



NASA Procedural Requirements

COMPLIANCE IS MANDATORY

NPR 7150.2B
Effective Date: November 19,
2014
Expiration Date: November
19, 2019

[Printable Format \(PDF\)](#)

Request Notification of Change (NASA Only)

Subject: NASA Software Engineering Requirements

Responsible Office: Office of the Chief Engineer

[| TOC](#) | [Preface](#) | [Chapter1](#) | [Chapter2](#) | [Chapter3](#) | [Chapter4](#) | [Chapter5](#) | [Chapter6](#) | [AppendixA](#) | [AppendixB](#) | [AppendixC](#) | [AppendixD](#) | [AppendixE](#) | [ALL](#) |

Chapter 3: Software Management Requirements

The software management activities define and control the many software aspects of a project from beginning to end. This includes the interfaces to other organizations, determination of deliverables, estimates and tracking of schedule and cost, risk management, formal and informal reviews as well as other forms of verification and validation, and determination of the amount of supporting services. The planned management of these activities is captured in one or more software and/or system plans.

3.1 Software Life Cycle Planning

3.1.1 Software life cycle planning covers the software aspects of a project from inception through retirement. The software life cycle planning cycle is an organizing process that considers the software as a whole and provides the planning activities required to ensure a coordinated, well-engineered process for defining and implementing project activities. These processes, plans, and activities are coordinated within the project. At project conception, software needs for the project are analyzed, including acquisition, supply, development, operation, maintenance, retirement, and supporting activities and processes. The software effort is scoped and the processes, measurements, and activities are documented in software plan(s). As noted earlier in Section 1.1.4, this NPR makes no recommendation for a specific software life-cycle model (i.e., it allows agile, incremental, spiral, etc., life-cycle models). However, expectations from the system project life-cycle models need to be adequately addressed in the software plan(s).

3.1.2 The project manager shall develop, maintain, and execute software plans that cover the entire software life cycle and, as a minimum, address the requirements of this directive with approved tailoring. [SWE-013]

Note: The recommended practices and guidelines for the content of different types of software planning activities (whether stand-alone or condensed into one or more project level or software documents or electronic files) are defined in NASA-HDBK-2203.

3.1.3 The project manager shall track the actual results and performance of software activities against the software plans. [SWE-024]

- a. Corrective actions are taken, recorded, and managed to closure.
- b. Changes to commitments (e.g., software plans) that have been agreed to by the affected groups and individuals.

3.2 Software Cost Estimation

3.2.1 The project manager shall establish, document, and maintain two cost estimates and associated cost parameters for all software Class A and B projects that have an estimated project cost of \$2 million or more or one software cost estimate and associated cost parameter(s) for other software projects. [SWE-015]

3.2.2 The project manager's software cost estimate(s) shall satisfy the following conditions: [SWE-151]

- a. Covers the entire software life cycle.

- b. Is based on selected project attributes (e.g., assessment of the size, functionality, complexity, criticality, reuse code, modified code, and risk of the software processes and products).
- c. Is based on the cost implications of the technology to be used and the required maturation of that technology.
- d. Incorporates risk and uncertainty.
- e. Includes the cost for software assurance support.
- f. Includes other direct costs.

Note: In the event of a decision to outsource, it is a best practice that both the acquirer (NASA) and the provider (contractor/subcontractor) be responsible for developing software cost estimates. For any class of software that has significant risk exposure, consider performing at least two cost estimates.

3.3 Software Schedules

3.3.1 The project manager shall document and maintain a software schedule that satisfies the following conditions: [SWE-016]

- a. Coordinates with the overall project schedule.
- b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system.
- c. Reflects the critical path for the software development activities.
- d. Adhere to the guidance provided in NASA/SP-2010-3403, NASA Scheduling Management Handbook.

3.3.2 The project manager shall regularly hold reviews of software activities, status, and results with the project stakeholders and track issues to resolution. [SWE-018]

3.3.3 The project manager shall select and document a software development life cycle or model that includes phase transition criteria for each life-cycle phase. [SWE-019]

3.4 Software Project Specific Training

3.4.1 The project manager shall plan, track, and ensure project specific software training for project personnel. [SWE-017]

Note: This includes any software assurance personnel assigned to the project.

