerms
R37 - /UniformLanguage/DefineAcronyms
R38 - /UniformLanguage/AvoidAbbreviations
R39 - /UniformLanguage/StyleGuide
R40 - /Modularity/RelatedRequirements
R41 - /Modularity/Structured

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A32 - Trace to Interface Definition
A33 - Trace to Dependent Peer Requirements

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.2.1 – Analysis from Which Needs and Requirements are Derived; 3.2.2.3 – Consistency; 3.2.2.4 – Identify and Manage Interdependencies; 4.4.3 – Get Stakeholder Agreement; 4.5 – Lifecycle Concepts Analysis and Maturity; 4.5.3 – User of Diagrams and Models for Analysis; 4.5.7.1 – Model Development, Analysis, and Maturation; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.1.2 – Perform Needs Verification; 6.2 – Perform Design Input Requirements Definition; 6.2.2 – Establish Traceability; 6.2.2.1 – Establishing Traceability Between Dependent Peer Requirements; 6.2.3.6 – Interface Requirements Audit; 6.2.6.2 – Completeness, Correctness, and Consistency; 6.3 – Baseline and Manage Design Input Requirements; 6.4.3 – Allocation – Flow Down of Requirements; 6.4.5 – Budgeting of Performance, Resource, and Quality Requirements; 6.4.7 - User of Traceability and Allocation to Manage Requirements; 7.1.2 – Perform Design Input Requirement Verification; 7.2.2 -Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.4 – Design Verification; 8.5 – Design Validation; 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.4 - Managing Unknowns; 14.2.8 – Managing Interfaces

3.3 C12 - Feasible

Definition:
Sets of needs and requirements can be realized within entity constraints (for example, 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 sets of needs and 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.

Guidance:
While individual need 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 requirements. (Like the old saying “The straw that broke the camel’s back”.) For example, the following are feasible, individual requirements for a laptop computer: weighs less than 1.4 kg, has a storage capacity of 1 TByte, has 4 GByte of RAM, has a wireless network interface, has an Ethernet network interface, has a USB interface, 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 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 and requirement statements, determining feasibility of a set of needs or 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. A set of needs cannot be feasible if any of the individual needs are not feasible (C6). Feasibility of a specific need statement must be determined in the context of all needs for a specific entity set and can be evaluated without regard to the sources from which they were derived.

Needs within the same entity set of needs are analyzed together with respect to program constraints and technology maturity to determine feasibility. A set of needs could not be feasible due to inconsistency (C11), physical impossibility, or excessive program risk.

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, constraints, and risk. 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 cannot be feasible if any of the individual requirements are not feasible (C6). Feasibility of a specific requirement must be determined in the context of all requirements for a specific entity set and can be evaluated without regard to the parent requirements.

Requirements within the same entity set of requirements are analyzed together with respect to project constraints, technology maturity, and risk to determine feasibility. A set of requirements may not be feasible due to inconsistency (C11), physical impossibility, or excessive program risk.

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) for the set of requirements.

An approach that can be used is to consider the set of needs or requirements to be a “bucket” 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.

Rules that help establish this characteristic:
R26 - /Realism/AvoidAbsolutes
R33 - /Tolerance/ValueRange
R34 - /Quantification/Measurable

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A26 – Stability/Volatility
A31 - Status (of implementation)
A36 – Risk (of implementation)
A38 - Key Driving Need or Requirement (KDN/KDR)

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.2.1 – Analysis from Which Needs and Requirements are Derived; 4.3.6.2 – Technology Maturity; 4.3.7.1 – Classes of Risk – Development Risk; 4.5.1 – Feasibility; 4.5.7.4 – Zeroing in on a Feasible Architecture and Design; 4.6.3.4 – Needs Feasibility and Risk; 4.8 – Baseline and

Manage Lifecycle Concepts and Needs Definition Outputs; 5.2.2 – Perform Needs Validation; 6.2 – Perform Design Input Requirements Definition; 6.2.6.3 – Requirements Feasibility and Risk; 6.3 – Baseline and Manage Design Input Requirements; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.5 – Design Validation, 14.2.1 – Baseline Needs, Requirements, and Specifications;

3.4 C13 - Comprehensible

Definition:

The set of needs and set of resulting requirements must 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 (C7), and Correct (C8).

Guidance:

Information needed to understand the context of the needs and requirements should be included with 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, 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.

Using an integrated, collaborative, multidiscipline project team performing the development process activities concurrently will help stakeholder to 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.

Rules that help establish this characteristic:
R4 - /Accuracy/UserDefinedTerms
R18 - /Singularity/SingleSentence
R36 - /UniformLanguage/UseConsistentTerms
R37 - /UniformLanguage/DefineAcronyms
R38 - /UniformLanguage/AvoidAbbreviations
R39 - /UniformLanguage/StyleGuide
R40 - /Modularity/RelatedRequirements
R41 - /Modularity/Structured

Attributes that help establish this characteristic: (Refer to the NRM Section 15.)
A1 - Rationale

Activities and concepts associated with this characteristic: (Sections within the NRM)
3.2.1.2 – Power of Expression; 3.2.1.3 – Managing Sets of Needs and Requirements; 3.2.1.5 – Attributes; 3.2.1.6 – Formal Binding Agreement; 3.2.2.1 – Analysis from Which Needs and Requirements ae Derived; 3.2.2.2 – Completeness; 3.2.2.3 – Consistency; 3.2.2.4 – Identity and Manage Interdependencies; 3.2.2.5 – Support Simulations; 3.2.2.6 – Key to Understanding 4.4.3 – Get Stakeholder Agreement; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.2.2 – Perform Needs Validation; 6.2 – Perform Design Input Requirements Definition; 6.3 – Baseline and Manage Design Input Requirements; 7.2.2 – Perform Design input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.5 – Design Validation, 14.2.1 – Baseline Needs, Requirements, and Specifications;

3.5 C14 - Able to be validated

Definition:
It must be able to be validated that the integrated 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 able to be validated that the set of requirements will lead to the achievement of the integrated 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 the 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. It must be able to be shown at any lifecycle stage that achievement of the set of needs will result in the lifecycle concepts from which they were transformed and the achievement of the set of requirements will result in meeting the set of needs from which they were transformed.

For a set of needs or requirements to be able to be validated, the set must be Complete (C10), Consistent (C11), Feasible (C12), and Comprehensible (C13). Refer to the NRM and GtVV for more details on validating sets of needs and sets of requirements.

Design validation is making sure 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 making sure 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 to make sure they have been met—as a set. Refer to the NRM and GtVV for more details on design and system validation.

Guidance:

The system lifecycle operational scenarios, concepts, and use cases—from which the needs were transformed, and the requirements 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 from which they were derived, and the requirements can be validated to meet the needs 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.

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?” or “Have we correctly and completely captured the needs the system must address? 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 requirements are a major focus during early system validation as well as during design validation activities discussed in the NRM and GtVV.

Rules that help establish this characteristic: (Refer to the NRM Section 15.)

R3 - /Accuracy/SubjectVerb

R4 - /Accuracy/UseDefinedTerms

R36 - /UniformLanguage/UseConsistentTerms

R37 - /UniformLanguage/DefineAcronyms

R38 - /UniformLanguage/AvoidAbbreviations

R39 - /UniformLanguage/StyleGuide

R41 - /Modularity/Structured

Activities and concepts associated with this characteristic: (Sections within the NRM)

3.2.1.1 – Communication; 3.2.1.2 – Power of Expression; 3.2.1.6 – Formal, Binding Agreement; 3.2.1.7 – System Verification and System Validation; 3.2.2.1 – Analysis from Which Needs and Requirements ae Derived; 3.2.2.2 – Completeness; 3.2.2.3 – Consistency; 3.2.2.4 – Identity and Manage Interdependencies; 3.2.2.5 – Support Simulations; 3.2.2.6 – Key to Understanding; 4.4.3

– Get Stakeholder Agreement; 4.7 – Plan for System Validation; 4.8 – Baseline and Manage Lifecycle Concepts and Needs Definition Outputs; 5.1.2 – Perform Needs Verification; 5.2 – Needs Validation; 5.2.2 – Perform Needs Validation; 6.2 – Perform Design Input requirements Definition; 6.2.3.6 – Interface Requirements Audit; 6.3 – Baseline and Manage Design Input Requirements; 6.4.7 – Use of Traceability and Allocation to Manage Requirements; 7.1.2 – Perform Design Input Requirements Verification; 7.2 – Design Input Requirements Validation; 7.2.2 – Perform Design Input Requirements Validation; 8.1 – Design Definition Process Overview, 8.2 – Early System Verification and System Validation; 8.5 – Design Validation, 14.2.1 – Baseline Needs, Requirements, and Specifications; 14.2.7 – Combine Allocation and Traceability to Manage Requirements; 14.2.8 – Manage Interfaces, 14.2.9 – Managing System Verification and System Validation.

