# Requirements

3.1.1.1 The project shall document the software requirements.

## Notes

The requirements for the content of a Software Requirement Specification and a Data Dictionary document are defined in Chapter 5. The requirements definition activity also includes documenting key decisions, developing requirement rationales, and defining assumptions. The requirement development activities can use lessons learned in performing the logical decomposition process activities. The requirements definition activity provides an understanding of the derived technical requirements baseline, a logical decomposition model, traceability to technical requirements, and an understanding of the stakeholder's expectations.

## Implementation Notes from Appendix D

NPR 7150.2 does not include any notes for this requirement.

## Applicability Across Classes

This requirement applies to all classes and safety criticalities except:

* Class H

Class E and not Safety Critical is labeled with “P(Center). This means that local requirements or procedures describe implementation verification sufficiently to meet the intent of this requirement.

# Rationale

Requirements are the basis for a project. They identify the need to be addressed, the behavior of the system, and the constraints under which the problem is to be solved. They specify the product to be delivered by a provider to a customer.

Clearly defined, well written, and accurately captured requirements “reduce the development effort because less rework is required to address poorly written, missing, and misunderstood requirements.”4 Well-written requirements also provide “a realistic basis for estimating project costs and can be used to evaluate bids or price estimates” and “provide the stakeholders with a basis for acceptance of the system.”4

Requirements serve as the basis for verification activities allowing the developing organization and the customer to judge the completeness of the product.

# Guidance

A general process flow for documenting software requirements is shown below:



Guidance for the content of the documents which capture software requirements, the Software Requirement Specification (SRS) (SWE-109) and the Data Dictionary document (SWE-110) are found in other sections of this handbook. Additionally, software interface requirements may be captured in an Interface Control Document (ICD) or an Interface Requirements Document (IRD), along with hardware interface requirements. If an ICD or IRD is used, the SRS should reference that document.

The following roles may be involved in defining and documenting software requirements as appropriate for the project:

* Project stakeholders, including the customer and senior management
* Software Lead
* Software Requirements Engineer
* Systems Engineer
* Software architects
* Software Assurance Engineer

**Elements to capture**

When capturing software requirements, it is important to:

* Document key decisions and the person(s) who made them, for example:
  + Which requirements are “must-have”
  + Prioritization of requirements
  + Stakeholder decisions that form the basis for requirements
  + Resolutions to conflicting requirements
  + High level design choices that affect low-level requirements
* Develop requirement rationales, for example:
  + Reasons why one feature or performance requirement was chosen over another
  + Originating document or basis for a requirement, e.g., concept of operations, trade study, parent requirement
  + Stakeholder expectations
  + Risks which are the basis for a requirement
  + Technology limitations
  + Time constraints
  + Regulations, laws
* Define assumptions, for example:
  + Environmental or any other constraints
  + Mission type (e.g., human-rated vs. robotic)
  + Assumed technology availability
  + Preset budgetary restrictions
* Logically decompose the requirements
  + Based on top-level requirements and constraints
  + Typically an iterative process as lower levels are defined
  + Decompose requirements into a “set of make-to, buy-to, code-to, and other requirements from which design solutions can be accomplished”4
  + Identify derived requirements during this process
  + Resolve requirements conflicts
  + Lessons learned from previous decomposition activities may be helpful

**Information sources**

“Software Requirements Definition involves eliciting, producing, and analyzing customer, product, and product component requirements.”5 Inputs to this process may include:

* System and subsystem requirements documents, hardware schematics and specifications
* System architecture
* System models and simulations
* System safety analyses, including the preliminary hazard analysis (PHA), subsequent system hazard analyses, and software safety analyses
* Environmental requirements, including operations and hardware requirements, vehicle or facility requirements
* Standards
* External regulations
* Program/Project specification
* Concept of operations
* Interface requirements
* Legacy products
* Organizational requirements
* Quality attributes (e.g., reliability, availability, security, safety, maintainability, portability, usability)
* Structured interviews with customers, users (may include development of scenarios, examination of reports, analysis of competing products)
* Brainstorming sessions with customers, users, developers
* Stakeholder input or user needs (provided or elicited via interviews, prototypes, questionnaires, surveys, or other techniques)

