Appendix C: How to Write a Good Requirement

Use of Correct Terms

- Shall = requirement
- Will = facts or declaration of purpose
- Should = goal

Editorial Checklist

Personnel Requirement

1. The requirement is in the form "responsible party shall perform such and such." In other words, use the active, rather than the passive voice. A requirement must state who shall (do, perform, provide, weigh, or other verb) followed by a description of what must be performed.

Product Requirement

1. The requirement is in the form "product ABC shall XYZ." A requirement must state "The product shall" (do, perform, provide, weigh, or other verb) followed by a description of what must be done.
2. The requirement uses consistent terminology to refer to the product and its lower level entities.
3. Complete with tolerances for qualitative/performance values (e.g., less than, greater than or equal to, plus or minus, 3 sigma root sum squares).
4. Is the requirement free of implementation? (Requirements should state WHAT is needed, NOT HOW to provide it; i.e., state the problem not the solution. Ask, "Why do you need the requirement?" The answer may point to the real requirement.)
5. Free of descriptions of operations? (Is this a need the product must satisfy or an activity involving the product? Sentences like "The operator shall…" are almost always operational statements not requirements.)

Example Product Requirements

- The system shall operate at a power level of…
- The software shall acquire data from the…
- The structure shall withstand loads of…
- The hardware shall have a mass of…

General Goodness Checklist

1. The requirement is grammatically correct.
2. The requirement is free of typos, misspellings, and punctuation errors.
3. The requirement complies with the project's template and style rules.
4. The requirement is stated positively (as opposed to negatively, i.e., "shall not").
5. The use of "To Be Determined" (TBD) values should be minimized. It is better to use a best estimate for a value and mark it "To Be Resolved" (TBR) with the rationale along with what must be done to eliminate the TBR, who is responsible for its elimination, and by when it must be eliminated.
6. The requirement is accompanied by an intelligible rationale, including any assumptions. Can you validate (concur with) the assumptions? Assumptions must be confirmed before baselining.
7. The requirement is located in the proper section of the document (e.g., not in an appendix).

Requirements Validation Checklist

Clarity

1. Are the requirements clear and unambiguous? (Are all aspects of the requirement understandable and not subject to misinterpretation? Is the requirement free from indefinite pronouns (this, these) and ambiguous terms (e.g., "as appropriate," "etc.," "and/or," "but not limited to")?)
2. Are the requirements concise and simple?
3. Do the requirements express only one thought per requirement statement, a standalone statement as opposed to multiple requirements in a single statement, or a paragraph that contains both requirements and rationale?
4. Does the requirement statement have one subject and one predicate?

Completeness

1. Are requirements stated as completely as possible? Have all incomplete requirements been captured as TBDs or TBRs and a complete listing of them maintained with the requirements?
2. Are any requirements missing? For example have any of the following requirements areas been overlooked: functional, performance, interface, environment (development, manufacturing, test, transport, storage, operations), facility (manufacturing, test, storage, operations), transportation (among areas for manufacturing, assembling, delivery points, within storage facilities, loading), training, personnel, operability, safety, security, appearance and physical characteristics, and design.
3. Have all assumptions been explicitly stated?

Compliance

1. Are all requirements at the correct level (e.g., system, segment, element, subsystem)?
2. Are requirements free of implementation specifics? (Requirements should state what is needed, not how to provide it.)
3. Are requirements free of descriptions of operations? (Don't mix operation with requirements: update the ConOps instead.)

Consistency

1. Are the requirements stated consistently without contradicting themselves or the requirements of related systems?
2. Is the terminology consistent with the user and sponsor's terminology? With the project glossary?
3. Is the terminology consistently used through out the document?
4. Are the key terms included in the project's glossary?

Traceability

1. Are all requirements needed? Is each requirement necessary to meet the parent requirement? Is each requirement a needed function or characteristic? Distinguish between needs and wants. If it is not necessary, it is not a requirement. Ask, "What is the worst that could happen if the requirement was not included?"
2. Are all requirements (functions, structures, and constraints) bidirectionally traceable to higher level requirements or mission or system-of-interest scope (i.e., need(s), goals, objectives, constraints, or concept of operations)?
3. Is each requirement stated in such a manner that it can be uniquely referenced (e.g., each requirement is uniquely numbered) in subordinate documents?

Correctness

1. Is each requirement correct?
2. Is each stated assumption correct? Assumptions must be confirmed before the document can be baselined.
3. Are the requirements technically feasible?

Functionality

1. Are all described functions necessary and together sufficient to meet mission and system goals and objectives?

Performance

1. Are all required performance specifications and margins listed (e.g., consider timing, throughput, storage size, latency, accuracy and precision)?
2. Is each performance requirement realistic?
3. Are the tolerances overly tight? Are the tolerances defendable and cost-effective? Ask, "What is the worst thing that could happen if the tolerance was doubled or tripled?"

Interfaces

1. Are all external interfaces clearly defined?
2. Are all internal interfaces clearly defined?
3. Are all interfaces necessary, sufficient, and consistent with each other?

Maintainability

1. Have the requirements for system maintainability been specified in a measurable, verifiable manner?
2. Are requirements written so that ripple effects from changes are minimized (i.e., requirements are as weakly coupled as possible)?

Reliability

1. Are clearly defined, measurable, and verifiable reliability requirements specified?
2. Are there error detection, reporting, handling, and recovery requirements?
3. Are undesired events (e.g., single event upset, data loss or scrambling, operator error) considered and their required responses specified?
4. Have assumptions about the intended sequence of functions been stated? Are these sequences required?
5. Do these requirements adequately address the survivability after a software or hardware fault of the system from the point of view of hardware, software, operations, personnel and procedures?

Verifiability/Testability

1. Can the system be tested, demonstrated, inspected, or analyzed to show that it satisfies requirements? Can this be done at the level of the system at which the requirement is stated? Does a means exist to measure the accomplishment of the requirement and verify compliance? Can the criteria for verification be stated?
2. Are the requirements stated precisely to facilitate specification of system test success criteria and requirements?
3. Are the requirements free of unverifiable terms (e.g., flexible, easy, sufficient, safe, ad hoc, adequate, accommodate, user-friendly, usable, when required, if required, appropriate, fast, portable, light-weight, small, large, maximize, minimize, sufficient, robust, quickly, easily, clearly, other "ly" words, other "ize" words)?

Data Usage

1. Where applicable, are "don't care" conditions truly "don't care"? ("Don't care" values identify cases when the value of a condition or flag is irrelevant, even though the value may be important for other cases.) Are "don't care" conditions values explicitly stated? (Correct identification of "don't care" values may improve a design's portability.)