Section 4: Rules for Need and requirement statements and Sets of Needs and Requirements

This section defines the rules for individual need and requirement statements and sets of needs and requirements that help them to be formulated such that they will have the characteristics defined in the previous sections. Included with each rule is an explanation of the rule, examples of the application of the rule with a trace to the characteristics that are supported by the rule.

The rules in this Section provide a structure to the language used to write the need and requirements statements. Using this structured, natural language approach helps avoid much of the ambiguity when using an unstructured, natural language to expressing needs and requirements.

Note: Because need statements are written from the perspective of the stakeholders at a higher level of abstraction, some of the rules in this section may not always apply. 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. For example, rules that address the verifiability or completeness of a requirement statement probably will not apply to a need statement.

In addition to the rules 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).

4.1 Accuracy

4.1.1 R1 - /ACCURACY/SENTENCESTRUCTURE

Use a structured, complete sentence.

Elaboration:
The structure of needs and requirements statements must be in the form of a complete sentence, the simplest form of which is: <subject> <verb> <object>. The subject is essential because it is the entity undertaking the action; the verb is essential because it is the action being performed; and the object is essential because it must be clear as to which entity is being acted upon. The subject and objects are therefore normally entities and for a requirement statement the verb is “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 SOI will be verified against the requirement. Currently, no other form, other than textual “shall” requirement statements, have been shown to meet these characteristics. Some organizations may use other verbs such as “must”, “will”, “should”, or “may”—regardless, whichever verb is used, the

mandatory, contractually binding nature of the action must be defined somewhere in the terms and conditions of the contract to made clear the statement is a requirement.

Expanding on the <subject> <verb> <object> form, requirements have a general form of

The <entity> shall <entity response> <qualifying statement> which can be further expanded to

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

In terms of a function/performance requirement, the <action verb> <object> pair is communicated as a function/object pair (heat coffee) and the <measurable outcome> is a specific performance outcome of performing that function (to a temperature of 120deg F.) The <qualifying statement> is additional information needed to clearly communicate the intent of the requirement as discussed below.

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, and the <measurable outcome> is a pointer where the specific interaction is defined (e.g., Interface Control Document).

Need statements have a similar form:

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

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

For need statements the outcome may be at a higher level of abstraction and not as specific as <measurable outcome> for a requirement statement as discussed in Section 2 and R3.

If a <qualifying statement> is needed to clearly communicate the intent of the action verb on the object is not stated explicitly within the need or requirement statement, the need or requirement statement is not Complete (C4)), Verifiable (C7), nor Correct (C8) unless the qualifying statement 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 (i.e., and ICD).

ISO/IEC 29148 states that a more-complete, typical sentence form for a functional requirement is:

When <condition clause>, the <subject clause> shall <action verb clause> <object clause> <optional qualifying clause>.

There are several agreed types of condition clauses:

Event-driven requirements:

When <optional preconditions/trigger>, the <system name> shall <system response>.

In the event of <specific event>, the <system name> shall <system response>.

Behavior driven requirements:

If <optional preconditions/trigger>, then the <system name> shall <system response>.

State-driven requirements:

While <entering, exiting or in a specific state (or mode)>, the <system name> shall <system response>.

See Appendix C for additional example patterns using clauses.

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 (C7), nor Correct (C8) unless the condition clause is included.

Note: While IEEE/ISO/IEC 29148 shows the conditional phrase at the beginning of the sentence, some organizations prefer to put the conditional phrase at the end of the sentence.

Organizations need to define the desired form in their standards, guides, and processes and then be consistent in the form being used. As long as the result is clear and unambiguous, either form is acceptable.

See also R2, R3, R11, R18, R27, and Appendix C.

Examples:

Condition examples:

Event-driven requirements:

When Continuous_Ignition is commanded by the Aircraft, the Control_System shall switch on Continuous_Ignition.

In the event of a fire detection, the Security_System shall Unlock the Fire_Escape_Doors.

Behavior driven requirements:

If the Computed_Airspeed Fault_Flag is set, [then] the Control_System shall use Modelled_Airspeed.

State-driven requirements**:**

While the Aircraft is In-flight, the Control_System shall Maintain Engine_Fuel_Flow at greater than XX lbs/sec.

Qualification examples:

When in the 'ON' state, the System shall display the Time, without obscuring the Work_Space.

Characteristics that are established by this rule:

C3 - Unambiguous

C4 - Complete

C7 - Verifiable

C8 - Correct

C9 - Conforming

4.1.2 R2 - /ACCURACY/USEACTIVEVOICE

Use the active voice in the main sentence structure of the need 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. This 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 verify the SOI meets the requirement. Including the entity in the subject also helps ensure the requirement

refers to the appropriate level consistent with the entity name (see R3). If the entity name is not clearly identified as the subject of the sentence, the need or requirement is not Completed (C4).

Often when the phrase “shall be” is used, the statement is in the passive voice.

Examples:

Unacceptable: The Identity of the Customer shall be confirmed.

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

Acceptable: 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.]

Unacceptable: The Audio shall be recorded by the System.

[This is unacceptable because the entity that is responsible/accountable for recording the audio is at the end of the sentence rather than the beginning.]

Unacceptable: The Audio shall be recorded.

[This is unacceptable because the entity that is responsible/accountable for recording the audio is not stated.]

Acceptable: The System shall record the Audio_Feed.

[Note that “Audio_Feed” must be defined in the glossary and required performance and conditional clauses added for the requirement to be complete.]

Characteristics that are established by this rule:

C2 - Appropriate

C3 - Unambiguous

C4 - Complete

C7 - Verifiable

4.1.3 R3 - /ACCURACY/SUBJECTVERB

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

Elaboration:

Subject

The subject of a need or requirement statement must be appropriate to the entity to which it *refers*.

Requirements referring to the business management level therefore have the form “The <business> shall …”; those referring to the business operations level have the form “The < operations> shall …”; those referring to the system level have the form "The <system> shall …"; requirements referring to the subsystem level have the form "The <subsystem> shall ..."; and requirements referring to the component level have the form "The <component> shall ...".

As a general rule, sets of need and 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 system requirements would contain only needs or requirements that apply to the integrated system, a set subsystem requirements would contain only needs or requirements that apply to that subsystem, a set 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 be 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 the need or requirement applies (should there be a good 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.

Consequently, regardless of the entity to which a need or requirement applies, the subject of a need or requirement statement must be appropriate to the entity to which it *refers*. To continue our aircraft example above, although many requirements at the business management level 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.

Verb

Similarly, the verb of a need 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 (C7) . At the business management level, for example, the use of a verb such as “safe”, may be acceptable as long as it is unambiguous at that level, decomposed at the lower levels, and is verifiable at those levels.

Examples:

Subject examples*:*

Business management requirements have the form “The <business> shall …”—for example, “ACME_Transport shall …”.

Business operations requirements on personnel roles have the form: “The <personnel role> shall …”—for example, “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 <system> 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 …”; “The Landing_Gear shall …”.

Verb examples*:*

System level stakeholder need: “The stakeholders need the system to process data received from [other system] XYZ.”

System level requirement: “The system shall [process] data having the characteristics defined in [Other system that supplies the data] 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 level at which these sub functions are to be stated. If more than one subsystem is involved in any one of the sub functions, that requirement should be communicated at the system level and allocated to the applicable subsystems. If a sub function is to be implemented by a single subsystem, then the sub function requirement should be communicated at the subsystem level and traced back to the parent requirement from which it was decomposed.

