4.3.2.1 Requirements Definition and Analysis Concepts Requirements definition and analysis, like the set of SE processes, is an iterative activity in which new requirements are identified and constantly refined as the concept develops and additional details become known. The requirements are analyzed, and deficiencies and cost drivers are identified and reviewed with the customer to establish a requirements baseline for the project.

An objective of requirements analysis is to provide an understanding of the interactions between the various functions and to obtain a balanced set of requirements based on user objectives. Requirements are not developed in a vacuum. An essential part of the requirements development process is the OpsCon, the implicit design concept that accompanies it, and associated demands of relevant technology. Requirements come from a variety of sources, including the customer/users, regulations/codes, and corporate entities.

This complex process employs performance analysis, trade studies, constraint evaluation, and cost–benefit analysis. System requirements cannot be established without determining their impact (achievability) on lower-level elements. Therefore, requirements definition and analysis is an iteration and balancing process that works both "top-down" (called allocation and flowdown) and "bottom-up." Once the top-level set of system requirements has been established, it is necessary to allocate and flow them down to successively lower levels. As the allocation and flowdown process is repeated, it is essential that traceability be maintained to ensure that all system-level requirements are satisfied in the resulting design. The resulting requirements database usually contains many attributes for each requirement and is also used in verification. Although it is an objective to avoid requirements that constrain or define aspects of the system implementation, it is not always possible. There are often necessary constraints that need to be reflected. This includes the following:

- *Standards*—Identify standards required to meet quality or design considerations imposed as defined stakeholder requirements or derived to meet organization, industry, or domain requirements.
- *Utilization environments*—Identify the utilization environments and all environmental factors (natural or induced) that may affect system performance, impact human comfort or safety, or cause human error for each of the operational scenarios envisioned for system use.
- *Essential design considerations*—Identify design considerations including human systems integration (e.g., manpower, personnel, training, environment, safety, occupational health, survivability, habitability), system security requirements (e.g., information assurance, antitamper provisions), and potential environmental impact.
- *Design constraints*—Identify design constraints including physical limitations (e.g., weight, form/fit factors), manpower, personnel, and other resource constraints on operation of the system and defined interfaces with host platforms and interacting systems external to the system boundary, including supply, maintenance, and training infrastructures.

4.3.2.2 Characteristics and Attributes of Good Requirements In defining requirements, care should be exercised to ensure the requirement is appropriately crafted. The following characteristics should be considered for every requirement based on ISO/IEC/IEEE 29148, *Systems and software engineering—Life cycle processes—Requirements engineering* (2011), and the *INCOSE Guide for Writing Requirements* (INCOSE RWG, 2012):

- *Necessary*—Every requirement generates extra effort in the form of processing, maintenance, and verification. Only necessary requirements should therefore be included in specifications. Unnecessary requirements are of two varieties: (i) unnecessary specification of design, which should be left to the discretion of the designer, and (ii) a redundant requirement covered in some other combination of requirements.
- *Implementation independent*—Customer requirements may be imposed at any level they desire; however, when customer requirements specify design, it should be questioned. A proper requirement should deal with the entity being specified as a "black box" by describing what transformation is to be performed by the "box." The requirement should specify "what"

is to be done at that level, not "how" it is to be done at that level.