**General guidance**

Some general guidance to follow when defining and documenting software requirements includes:

* Provide a unique identifier for each requirement
* Express requirements as “shall” statements (avoiding words like “could”, “should”, “must”, and “will”)
* Clearly identify software safety requirements
* Structure requirements so they are complete, correct, consistent, traceable, independent, “clear, precise, unequivocal, verifiable, testable, maintainable and feasible”1
* Refine the initial set of requirements into a manageable set (e.g., remove duplicates, remove unnecessary requirements, combine requirements, clarify requirements, keep “must-haves”, drop some “nice-to-haves”)
* Have requirements reviewed by stakeholders (to identify and address ambiguous, conflicting, incomplete requirements; peer reviews/inspections is one technique)
* Capture software requirements in the required documents and in a requirements management tool for easy access, manipulation, and management; some tools may allow for generation of the requirements documents directly from the tool
* State the requirements, not how to fulfill them (i.e., avoid including design decisions in the requirements)
* State only one requirement per statement (i.e., compound requirements should be avoided)

**Common problems**

Defining and documenting requirements is not a simple task. Common problems that occur during or as a result of this activity and which should be avoided include:

* Failing to define needed requirements
* Writing requirements inconsistently or ambiguously
* Using inexperienced personnel to define the requirements
* Incorrect understanding of underlying assumptions or constraints
* Including unneeded features or capabilities
* No clear method for allocating requirements to subsystems
* Choosing solutions before defining the requirements and/or user needs
* Failing to spend enough time or resources on requirements definition6

Consult Center Process Asset Libraries (PALs) for Center-specific guidance and resources related to documenting software requirements, including templates, checklists, and sample documents.

Additional guidance related to documenting software requirements may be found in the following related requirements in this handbook:

|  |  |
| --- | --- |
| **SWE-050** | Software Requirements |
| **SWE-051** | Software Requirements Analysis |
| **SWE-052** | Bidirectional Traceability (software requirements to higher level requirements) |
| **SWE-053:** | Manage Requirements Change |
| **SWE-109:** | Software Requirements Specification |

## Small Projects

“Any project with resource limitations must establish the relative priorities of the requested features, use cases, or functional requirements. Prioritization helps the project manager plan for staged releases, make trade-off decisions, and respond to requests for adding more functionality. It can also help you avoid the traumatic ‘rapid descoping phase’ late in the project, when you start throwing features overboard to get a product out the door on time.”3

# Resources

1. NASA Technical Standard, “[NASA Software Safety Standard](https://standards.nasa.gov/documents/detail/3314914)”, NASA-STD-8719.13B w/ Change 1, 2004.
2. NASA Technical Standard, “[NASA Software Safety Guidebook](https://standards.nasa.gov/documents/detail/3315126)”, NASA-GB-8719.13, 2004.
3. K. Wiegers, “When Telepathy Won’t Do: Requirements Engineering Key Practices”.
4. NASA Scientific and Technical Information (STI), NASA Center for AeroSpace Information, “[NASA Systems Engineering Handbook](http://www.ap233.org/ap233-public-information/reference/20080008301_2008008500.pdf)”, NASA/SP-2007-6105, Rev1, 2007.
5. Software Development Process Description Document, EI32-OI-001, Revision R, 2010.
6. Kandt, Ronald Kirk, Jet Propulsion Lab, “Software Quality Improvement, Software Requirements Engineering: Practices and Techniques”, JPL Document D-24994, 2003.
7. “Product Requirements Development and Management Procedure”, 5526\_7-21-06\_Req\_RevA\_generic-R1V0, 2006.
8. Software Engineering Institute, “[CMMI for Development, Version 1.3](http://www.sei.cmu.edu/reports/10tr033.pdf)”, CMU/SEI-2010-TR-033, 2010.