Unacceptable system requirement: “The User shall ……” [

[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 user need into a system requirement. Ask, what does the system have to do so that the user need can be achieved?]

Acceptable*:* “The <system> shall …….”

Characteristics that are established by this rule:
C2 - Appropriate
C3 - Unambiguous
C7 - Verifiable
C10 - Complete
C14 - Able to be Validated

4.1.4 R4 - /ACCURACY/USEDEFINEDTERMS

Define terms.

Elaboration:

Most languages are rich with words having several synonyms, each with a subtly different meaning based on context. In need 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 or requirement statement to know exactly what the writer meant 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 should be agreed upon to make the use of glossary terms identifiable in the need and requirements text statements; for example, glossary items may be capitalized and multiple words

in single terms joined by an underscore (e.g., “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 section. This standard should be implemented and enforced within all SE tools used by the project to help ensure consistency.

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

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 system shall display the current time.

[This is unacceptable because it is ambiguous--what is “current”, in which time zone, to what degree of accuracy, in what format?]

Acceptable: The system shall display the Current_Time in "DD/MM/YYYY" format.

[Note that “Current_Time” must then be defined in the glossary in terms of accuracy, format, time zone, and units.]

Characteristics that are established by this rule:

C3 - Unambiguous

C7 - Verifiable

C11 - Consistent

C13 - Comprehensible

C14 - Able to be Validated

4.1.5 R5 - /ACCURACY/USEDEFINITEARTICLES

Use definite article “the” rather than the indefinite article “a.”

Elaboration:

The definite article is “*the*”; the indefinite article is “*a*.” When referring to entities, use of the indefinite article can lead to ambiguity. For example, if the need or requirement refers to “a user” it is unclear whether it means any user or one of the defined users for which the system has been designed. This causes further confusion during verification and validation activities across the lifecycle—for example, babies are arguably users of baby food, but the system would fail if the test agency sought to verify or validate that a baby could order, receive, open, and serve (or even independently consume) baby food. 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.

Examples:

Unacceptable: The system shall provide a time display.

[This is unacceptable for a requirement because it is ambiguous--will any time display 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, 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.

As a stakeholder need statement: "The stakeholders need the system to provide a time display." The statement is acceptable. As part of the transformation process the requirement writers will remove the ambiguity resulting in the following requirement statement.]

Acceptable*:* The system shall display the Current_Time.

[Note that "Current_Time" must be defined in the glossary since there are a number of possible meanings and formats of the more-general term "current time." There may also be other requirements addressing where time needs to be displayed, accuracy, etc. if not addressed in the glossary or data dictionary.]

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.

Characteristics that are established by this rule:

C3 - Unambiguous

C7 - Verifiable

4.1.6 R6 - /ACCURACY/UNITS

Use appropriate units when stating quantities.

Elaboration:

All numbers should have units of measure explicitly stated in terms of the measurement system used or the thing the number refers.

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.

There are three primary measurement systems: British imperial, US, Metric

For temperatures, the following are used: Celsius, Fahrenheit, or Kelvin, etc.

Examples:

Unacceptable: The Circuit_Board shall … a temperature less than 30 degrees.

[This is unacceptable because the units used are stated without indicating the specific measurement system used.]

Acceptable: The Circuit_Board shall … a temperature of less than 30 degrees Celsius.

Unacceptable: The system shall establish communications with at least 4 in less than or equal to 10 seconds.

[This is unacceptable because the units used are incomplete. "4" of what?]

Acceptable: The system shall establish communications with at least 4 satellites in less than or equal to 10 seconds.

[Note that the term "communications" is acceptable at the business level but would need further elaboration at lower levels so that the requirement is decomposed and verifiable. E.g., frequency, type of communications (voice? data?), quality, bandwidth, etc.]

Characteristics that are established by this rule:

C3 - Unambiguous

C4 - Complete

C7 - Verifiable

C8 - Correct

4.1.7 R7 - /ACCURACY/AVOIDVAGUETERMS

Avoid the use of vague terms.

Elaboration:

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."

Vague adjectives can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations.

Adverbs qualify actions in some way and are particularly troublesome. Avoid vague adverbs, such as "usually", "approximately", "sufficiently", and "typically".

Vague adverbs can lead to ambiguous, unverifiable requirements that do not reflect accurately the stakeholder expectations.

Because the true intent is not being communicated, the need or requirement is not complete (C4).

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

Examples:

Unacceptable: The Flight_Information_System shall usually be online.

[This is unacceptable because "usually" is ambiguous - is availability what is meant?]

Acceptable: The Flight_Information_System 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 since there are a number of possible ways of calculating that measure.]

Unacceptable: The Flight_Information_System shall display the Tracking_Information for relevant aircraft.

[This is unacceptable because it does not make explicit which aircraft are relevant. Additionally, the statement allows the developer to decide what is relevant; such decisions are in the province of the customer, who should make the requirement explicit.]

Acceptable: The Flight_Information_System shall display the Tracking_Information of each Aircraft located less than or equal to 20 kilometers from the Airfield.

[Now it is clear for which aircraft the information needs to be displayed. Note that "Aircraft", "Tracking_Information", and "Airfield" must be defined in the glossary.]

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.

Characteristics that are established by this rule:

C3 - Unambiguous

C4 - Complete

C7 - Verifiable

4.1.8 R8 - /ACCURACY/NOESCAPECLAUSES

Avoid escape clauses.

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.

Such clauses provide vague conditions or possibilities, using phrases 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."

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 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 from which they were transformed.

Examples:

Unacceptable: The GPS shall, *where there is sufficient space*, display the User_Location.

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

Acceptable: The GPS shall display the User_Location.

[Note that "GPS" and "User_Location" must be defined in the glossary. Specific performance requirements need to be defined as well such as within what time, format, and accuracy.]

Unacceptable: The Television shall, *to the extent necessary*, conform to the requirements in Section a.1 of Standard XYZ.

[This is unacceptable because it is unclear who is to determine the meaning of 'necessary'. The requirement is clearer without the escape clause.]

Acceptable: The Television shall conform to the requirements in Section a.1 of Standard XYZ.

[Note that this form of requirement means that each requirement of Standard XYZ is to be met—if only a subset of a standard or regulation is required, that subset must be either listed explicitly or implementing requirements derived that meet the intent of the specific requirements in the standard and regulation, with a trace to the source or parent requirement.] Refer to the NRM and GtNR for a more detailed discussion concerning compliance with requirements within standards and regulations.

Characteristics that are established by this rule:

C3 - Unambiguous

C7 - Verifiable

4.1.9 R9 - /ACCURACY/NOOPENENDED

Avoid open-ended clauses.

Elaboration:

Open-ended clauses imply there is more required without stating exactly what. Avoid non-specific phrases such as "including but not limited to", "etc." and "and so on."

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.

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.

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

Depending on the contract type (fixed price versus level of effort or cost plus) open-ended needs and requirement statements can lead to serious interpretation problems concerning what is in or out of scope of the contract; possibly resulting in expensive contract changes.

Examples:

Unacceptable: The ATM shall display the Customer Account_Number, Account_Balance, and so on.

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

Acceptable: *[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.
The ATM shall display the Customer Account_Balance.
The ATM shall display the Customer Account_Type.
The ATM shall display the Customer Account_Overdraft_Limit.

[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.]

Characteristics that are established by this rule:

C3 - Unambiguous
C4 - Complete
C5 - Singular
C7 - Verifiable

4.2 Concision

4.2.1 R10 - /CONCISION/SUPERFLUOUSINFINITIVES

Avoid superfluous infinitives.

Elaboration:

We sometimes see a need or requirement that contains more verbs than necessary to describe a basic action, such as “The system shall be designed to be able to ...” or “The system shall be designed to be capable of ...” rather than simply “The system shall ...” Think ahead to design verification and system verification. If the system “is able to”, or “is capable of”, doing something one time but fails 99 times, the system has not met the requirement.

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 parent requirements will be allocated appropriately to the people, processes, and products (SOI), as appropriate.