3.5 Software Classification and Planning Assessments

3.5.1 The project manager shall classify each system and subsystem containing software in accordance with the highest applicable software classification definitions for Classes A, B, C, D, E, F, G, and H software in Appendix D. [SWE-020]

Note: The expected applicability of requirements in this directive to specific systems and subsystems containing software is determined through the use of the NASA-wide definitions for software classes in Appendix D and the designation of the software as safety critical or non-safety critical in conjunction with the Requirements Mapping and Compliance Matrix in Appendix C. These definitions are based on: (1) usage of the software with or within a NASA system, (2) criticality of the system to NASA's major programs and projects, (3) extent to which humans depend upon the system, (4) developmental and operational complexity, and (5) extent of the Agency's investment. Software classification tool details are defined in NASA-HDBK-2203.

3.5.2 The project's software assurance manager shall perform an independent classification assessment. [SWE-132]

Note: Engineering and software assurance must reach agreement on the software classification determination of the software. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains.

3.5.3 The project manager, in conjunction with the Safety and Mission Assurance organization, shall determine the software safety criticality in accordance with NASA-STD-8719.13. [SWE-133].

Note: Software Safety Critical Assessment Tool, in NASA-HDBK-2203, can be used to determine the software

safety criticality. Engineering and software assurance must reach agreement on safety-critical determination of the software. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains.

3.5.4 If a system or subsystem evolves to a higher or lower software classification as defined in Appendix D, or there is a change in the safety criticality of the software, then the project manager shall update their plan to fulfill the applicable requirements per the Requirements Mapping and Compliance Matrix in Appendix C and any approved tailoring. [SWE-021]

3.5.5 If a software component is determine to be safety critical software then software component classification shall be Software Class D or higher. [SWE-160]

3.6 Software Assurance and Software IV&V

3.6.1 The project manager shall plan and implement software assurance per NASA-STD-8739.8. [SWE-022]

Note: Software assurance activities occur throughout the life of the project. Some of the actual analyses and activities may be performed by engineering or the project.

3.6.2 For projects reaching Key Decision Point (KDP) A after the effective date of this directive's revision, the program manager shall ensure that software IV&V is performed on the following categories of projects: [SWE-141]

- a. Category 1 projects as defined in NPR 7120.5.
- b. Category 2 projects as defined in NPR 7120.5 that have Class A or Class B payload risk classification per NPR 8705.4.
- c. Projects specifically selected by the NASA CSMA to have software IV&V.

Note: The NASA IV&V Board of Advisors supports the NASA CSMA by recommending significant project needs for software IV&V beyond projects meeting the criteria in items a. and b. of SWE-141. Waivers to the above requirement will be written by the project and responsible Center SMA organization, adjudicated by the NASA IV&V Board of Advisors, with the final decision by the NASA CSMA. Additional projects, projects in other phases, or projects without a payload risk classification can be selected by the NASA CSMA to be required to have software IV&V. It is NASA policy to use the NASA IV&V Facility as the sole provider of IV&V services when software created by or for NASA is selected for IV&V by the NASA CSMA. IV&V support is funded and managed independent of the selected project.

3.6.3 If software IV&V is performed on a project, the project manager shall ensure that an IV&V Project Execution Plan (IPEP) is developed. [SWE-131]

Note: The scope of IV&V services is determined by the project and the IV&V provider, and is documented in the IPEP. The IPEP is developed by the IV&V provider and serves as the operational document that will be shared with the project receiving IV&V support. In accordance with the responsibilities defined in NPD 7120.4, section 5.J.(5), projects ensure that software providers allow access to software and associated artifacts to enable implementation of IV&V. A template and instructions for an IPEP may be found in the NASA IV&V Management System, accessible at <http://www.nasa.gov/centers/ivv/ims/home/index.html>

3.7 Safety-critical Software

3.7.1 When a project is determined to have safety-critical software, the project manager shall implement the requirements of NASA-STD-8719.13. [SWE-023]

3.7.2 When a project is determined to have safety-critical software, the project manager shall implement the following items in the software: [SWE-134]

- a. Safety-critical software is initialized, at first start and at restarts, to a known safe state.
- b. Safety-critical software safely transitions between all predefined known states.
- c. Termination performed by software of safety critical functions is performed to a known safe state.
- d. Operator overrides of safety-critical software functions require at least two independent actions by an operator.
- e. Safety-critical software rejects commands received out of sequence, when execution of those commands out of sequence can cause a hazard.
- f. Safety-critical software detects inadvertent memory modification and recovers to a known safe state.
- g. Safety-critical software performs integrity checks on inputs and outputs to/from the software system.
- h. Safety-critical software performs prerequisite checks prior to the execution of safety-critical software commands.

- i. No single software event or action is allowed to initiate an identified hazard.
- j. Safety-critical software responds to an off nominal condition within the time needed to prevent a hazardous event.
- k. Software provides error handling of safety-critical functions.
- l. Safety-critical software has the capability to place the system into a safe state.
- m. Safety-critical elements (requirements, design elements, code components, and interfaces) are uniquely identified as safety-critical.
- n. Requirements are incorporated in the coding methods, standards, and/or criteria to clearly identify safety-critical code and data within source code comments.