## Tools

Please reference table XYZ in this handbook for a list of tools in use at NASA that may be relevant to this requirement. Note that this table should not be considered all-inclusive, nor is it an endorsement of any particular tool. Check with your Center to see what tools are available to facilitate compliance with this requirement.

# Lessons Learned

The NASA Lessons Learned database contains the following lessons learned related to defining and documenting software requirements:

* **Developing a complete set of requirements:** “Development of a complete and consistent set of engineering requirements requires a robust systems engineering process that defines performance and resource utilization requirements, traces requirements to higher and lower-level requirements, ensures review of requirements by key stakeholders and by parties independent of the engineering of the requirements, and assesses the requirements using a checklist of questions that address quality concerns.” (<http://www.nasa.gov/offices/oce/llis/imported_content/lesson_2218.html>)
* **COTS/GOTS/MOTS software**: “Define “must meet,” “highly desirable” and “nice to have” requirements. Ability of the unit to meet those requirements, and at what cost, will be a major deciding factor in the COTS decision.” (<http://www.nasa.gov/offices/oce/llis/1370.html>)
* **Customer role in requirements definition:** “Additionally, a collaborative relationship between the customer using the software and the developer providing the software is paramount to the success of the software project. More specifically, the users/customers must effectively define and accurately communicate their requirements to the developer. For example, the user’s defined requirements should be clearly stated and unambiguous, concise, complete, autonomous, able to be implemented, and testable.” (<http://www.nasa.gov/offices/oce/llis/imported_content/lesson_3377.html>)
* **Include hardware requirements**: As part of the probable scenario for Mars Polar Lander Mission Loss, a lesson learned was that “All known hardware operational characteristics, including transients and spurious signals, must be reflected in the software requirements documents and verified by test.” (<http://www.nasa.gov/offices/oce/llis/0938.html>)
* **Stable requirements key to program stability**: “Stable Requirements are key to program stability. Do not allow requirements to creep. One step in this process is to involve all stakeholders in developing and finalizing requirements. Stakeholders may include potential development and operations contractors, engineering and other organizations and NASA Headquarters.” (<http://www.nasa.gov/offices/oce/llis/0987.html>)
* **Involve operation and systems engineering**: “Include operations in requirements definition and early design…Do not overlook the need for strong systems engineering involvement in requirements and verification traceability.” (<http://www.nasa.gov/offices/oce/llis/0987.html>)
* **Formal requirements documentation**: “Formal documentation of requirements, resolutions, and decisions -- including maintaining records of the basis and justification for each engineering decision -- was once a standard NASA practice. The increased use of informal records such as presentation slides and e-mail may be inhibiting the ability of NASA programs and projects to reference technical decisions and to validate or verify engineering designs.” (<http://www.nasa.gov/offices/oce/llis/imported_content/lesson_1715.html>)

|  |  |  |  |
| --- | --- | --- | --- |
| **Responsible Org.** | **Milestone Review** | **Software Classes** | **Documents** |
| PM  CSMA  HQSMA  CCE  HQCE  SAM  CD  IVV  SRA  MD  **CTO** | MCR  SRR  SwRR  MDR  SDR  PDR  CDR  PRR  SIR  TRR  SAR  ORR  FRR | ASC  ANSC  BSC  BNSC  CSC  CNSC  DSC  DNSC  ESC  ENSC  F  G  H | Plan  Procedure  Process  Studies  Reports  Analysis  Records  Prod Desc |
|  |  |  | **Acquisition** |
|  |  |  | Plan  In/Over  SM |
| **Product Dev.** |  |  |  |
| SSE  SRE  Design  C&I  V&V  PR  Sus Engr |  |  |  |
|  | **Org. Support** | **Project Mgt.** | **Assets** |
|  | CM  M&A  PE  SA  Train | PF  PP  PM&C  PI  PC | BiCE  BP  LL  Tool |