When the resulting requirement sets are implemented for all three areas, the capability will exist to meet the need.

This is also true for needs at the system or system element levels. The stakeholders may have a need for the system to “be able to” or “have the capability to” do something. For example: “The stakeholders need the system to provide the capability for users to [do something] ...”. As part of the transformation process from need to requirement, the requirement writer will determine what the system needs to do to provide that capability. The resulting well-formed requirement(s) will then define what the system needs to do to supply that capability in unambiguous and verifiable language.

If the intent of using "be able to" or "be capable of" type wording is to communicate the system will only need to do the required action based on some condition, trigger, or state, then it is best to state the condition, trigger, or state as part of the requirements. When this is the case, refer to R1 and R11.

Examples:

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

[This is unacceptable because it contains the superfluous infinitive "be able to". However, this is acceptable for the stakeholder need: "The stakeholders need the Weapon_Subsystem to be able to store the location of each Ordnance."]

Acceptable: The Weapon_Subsystem shall store the Location of each Ordnance.

[Note that the terms "Weapon_Subsystem", "Location", and "Ordnance" must be defined in the glossary or data dictionary.]

Characteristics that are established by this rule:

C3 - Unambiguous

C7 - Verifiable

4.2.2 R11 - /CONCISION/SEPARATECLAUSES

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 or constraints). A single, clearly identifiable clause should be used for each condition or qualification expressed.

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 (C7), nor Correct (C8) unless the condition clause is included.

If a qualifying statement that is needed to clearly communicate the intent of the action verb on the object is not stated explicitly within the need or requirement statement, the need or requirement statement is not Complete (C4)), Verifiable (C7), nor Correct (C8) unless the 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 (i.e., and 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, Patterns.

Examples:

Unacceptable: The Navigation_Beacon shall provide Augmentation_Data at an accuracy of less than or equal to 20 meters to each Maritime_User during Harbor_Harbor_Approach_Maneuvering (HHAM).

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

Acceptable: The Navigation_Beacon shall provide Augmentation_Data to each Maritime_User engaged in Harbor_Harbor_Approach_Maneuvering (HHAM), at an accuracy of less than or equal to 20 meters.

[This rewrite places the basic function in an unbroken clause followed by the sub-clause describing performance.]

[Note that: "Navigation_Beacon", "Maritime_User", and "Harbor_Harbor_Approach_Maneuvering (HHAM)" must be defined in the glossary.]

Characteristics that are established by this rule:
C3 - Unambiguous
C4 - Complete
C7 - Verifiable
C8 - Correct

4.3 Non-ambiguity

4.3.1 R12 - /NONAMBIGUITY/CORRECTGRAMMAR

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 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 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 needs and requirements 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 and requirement statements must be translated. An interesting exercise is to translate a requirement from one language to another and translate the result back to the original language.

See also R4 – Define terms.

Examples:

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

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

Acceptable: The Weapon_Subsystem 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.]

Unacceptable: When in the Active_State, the Record_Subsystem shall display each of the Names of the Line_Items, without obscuring the User_ID.

[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.]

Acceptable: When in the Active_State, the Record_Subsystem shall display the Name of each Line_Item, without obscuring the User_ID.

Characteristics that are established by this rule:
C3 - Unambiguous
C7 - Verifiable
C8 - Correct C9 - Conforming

4.3.2 R13 - /NONAMBIGUITY/CORRECTSPELLING

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" or "ordinance" versus "ordnance." 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" which has the 2 different meanings for "clear". *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 (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 design input requirements.
- Capital letters in other non-acronyms concepts: avoid "Requirements Working Group" and "Requirements working group" in the same set of design input 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.

Examples:

Unacceptable: The Weapon_Subsystem shall store the location of all *ordinance*.

[This is unacceptable because the word "ordinance" means regulation or law. It is unlikely that the Weapon_Subsystem 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".]

Acceptable: The Weapon_Subsystem shall store the Location of all Ordnance.

[Note that "Location" and "Ordnance" must be defined in the glossary to be explicit about the types of weapons and ammunition.]

Characteristics that are established by this rule:
C3 - Unambiguous
C7 - Verifiable
C8 - Correct

4.3.3 R14 - /NONAMBIGUITY/CORRECTPUNCTUATION

Use correct punctuation.

Elaboration:
Incorrect punctuation can cause confusion between sub-clauses in a need or requirement statement. A requirement with incorrect punctuation is not Correct (C8) Note also that the more punctuation in the need or requirement statement, the greater the opportunity for ambiguity.

Examples:
Unacceptable: The Navigation_Beacon shall provide Augmentation_Data to each Maritime_User, engaged in Harbor_Harbor_Approach_Maneuvering (HHA) at an accuracy of 20 meters or less at least 99.7% of the time. [This is unacceptable because the incorrectly placed comma in this sentence confuses the meaning, leading the reader to believe that the accuracy is related to the maneuver rather than to the augmentation data.] Acceptable: The Navigation_Beacon shall provide Augmentation_Data to each Maritime_User engaged in Harbor_Harbor_Approach_Maneuvering (HHA), at an accuracy of less than or equal to 20 meters greater than 99.7% of the time. [The positioning of the comma now makes it clear that the accuracy and availability relate to the data.]

Characteristics that are established by this rule:
C3 - Unambiguous
C8 - Correct

4.3.4 R15 - /NONAMBIGUITY/LOGICALCONDITION

Use a defined convention to express logical expressions.

Elaboration:
Define a convention for logical expressions such as “[X AND Y]”, “[X OR Y]”, [X XOR Y]”, “NOT [X OR Y]”. 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", or similar wording.

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

The use of logical expressions within design input requirements 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 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].] Acceptable: The Engine_Management_System shall disengage the Speed_Control_Subsystem, when [the Cruise_Control is engaged AND the Driver applies the Accelerator].

Exceptions and relationships:
While R21: /Singularity/AvoidParentheses 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.

Characteristics that are established by this rule:
C3 – Unambiguous
C7 - Verifiable

4.3.5 R16 - /NONAMBIGUITY/AVOIDNOT

Avoid the use of "not."

Elaboration:
The presence of "not" in a need or requirement statement implies "not ever", which is impossible to verify in a finite time. In cases like this, the need or requirement statement is not correct (C8). Rewriting the need or requirement statement to avoid the use of "not" results in a need or requirement statement that is clearer and is verifiable or able to be validated.

Examples:
Unacceptable: The <system> shall not fail. [This is unacceptable because verification of the requirement would require infinite time.] *Acceptable:* The <system> shall have an Availability of greater than or equal to 95%. Or The <system> shall have a Mean Time Between Failures (MTBF) of 6 months. Unacceptable: The <system> 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.] *Acceptable:* The <system> shall limit metallic mercury exposures to workers to less than or equal 0.025 mg/m^3 over an 8-hour workday.

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 Rule 15, 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 <system> 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).

Characteristics that are established by this rule:
C3 - Unambiguous C7 - Verifiable C8 - Correct

4.3.6 R17 - /NONAMBIGUITY/OBLIQUE

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 (e.g., “user/operator”, “budget/schedule” or the construct “and/or” discussed in R15) can lead to ambiguous need and requirement statements that do not reflect accurately the true customer needs or lifecycle concepts from which the needs were derived. An exception to this rule is where the oblique symbol is used in SI units (for example “km/h”) or when communicating a symmetrical range of a value (for example +/- 5 degrees F.) Also see R19.

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: open, close, or both?] Acceptable: (Split into two requirements) The User_Management_System shall Open the User_Account in less than 1 second. The User_Management_System shall Close the User_Account in less than 1 second. Unacceptable: The Engine_Management_System shall disengage the Speed_Control_Subsystem when the Clutch is disengaged and/or the Driver applies the Brake. [This is unacceptable because of the use of “and/or.” If “and” is meant, split the two thoughts into separate requirements. If “or” is meant, write the requirement as an exclusive “or.”] Acceptable: (Split into two requirements) The Engine_Management_System shall disengage the Speed_Control_Subsystem when the Clutch is disengaged. The Engine_Management_System shall disengage the Speed_Control_Subsystem when the Driver is applying the Brake. Acceptable: (as one requirement if exclusive or is intended) The Engine_Management_System shall disengage the Speed_Control_Subsystem when [the Clutch is disengaged OR the Driver is applying the Brake].