- *Unambiguous*—Requirements must convey what is to be done to the next level of development. Its key purpose is to communicate. Is the requirement clear and concise? Is it possible to interpret the requirement in multiple ways? Are the terms defined? Does the requirement conflict with or contradict another requirement? Each requirement statement should be written to address one and only one concept. Requirements with "and," "or," "commas," or other forms of redundancy can be difficult to verify and should be avoided as it can be difficult to ensure all personnel have a common understanding. Requirements must therefore be written with extreme care. The language used must be clear, exact, and in sufficient detail to meet all reasonable interpretations. A glossary should be used to precisely define often-used terms or terms, such as "process," that could have multiple interpretations.
- *Complete*—The stated requirement should be complete and measurable and not need further amplification. The stated requirement should provide sufficient capability or characteristics.
- *Singular*—The requirement statement should be of only one requirement and should not be a combination of requirements or more than one function or constraint.
- *Achievable*—The requirement must be technically achievable within constraints and requires advances in technology within acceptable risk. It is best if the implementing developer participate in requirements definition. The developer should have the expertise to assess the achievability of the requirements. In the case of items to be subcontracted, the expertise of potential subcontractors is very valuable in the generation of the requirements. Additionally, participation by manufacturing and customers/users can help ensure achievable requirements. When it is not possible to have the right mix of developer, subcontractor, and/or manufacturing expertise during the requirements generation, it is important to have their review at the first point possible to ensure achievability.
- *Verifiable*—Each requirement must be verified at some level by one of the four standard methods (inspection, analysis, demonstration, or test). A customer may specify, "The range shall be as long as possible." This is a valid but unverifiable requirement. This type of requirement is a signal that a trade study is needed to establish a verifiable maximum range requirement. Each verification requirement should be verifiable by a single method. A requirement requiring multiple methods to verify should be broken into multiple requirements. There is no problem with one method verifying multiple requirements; however, it indicates a potential for consolidating requirements. When the system hierarchy is properly designed, each level of specification has a corresponding level of verification during the verification stage. If element specifications are required to appropriately specify the system, element verification should be performed.
- *Conforming*—In many instances, there are applicable government, industry, and product standards, specifications, and interfaces with which compliance is required. An example might be additional requirements placed on new software developments for possible reusability. Another might be standard test interface connectors for certain product classes. In addition, the individual requirements should conform to the organization's standard template and style for writing requirements—when all requirements within the same organization have the same look and feel, each requirement is easier to write, understand, and review.

Note: ISO/IEC/IEEE 29148 uses the term "consistent" instead of "conforming," stating that the requirement must be free of conflicts with other requirements. While that is correct, a requirement cannot, in and of itself, be consistent—consistency is more correctly a characteristic of the set of requirements (as described in the following paragraph).

In addition to the characteristics of individual requirements, the characteristics of a set of requirements as a whole should be addressed to ensure that the set of requirements collectively provides for a feasible solution that meets the stakeholder intentions and constraints. These include:

- *Complete*—The set of requirements contains everything pertinent to the definition of system or system element being specified.

- *Consistent*—The set of requirements is consistent in that the requirements are not contradictory nor duplicated. Terms and abbreviations are used consistently in all requirements, in accordance with the glossary.
- *Feasible/affordable*—The set of requirements can be satisfied by a solution that is obtainable within LCC, schedule, and technical constraints.

Note: ISO/IEC/IEEE 29148 uses the term "affordable" instead of "feasible," stating that the set of requirements are feasible within the life cycle constraints of cost, schedule, technical, and regulatory. The INCOSE Guide for Writing Requirements (INCOSE RWG, 2012) includes the word "feasible," stating that "Feasible/Affordable is a more appropriate title for this characteristic of the requirement set."

- *Bounded*—The set of requirements define the required scope for the solution to meet the stakeholder needs. Consequently, all necessary requirements must be included; irrelevant requirements must be excluded.

In addition to the characteristics listed above, individual requirement statements may have a number of attributes attached to them (either as fields in a database or through relationships with other artifacts):