Characteristics that are established by this rule:
C3 - Unambiguous C7 - Verifiable

4.4 Singularity

4.4.1 R18 - /SINGULARITY/SINGLESENTENCE

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

Elaboration:

Based on the concepts of allocation, traceability, validation, and verification, need and requirement statements must contain a single thought allowing needs to be traced to their source and 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, and allowing design and system validation and design and system verification against the single thought need or requirement.

Sometimes a need or requirement statement is only applicable under a specific condition or multiple conditions. See R1, R11, and R27.

If multiple actions are needed for a single condition, each action should be repeated in the text of a separate need 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 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.

Compound sentences are to be avoided that contain more than subject/verb/object.

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 this information in the attribute A1 - *Rationale* as part of the requirement expression as discussed in the NRM and include the condition of use within the need or requirement statement.

See also R1, R11, R22.

Examples:

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

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

Acceptable: (Split into two separate requirements)

When in the Active_State, the Record_Subsystem shall display the Name of each Line_Item, without obscuring the User_ID.

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

[Note use of the glossary to define terms.]

Unacceptable: The Control_Subsystem will close the Inlet_Valve until the temperature has reduced to 85 °C, when it will reopen it.

[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'.]

Acceptable: (Split into two requirements)

If the temperature of water in the Boiler is greater than 85 °C, the Control_Subsystem shall close the Inlet_Valve in less than 3 seconds

When the temperature of water in the Boiler is reduced to less than or equal to 85 °C., the Control_Subsystem shall open the Inlet_Valve in less than 3 seconds.

[Note use of the glossary to define terms.]

Unacceptable:

In the event of a fire detection:

- Power to electromagnetic magnetic fire door catches shall be turned off.
- 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 requirement statement. For multiple conditions for a single action see R28.]

Acceptable: (Split into three requirements)

In the event of a fire detection, the Security_System shall turn off power to electromagnetic magnetic fire door catches.

In the event of a fire detection, the Security_System shall set security entrances to the Free_Access_Mode.

In the event of a fire detection, the Security_System shall unlock fire escape doors.

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 constraint "The Ambulance_Control_System shall communicate Incident_Details to the Driver *while simultaneously maintaining communication with the Caller."*

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.

Characteristics that are established by this rule:

C3 - Unambiguous

C5 - Singular

C7 - Verifiable

C9 - Conforming

C13 - Comprehensible

4.4.2 R19 - /SINGULARITY/AVOIDCOMBINATORS

Avoid combinators.

Elaboration:

Combinators are words that join clauses, such as “and”, “or”, “then”, “unless”, “but”, “as well as” “but also”, “however”, “whether”, “meanwhile”, “whereas”, “on the other hand”, or “otherwise.” Their presence in a requirement usually indicates that multiple requirements should be written.

Exception: 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.

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.]

Acceptable: The Security_System shall categorize each User as [EITHER Trusted OR Not_Trusted).

Unacceptable: *The “command the sidelights” function shall* ensure the regulatory consistency of lights illumination *by ensuring* the illumination of sidelights is maintained during the illumination of the side lamps, or the head lights, or the fog lights or the rear fog lights or a combination of these lights.

[Again, this is unacceptable because it is non-singular, uses dubious grammar, contains elements of purpose, and is generally confusing.]

Acceptable: The Control_Sidelights function shall illuminate the Sidelights while any combination of the following conditions is true: the Side_Lamps are illuminated, the Head_Lamps are illuminated, the Front_Fog_Lamps are illuminated, the Rear_Fog_Lamps are illuminated.

[Rules 16 and 17 have also been applied to remove combinators and clarify the exact conditions. Note that the use of “any” in this example is not ambiguous and thus is an exception to R32. See also R28.]

Characteristics that are established by this rule:
C3 - Unambiguous
C5 - Singular

4.4.3 R20 - /SINGULARITY/AVOIDPURPOSE

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

Elaboration:
Needs and requirement statements should be as concise as possible and be complete (C4). To avoid ambiguity or communicate why the need or requirement is necessary (C1), some writers feel the need to include additional information to make the intent and reason clearer by including additional text within the need or requirement statement. The text of a need or requirement statement does not and should not have to carry around this extra text. Expressions of purpose are often indicated by the presence of phrases such as "……to", "in order to", "so that", and "thus allowing." This extra information should be included separately in the need 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.] Acceptable: The Record_Subsystem shall display the Name of each Line_Item. [The text "so that the Operator can confirm that it is the correct Item" should be included in the rationale attribute.] Unacceptable: The Operation_Logger shall record each Warning_Message produced by the system. [This is unacceptable because of the superfluous words "produced by the system".] Acceptable: The Operation_Logger shall record each Warning_Message.

Characteristics that are established by this rule:
C1 - Necessary
C5 - Singular

4.4.4 R21 - /SINGULARITY/AVOIDPARENTHESES

Avoid parentheses and brackets containing subordinate text.

Elaboration:
If the text of a need or requirement statement contains parentheses or brackets, it usually indicates the presence of superfluous information that can simply be removed or 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, but such conventions should be documented at the start of the needs or requirements document.

Examples:
Unacceptable: The Control_Unit shall disconnect power from the Boiler when the water temperature exceeds 85 °C (usually towards the end of the boiling cycle.) [This is unacceptable because of the parenthetical phrase as well as the ambiguous word "usually".] Acceptable: The Control_Unit shall disconnect power from the Boiler when the water temperature is greater than 85 °C. [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.]

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.

Characteristics that are established by this rule:
C5 - Singular

4.4.5 R22 - /SINGULARITY/ENUMERATION

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

Elaboration:
If a number of functions are implied, a need 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. The specific functions should be enumerated explicitly in separate requirement statements.]

Acceptable: (three separate requirements*):*

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

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

The Thermal_Control_System shall store the history of Current_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 HR functions centrally.” or a system level need statement may state: “The stakeholders need the system 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, the during the transformation of the need statement into design input 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 design input requirements, when realized, would meet the intent of the need statement.

Characteristics that are established by this rule:

C3 – Unambiguous

C5 – Singular

4.4.6 R23 - /SINGULARITY/CONTEXT

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

Elaboration:
Sometimes it can be difficult to express a complicated need or requirement in words, and it is better simply to refer to a 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 system 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.

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 expressed in the form of a diagram.] Acceptable: When the Product temperature exceeds 95 °C, the Control_System shall close Input_Valves as specified in timing diagram 6. [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 below the line as a design output. The design input requirement should focus on why this behavior is needed.]

Characteristics that are established by this rule:
C3 - Unambiguous C4 - Complete *C5 - Singular*

4.5 Completeness

4.5.1 R24 - /COMPLETENESS/AVOIDPRONOUNS

Avoid the use of pronouns and indefinite pronouns.

Elaboration:
Repeat nouns in full instead of using pronouns to refer to nouns in other need or requirement statements. Pronouns are words such as "it", "this", "that", "he", "she", "they", and "them."

When writing stories, pronouns are a useful device for avoiding the repetition of nouns; but when writing need and requirement statements, pronouns are effectively cross-references to nouns in other need 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 in the previous need or requirement statement, however, as the set of requirements mature, individual requirements may be added, deleted, reordered, or regrouped, and the defining requirement is no longer nearby. 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.”]

Acceptable: The Controller shall send the Driver_Itinerary for the day to the Driver at least 8 hours prior to the Driver_Shift.

[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.]

Characteristics that are established by this rule:

C3 - Unambiguous

C4 - Complete

C7 - Verifiable

4.5.2 R25 - /COMPLETENESS/USEOFHEADINGS

Avoid relying on headings to support explanation or understanding of the 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 requirement applies to contribute to the explanation of the need or requirement statement. The need or requirement statement should be complete in and of itself and not require the heading for the intent of the need or requirement 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 requirements management tool (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 system 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]
Acceptable: The system shall activate the Alert_Buzzer for greater than 20 minutes.

Exceptions and relationships:
The use of headings is acceptable when producing an output of a RMT, grouping like types or categories of needs or requirements. See R40 and R41.

Characteristics that are established by this rule:
C4 - Complete

4.6 Realism

4.6.1 R26 - /REALISM/AVOIDABSOLUTES

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 would you prove 100% availability? Even if you could build such a system, could you afford to verify or validate it?
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 (C7), nor Correct (C8).

Examples:
Unacceptable: The system 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?]
Acceptable: The system shall have an Availability of greater than or equal to 98% during Operating_Hours.
[Note use of the glossary to define Operating_Hours.]

Characteristics that are established by this rule:
C6 – Feasible
C7 - Verifiable
C8 – Correct

4.7 Conditions

4.7.1 R27 - /CONDITIONS/EXPLICIT

State applicability conditions explicitly.

Elaboration:

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

Sometimes need or requirement statements are only applicable under certain conditions. If so, the condition should be repeated in the text of each need or requirement statement, rather than stating the condition and then listing the actions to be taken.

There are several agreed to types of condition:

Event-driven requirements:

When <optional preconditions/trigger> the <system name> shall <system response>.

In the event of <specific event> the <system name> shall <system response>.

Behavior driven requirements:

If <optional preconditions/trigger>, then the <system name> shall <system response>.

State-driven requirements:

While <entering, exiting or in a specific state (or mode)> the <system name> shall <system response>.

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 (C7), nor Correct (C8) unless the condition clause is included.

See also R1, R11, R18.

Examples:

Unacceptable:

In the event of a fire detection:

- Power to electromagnetic magnetic fire door catches shall be turned off.
- 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. Also, note the actions are written in passive voice which violates R2.]

Acceptable: (Split into three separate requirements.)

In the event of a fire detection, the Security_System shall turn off power to electromagnetic magnetic fire door catches.

In the event of a fire detection, the Security_System shall set security entrances to the Free_Access_Mode.

In the event of a fire detection, the Security_System shall unlock fire escape doors.

Characteristics that are established by this rule:
C4 - Complete C7 - Verifiable C8 - Correct

4.7.2 R28 - /CONDITIONS/EXPLICITLISTS

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

Elaboration:
When a list of conditions is given for a single action in a single need or 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 need or requirement should make this clear. When the combination of conditions and actions that trigger another action is complex, consideration should be given to the use of a diagram or table (see R23).

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.] Acceptable if interpreted as a disjunction: The Audit_System shall allow the Audit_Clerk to change the status of the action item when one or more of the following conditions are true: - the Audit_Clerk originated the item; - the Audit_Clerk is the actionee; - the Audit_Clerk is the reviewer. Alternately, this could be stated as three distinct requirements as this makes each atomic and overall has the same meaning. From an allocation, traceability, and verifiability perspective this form is preferable. The Audit_System shall allow the Audit_Clerk to change the status of the action item when the Audit_Clerk originated the item. The Audit_System shall allow the Audit_Clerk to change the status of the action item when the Audit_Clerk is the actionee. The Audit_System shall allow the Audit_Clerk to change the status of the action item when the Audit_Clerk is the reviewer.

Acceptable if interpreted as a conjunction: The Audit_System shall allow the Audit_Clerk to change the status of the action item when the following conditions are true: - the Audit_Clerk originated the 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 be true for the logical condition to be true.]

Characteristics that are established by this rule:
C3 - Unambiguous
C7 - Verifiable

4.8 Uniqueness

4.8.1 R29 - /UNIQUENESS/CLASSIFY

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 sets of needs or requirements. It is useful to assign to each need 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 by asking to be shown all the maintenance requirements, engineers developing test plans could 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 duplication, conflicts, or absence of requirements. 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 R40 - /Modularity/RelatedRequirements

Characteristics that are established by this rule:
C10 - Complete C11 - Consistent

4.8.2 R30 - /UNIQUENESS/EXPRESSONCE

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).

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 system shall generate a report of financial transactions containing the information defined in <some standard or contract deliverable listing>" and "The system shall generate a financial report." are overlapping in that the first is a subset of the second. Avoidance of duplication can be aided by classification (R29) so a subset of needs or requirements can be compared.

Characteristics that are established by this rule:
C1 - Necessary C9 - Conforming C11 - Consistent

4.9 Abstraction

4.9.1 R31 - /ABSTRACTION/SOLUTIONFREE

When defining design inputs avoid stating a solution unless there is rationale for constraining the design.

Elaboration:
Refer to Section 1.7 concerning the differences between needs and requirements. As design inputs, every system endeavor should have a level of needs and requirements for an entity that captures the problem to be solved (design inputs) without including solutions (design

outputs). System level needs and requirements should provide a system-level requirement for the overall problem 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 a solution, 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 you 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 problem being solved.
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 a solution (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 [xxxxx]."

  [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 [xxxxxxx] is feasible with acceptable risk for this lifecycle stage. As stated, this is an appropriate level of detail for a design input requirement.]

This system level design input accuracy requirement is then allocated to the key parts of the system that have a role in meeting the overall system accuracy requirement. 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 allocation and not how that accuracy 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 deign output artifacts (specifications).

Unacceptable: Traffic lights shall be used to control pedestrian 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]

Acceptable: (several requirements):

The Traffic_Control_System shall signal "Walk" when the Pedestrian is permitted to cross the road at the Intersection.

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

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

Unacceptable: By pressing a button on the traffic-light pillar, the pedestrian signals his presence, and the light turns 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 the subject is the user over which we have no control—that is, we cannot tell the user what to do in a requirement statement for the system being developed.]

Acceptable: The Traffic_Control_System shall allow the Pedestrian to signal intent to cross the road at the Intersection.

[This requirement allows freedom in determining the best solution (design input), which may be a means of automatic detection rather than button pushing.]

[Note that glossary definitions should be used for "Pedestrian", "Traffic_Control_System", 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 a commercial off the shelf (COTS) 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 is when an existing system or COTS 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 COTS. 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 COTS. Refer to the NRM for a detailed discussion the use of OTC and COTS entities.
4. As part of the lifecycle concept analysis and maturation activities a prototype was developed 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.

Characteristics that are established by this rule:
C2 - Appropriate

4.10 Quantifiers

4.10.1 R32 - /QUANTIFIERS/UNIVERSALS

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

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

Examples:
Unacceptable: The Operation_Logger shall record any *(or all)* warning messages. [This is unacceptable because of the use of the word "any", which is then made worse by the addition of "(or all)".] Acceptable: The Operation_Logger shall record each Warning_Message. [Note that Warning_Message must be defined so that it is clear that the system only will record each defined Warning Message.] Unacceptable: The Record_Subsystem shall display the Names of both of the Line_Items. [This is unacceptable because of the use of the word "both".] Acceptable: The Record_Subsystem shall display the Name of each Line_Item. *Acceptable*: The Record_Subsystem shall display each Line_Item_Name.

Characteristics that are established by this rule:
C3 - Unambiguous C7 - Verifiable C8 - Correct

4.11 Tolerance

4.11.1 R33 - /TOLERANCE/VALUERANGE

Define quantities with a range of values appropriate to the entity to which the apply and to which the entity will be verified or validated against.

Elaboration:

When it comes to defining performance, single-point values are seldom sufficient and are difficult to test. Ask the question: "if the performance was a little less (or more) than this, would I still buy it?" 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 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 very 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.

Examples:

Unacceptable: The Pumping_Station shall maintain the flow of water at 120 liters per second for 30 minutes.

[This is unacceptable because we do not know whether a solution that addresses more or less than the specified quantities is acceptable.]

Acceptable: The Pumping_Station shall maintain a flow of water at 120 ±10 liters per second for at least 30 minutes.

[Now the range of acceptable flow performance is clear and that the 30 minutes is a minimum acceptable performance.]

Unacceptable: The Flight_Information_System shall display the current altitude to approximately 1 meter resolution.

[This is unacceptable because it is imprecise. What is "approximately" in the context of a distance of 1 meter? Who has the option of deciding what is "approximately"? How will "approximately" be verified? What is the acceptable tolerance? Further, altitude is more commonly described in units of feet, not meters, so is +3 feet an acceptable equivalent to 1 meter since 3 feet is less than 1 meter?]