- *Trace to parent*—A child requirement is one that has been derived or decomposed from the parent—the achievement of all the children requirements will lead to the achievement of the parent requirement. Each of the children requirements must be able to be traced to its parent requirement (and thence to any antecedent requirement and ultimately to the system need/mission).
- *Trace to source*—Each requirement must be able to be traced to its source—this is different from tracing to a parent because it identifies where the requirement came from and/or how it was arrived at (rather than which other requirement is its parent).
- *Trace to interface definition*—The interactions between two systems are described in interface definitions that are often contained in a document that has a title such as an ICD. The interface requirements contained in each of the interacting systems will include reference to where the interaction is defined. This attribute provides a link trace between any of the interface requirements to where the interaction is defined.
- *Trace to peer requirements*—This attribute links requirements that are related to each other (other than parent/child) at the same level. Peer requirements may be related for a number of reasons, such as the following: they may be in conflict, or codependent, or bundled, or a complimentary interface requirement of another system to which the system has an interface.
- *Trace to verification method*—This could be a simple statement of the way in which the requirement is to be verified (inspection, demonstration, test, analysis, simulation), or it could be a more elaborate statement that effectively provides the outline of an appropriate test plan.
- *Trace to verification requirement(s)*—The verification method for each requirement simply states the planned method of verification (inspection, demonstration, test, analysis, simulation). In addition to stating the verification method, some organizations write a set of verification requirements in addition to the system requirements. This is beneficial as it forces the systems engineer to consider how each requirement is to be verified and in doing so helps identify and remove requirements that are not verifiable.
- *Trace to verification results*—The results of each verification will most often be contained in a separate document. This attribute traces each requirement to the associated verification results.
- *Requirements verification status*—It is useful to include an attribute that indicates whether the requirement has been verified or not.
- *Requirements validation status*—Requirements validation as the process of ensuring requirements and sets of requirements meet the rules and characteristics in the guide. Some organizations include an attribute field "requirement validated" to indicate whether the individual requirement has been validated.
- *Priority*—This is how important the requirement is to the stakeholder. It may not be a critical requirement (i.e., one the system must possess or it won't work at all), but simply something that the stakeholder(s)

holds very dear. Priority may be characterized in terms of a level (1, 2, 3 or high, medium, low). Priority may be inherited from a parent requirement.

- *Criticality*—A critical requirement is one that the system must achieve or the system cannot function at all—perhaps can be viewed as one of the set of minimum essential requirements. Criticality may be characterized in terms of a level (1, 2, 3 or major, medium, minor). Criticality may be inherited from a parent requirement.
- *Risk*—This is the risk that the requirement cannot be achieved within technology, schedule, and budget. For example, the requirement may be possible technically (i.e., the requirement may be feasible) but have risk of achievement within available budget and schedule. Risk may be characterized in terms of a level (e.g., high, medium, low). Risk may be inherited from a parent requirement.
- *Key driving requirement (KDR)*—A KDR is a requirement that, to implement, can have a large impact on cost or schedule. A KDR can be of any priority or criticality—knowing the impact a KDR has on the design allows better management of requirements. When under schedule or budget pressure, a KDR that is low priority or low criticality may be a candidate for deletion.
- *Owner*—This is the person or element of the organization that has the right to say something about this requirement. The owner could be the source but they are two different attributes.
- *Rationale*—Rationale defines why the requirement is needed and other information relevant to better understand the reason for and intent of the requirement. Rationale can also define any assumptions that were made when writing the requirement, what design effort drove the requirement, the source of any numbers in the requirement, and note how and why a requirement is constrained (if indeed it is so).
- *Applicability*—This field may be used by an organization that has a family of similar product lines to identify the applicability of the requirement (e.g., to product line, region, or country).
- *Type*—It is often useful to attach to each requirement an attribute of type. While each organization will define types based on how they may wish to organize their requirements, examples of type include input, output, external interfaces, reliability, availability, maintainability, accessibility, environmental conditions, ergonomic, safety, security, facility, transportability, training, documentation, testing, quality provision, policy and regulatory, compatibility with existing systems, standards and technical policies, conversion, growth capacity, and installation. The type field is most useful because it allows the requirements database to be viewed by a large number of designers and stakeholders for a wide range of uses.

More detail on writing text-based requirements can be found in the *INCOSE Guide for Writing Requirements* (INCOSE RWG, 2012), which focuses on the writing of requirements and addresses the characteristics of individual requirement statements, the characteristics of sets of requirements, the attributes of individual requirement statements, and the rules for individual requirement statements.