[Note that care must be taken to confirm both units and transformation from one set of units to another set to ensure accuracy, acceptability and consistency. See also R6.]

Acceptable: The Flight_Information_System shall display Current_Altitude with an accuracy of ±3 feet.

[Note that "Current_Altitude" must be defined in the glossary since there are a number of possible interpretations of the term.]

Unacceptable: The System 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 System shall limit arsenic contamination in the drinking water to 1 part per trillion. Rationale: 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.]

Acceptable: The System shall limit arsenic contamination in the drinking water to less than10 parts per billion. Rationale: 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.

Characteristics that are established by this rule:
C3 - Unambiguous
C4 - Complete
C6 - Feasible
C7 - Verifiable
C8 - Correct
C12 - Feasible

4.12 Quantification

4.12.1 R34 - /QUANTIFICATION/MEASURABLE

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

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

Examples:
Unacceptable: The system shall use minimum power. [This is unacceptable because “minimum” is ambiguous.] Acceptable: The system shall use 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.] *Acceptable:* The Engine shall achieve an emissions level that is at least xxx. [where xxx represents the required threshold value.]

Exceptions and relationships:
Some quantification such as “minimum”, “maximum”, “optimal” is 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.

Characteristics that are established by this rule:
C3 - Unambiguous C4 - Complete C7 - Verifiable C12 - Feasible

4.12.2 R35 - /QUANTIFICATION/TEMPORALINDEFINITE

Define temporal dependencies explicitly instead of using indefinite temporal keywords.

Elaboration:
Some words and phrases signal non-specific timing, such as “eventually”, “until”, “before”, “after”, “as”, “once”, “earliest”, “latest”, “instantaneous”, “simultaneous”, and “at last” are ambiguous and not verifiable. Indefinite temporal keywords can cause confusion or unintended meaning. These words should be replaced by specific timing constraints.

Examples:
Unacceptable: Continual operation of the pump shall eventually result in the tank being empty. [This is unacceptable because “eventually” is ambiguous. Also, this statement is not written in the proper form. It is written on “operation” rather than on a requirement on the pump which violates R3.] Acceptable: The Pump shall remove greater than 99% of the Fluid from the Tank in less than 3 days of continuous operation.

Characteristics that are established by this rule:
C3 - Unambiguous C4 - Complete C7 - Verifiable

4.13 Uniformity of Language

4.13.1 R36 - /UNIFORMLANGUAGE/USECONSISTENTTERMS

Use each term and units of measure consistently throughout need and requirement sets.

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 terms 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 this is also 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, however the ability to maintain consistency is higher priority

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

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, in a subsystem set of design input requirements, 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).

Acceptable: 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”.]

Acceptable: 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 (e.g., US - feet) and another requirement to use another unit of measure (e.g., Metric - meter).

Acceptable: 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.

Characteristics that are established by this rule:

C3 - Unambiguous

C8 - Correct

C9 - Conforming

C11 - Consistent

C13 - Comprehensible

C14 - Able to be Validated

4.13.2 R37 - /UNIFORMLANGUAGE/DEFINEACRONYMS

If acronyms are used in need and requirement statements, c.

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 of SE, in need and requirement sets that are generated and managed within an RMT, there is no guarantee that the set will be extracted in any particular order, so the old practice is not useful. Because of this, the use of acronyms should be avoided. However, if used, acronyms must be used consistently throughout not only the sets of needs and requirements, but all artifacts developed across all lifecycle stages. To address this issue, make sure acronyms are defined in a project wide acronym list or glossary. Acronyms must be written in a consistent way in terms of capitalization and periods. For example, always “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. Acceptable: 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.” Acceptable: 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.

Characteristics that are established by this rule:
C3 - Unambiguous
C9 - Conforming
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

4.13.3 R38 - /UNIFORMLANGUAGE/AVOIDABBREVIATIONS

Avoid the use of abbreviations in needs and requirement statements.

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. Acceptable: Avoid the abbreviation and use the full term every time.

Characteristics that are established by this rule:
C9 - Conforming C11 - Consistent C13 - Comprehensible C14 - Able to be Validated

4.13.4 R39 - /UNIFORMLANGUAGE/STYLEGUIDE

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

Elaboration:
The style guide provides a template and patterns for developing requirements, defines the attributes that the organization chooses to document with each requirement statement, selects the requirement patterns to be used for specific types of requirements, and defines what other information needs to 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 requirements electronically (versus in a printed document) within a requirement management tool (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 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 a requirement management tool, a standard schema should be defined as part of the need and requirement 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 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 and requirement patterns should be used for writing needs and requirements of a particular type, the selection and use of attributes, standard abbreviations and acronyms, layout and use of figures and tables, layout of needs and requirements documents or databases and, need and requirement statement numbering. Terms used throughout the sets of needs and requirements are defined in an ontology or at least a project glossary.

Characteristics that are established by this rule:
C4 - Complete
C5 - Singular C9 - Conforming C11 - Consistent C13 - Comprehensible C14 - Able to be Validated

4.14 Modularity

4.14.1 R40 - /MODULARITY/RELATEDREQUIREMENTS

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 R41 and the NRM for more guidance on organizing needs and requirements. An example grouping could be form, fit, function, quality, and compliance. See R41. 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, or redundant requirements. See also R29.

Examples:
Requirements may be related by: - type (for example, safety requirements); - scenario (for example, requirements arising from a single scenario). See R29 and the NRM for more guidance on organizing needs and requirements.

Characteristics that are established by this rule:
C9 - Conforming C10 - Complete C11 - Consistent C13 - Comprehensible

4.14.2 R41 - /MODULARITY/STRUCTURED

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 and requirement statements, it is often easier to understand when it is placed in context with other related requirements. Whether managed within a in document or within an RMT, the ability to draw together related groups of needs or requirements is a vital tool in identifying repetition and conflict between need 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 R40

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, safety, security, 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.)

Characteristics that are established by this rule
C10 - Complete
C11 - Consistent
C13 - Comprehensible
C14 - Able to be Validated

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., etal. (2004). Requirements Completeness. Proceedings of INCOSE 2004. 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 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.

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

8 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.

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

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

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

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

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

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

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

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

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

18 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.

19 Ryan, M., Wheatcraft, L., “On the Use of the Terms Verification and Validation”, *INCOSE International Symposium IS2017*, July 2017.
20 *Simplified Technical English (STE)*, Specification (ASD-STE100), ASD Belgium, http://www.asd-ste100.org/.
21 US Code of Federal Regulations: Title 21, Part 820, Quality System Regulation
22 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.
23 Wiegers, K.E., *Software Requirements*, Redmond, WA: Microsoft Press, 2003.
24 Wiegers, K.E., *More About Software Requirements*, Redmond, WA: Microsoft Press, 2006.

Appendix B: ACRONYMS AND ABBREVIATIONS

AD²	Advancement Degree of Difficulty
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 Standards Organization
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: Requirement Patterns

C.1 Introduction to Requirement Patterns

The notion of requirement patterns (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 are shown a good example of how to form a requirement statement based on its type, the difficulty in writing a properly formed requirement was largely overcome.

The use of patterns also enables the development of NLP/AI tools or "digital assistants". These tools, along with a project ontology, can aid writers when writing need and requirement statements and helping ensure the rule and characteristics defined within this Guide are met.

"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. These place holders are generally called pattern slots. The requirement text is written per a requirement pattern that is appropriate to what is to be communicated.

For example, the following example represents a requirement statement and the associated candidate pattern. In this example, the requirement pattern has 6 pattern slots.

The Power_Supply shall have an Availability greater than or equal to 98%.

<Subsystem> <shall> <have> <a: Optional> <PHYSICAL_PROPERTY> <OPERATOR> <PERCENTAGE>

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

One common approach uses "requirement templates". The term "template" is most often used in the field of requirements management to refer to the structure and organization of a requirements document. The use of such templates helps ensure that the authors consider the complete range of concerns when organizing a set of requirement statements. 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) use "boilerplates" to refer to a grammatical structure for an individual textual requirement.

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 common natural language requirements structures representing syntactic and semantic information (restrictions and properties).

As this guide describes, there are multiple factors that contribute to the realization of high-quality requirement statements and sets of requirements. One important way to assist in achieving high quality of requirement statements and sets of requirements is by the proper definition and agreement of a set of requirement patterns with which the requirements must comply. This appendix provides additional detail on requirement patterns.

The concept of a requirement pattern is mentioned in several sections of this guide—patterns are used to ensure that requirements and requirement sets obey the rules in this guide so that they have the desired characteristics.

C.2 Benefits of Using Patterns to Form Requirement Statements

Ensuring that requirements conform to agreed-to patterns contributes to the requirement verification activity and helps to ensure the requirements effectively communicate needs. In addition, writing requirements following a set of agreed-upon patterns also makes it easier to:

- write concise, easy to read and atomic requirement statements;
- find and classify requirements within a large document;
- find missing requirements (*completeness*);
- find inconsistences within a large set of requirements;
- find duplicated requirements (and finding and reusing requirements in general); and
- follow with other activities such as analysis and implementation.

In addition, following a set of agreed-upon patterns also leads to a consistent set of requirements, and the automatization of several requirement management activities such as:

- the semi-automatic requirement verification that requirement statements have the characteristics and conform to the rules defined in this guide;
- the verification of other consistency and completeness rules associated with systems engineering artifacts generated across all system lifecycles (e.g., consistency of requirements versus design as stated in a models and consistent use of terms defined in the project ontology);
- the extraction of entities, properties, etc. from requirements or unstructured documents; and
- the translation of requirements to different languages.

Finally, requirement patterns can also facilitate the process of properly teaching a good and common way of writing requirements and leverage the use of consistent vocabulary.

C.3 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 the structure of sub-patterns used as building blocks that may be in a catalog of patterns.

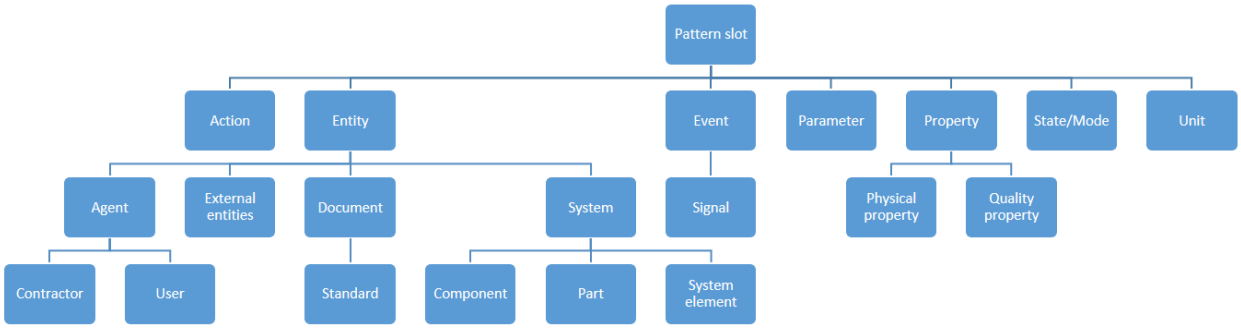


Figure C-1: Requirement pattern building blocks

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.

Other sources of requirement patterns 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/
- The PABRE Catalog: http://www.upc.edu/gessi/PABRE/index.html
- ARTEMIS CRYSTAL EU Research Project: http://www.crystal-artemis.eu/

Appendix D: Cross Reference Matrices

Rules to Characteristics Cross Reference Matrix

			Characteristis for Indivdual needs and requirements									Characterisits for Sets of needs requirements				
			Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be validated
Quality Focus	**Rule**	**Subject**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**
Accuracy	**R1**	Structured, complete sentence			X	X			X	X	X					
	R2	Use active voice		X	X	X			X							
	R3	Use appropriate subject verb		X	X				X			X				X
	R4	Define terms			X				X				X		X	X
	R5	Use definite article "the" vs "a"			X				X							
	R6	Use appropriate units			X	X			X	X						
	R7	Avoid vague terms			X	X			X							
	R8	Avoid escape clauses			X				X							
	R9	Avoid open-ended clauses			X	X	X		X							
Concision	**R10**	Avoid superfluous infinitives			X				X							
	R11	Use a separate clause			X	X			X	X						
Non-ambiguity	**R12**	Use correct grammar			X				X	X	X					
	R13	Use correct spelling			X				X							
	R14	Use correct punctuation			X					X						
	R15	Logical expressions			X				X							
	R16	Avoid the use of “not.”			X				X	X						
	R17	Avoid the oblique ("/") symbol			X				X							
Singularity	**R18**	Use sngle thought sentence			X		X		X		X				X	
	R19	Avoid combinators			X		X									
	R20	Avoid phrases of purpose or reason	X				X									
	R21	Avoid parentheses & brackets					X									
	R22	Enumerate sets explicitly			X		X									
	R23	Supporting diagram or model			X	X	X									
Completeness	**R24**	Avoid pronouns & indefinite pronouns			X	X			X							
	R25	Avoid relying on headings				X										
Realism	**R26**	Avoid using unachievable absolutes						X	X	X				X		
Conditions	**R27**	State applicability conditions explicitly				X			X	X						
	R28	Single condition for a specific action			X				X							
Uniqueness	**R29**	Classifyi by type or category										X	X			
	R30	Express once and only once	X								X		X			
Abstraction	**R31**	Avoid stating a solution		X												
Quantifiers	**R32**	Use “each” for universal quantification			X				X	X						
Tolerance	**R33**	Define quantities with a range of values			X	X		X	X	X				X		
Quantification	**R34**	Specific measurable performance targets			X	X			X					X		
	R35	Define temporal dependencies explicitly			X	X			X							
Uniformity of Language	**R36**	Use terms & units of measure consistently			X					X	X		X		X	X
	R37	Use a consistent set of acronyms			X						X		X		X	X
	R38	Avoid the use of abbreviations									X		X		X	X
	R39	Use a project-wide style guide				X	X				X		X		X	X
Modularity	**R40**	Group related needs & requirements				X					X	X	X		X	
	R41	Conform to a defined structure or template										X	X		X	X

NRM Concepts and Activities to Characteristics Cross Reference Matrix Part 1

		Characteristis for Indivdual needs and requirements									Characterisits for Sets of needs requirements				
		Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be validated
NRM Concepts and Activities		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14
SECTION 3: INFORMATION-BASED NEEDS AND															
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 ae Derived	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
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
4.4.4	Completeness										X				
4.5	Lifecycle Concepts Analysis and Maturation	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			
4.5.4	Levels of Detail and Abstraction		X												
4.5.7.1	Model Development, Analysis, and Maturation	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						
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
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

NRM Concepts and Activities to Characteristics Cross Reference Matrix Part 2

		Characteristis for Indivdual needs and requirements									Characterisits for Sets of needs requirements				
		Necessary	Appropriate	Unambiguous	Complete	Singular	Feasible	Verifiable	Correct	Conforming	Complete	Consistent	Feasible	Comprehensible	Able to be validated
NRM Concepts and Activities		**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**C10**	**C11**	**C12**	**C13**	**C14**
SECTION 6: DESIGN INPUT REQUIREMENTS DEFINITION															
6.2	Perform Design Input Requirements Definition	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				
6.2.1.4	Appropriate to Level		X												
6.2.1.5	Managing Unknowns			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
6.2.5	Plan for System Verification							X							
6.2.6.2	Completeness, Correctness, and Consistency								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
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
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
SECTION 8: DESIGN VERIFICATION AND DESIGN VALIDATION															
8.1	Design Definition Process Overview			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
8.4	Design Verification			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
14.2.4	Managing Unknowns			X	X		X	X	X			X			
14.2.7	Combine Allocation and Traceability to Manage Requirements	X							X		X				X
14.2.8	Managing Interfaces										X	X			X
14.2.9	Managing System Verification and System Validation							X							X

Appendix E: 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 (e.g., 2.1.1, no alpha)	Specific reference (e.g., 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