Note: These requirements are applicable to components that reside in a safety-critical system, and the components control, mitigate, or contribute to a hazard as well as software used to command hazardous operations/activities.

3.8 Automatic Generation of Software Source Code

3.8.1 The project manager shall define the approach to the automatic generation of software source code including: [SWE-146]

- a. Validation and verification of auto-generation tools.
- b. Configuration management of the auto-generation tools and associated data.
- c. Identification of the allowable scope for the use of auto-generated software.
- d. Verification and validation of auto-generated source code.
- e. Monitoring the actual use of auto-generated source code compared to the planned use.
- f. Policies and procedures for making manual changes to auto-generated source code.
- g. Configuration management of the input to the auto-generation tool, the output of the auto-generation tool, and modifications made to the output of the auto-generation tools.

3.9 Use of Commercial, Government, Legacy, Heritage, and Modified Off-the-Shelf Software

3.9.1 Projects utilizing commercial, government, legacy, heritage, and MOTS software components typically take into consideration the importance of planning and managing the inclusion of those components into the project software. The off-the-shelf software discussed here applies only when the off-the-shelf software elements are to be included as part of a NASA system (per Section P.2.b). The following requirements do not apply to stand-alone desktop applications (e.g., word processing programs, spreadsheet programs, presentation programs). When software components use COTS applications (e.g., spreadsheet programs, database programs) within a NASA system/subsystem application, the software components typically are assessed and classified as part of the software subsystem in which they reside. Note that commercial, government, legacy, heritage, and MOTS software also have to meet the applicable requirements for each class of software.

3.9.2 The project manager shall satisfy the following conditions when a COTS, GOTS, MOTS, or reused software component is acquired or used: [SWE-027]

- a. The requirements to be met by the software component are identified.
- b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).
- c. Proprietary rights, usage rights, ownership, warranty, licensing rights, and transfer rights have been addressed.
- d. Future support for the software product is planned and adequate for project needs.
- e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.
- f. The project has a plan to perform periodic assessments of vendor reported defects to ensure the defects do not impact the selected software components.

Note: The project responsible for procuring off-the-shelf software is responsible for documenting, prior to procurement, a plan for verifying and validating the software to the same level that would be required for a developed software component. The project ensures that the COTS, GOTS, MOTS, reused, and auto generated code software components and data meet the applicable requirements in this directive assigned to

its software classification as shown in Appendix C. Open source requirements are in Section 3.15.

3.10 Software Verification and Validation

3.10.1 Ensuring that the software products meet their requirements and intended usage, and that the products were built correctly is the purpose of verification and validation. Both software validation and software verification activities span the entire software life cycle and need to be planned. Software validation and software verification activities can include software formal and informal reviews, software peer reviews, software inspections, software testing, software demonstrations, and software analyses. Because software peer reviews and inspections are such an important verification and validation tool with proven value, specific software peer review and inspection requirements are contained in Chapter 5 of this directive.

3.10.2 The project manager shall plan software verification activities, methods, environments, and criteria for the project. [SWE-028]

3.10.3 The project manager shall plan the software validation activities, methods, environments, and criteria for the project. [SWE-029]

3.10.4 The project manager shall record, address, and track to closure the results of software verification activities. [SWE-030]

3.10.5 The project manager shall record, address, and track to closure the results of software validation activities. [SWE-031]

3.11 Software Development Processes

3.11.1 The use of the CMMI model is included to make sure NASA projects are supported by software development organization(s) having the necessary skills and processes in place to produce reliable products within cost and schedule estimates. The CMMI requirement, SWE-032, provides NASA with a methodology to:

- a. Measure software development organizations against an industry-wide set of best practices that address software development and maintenance activities applied to products and services.
- b. Measure and compare the maturity of an organization's product development and acquisition processes with industry state of the practice.
- c. Measure and ensure compliance with the intent of the NPR 7150.2 process related requirements using an industry standard approach.
- d. Assess internal and external software development organization's processes.
- e. Identify potential risk areas within a given organization's software development processes.

3.11.2 The CMMI-DEV is an internationally used framework for process improvement in development organizations. It is an organized collection of best practices and proven processes that thousands of software organizations have both used and been appraised against for over the past two decades. CMMI defines practices that businesses have implemented on their way to success. Practices cover topics that include eliciting and managing requirements, decision making, measuring performance, planning work, handling risks, and more. Using these practices, NASA can improve NASA software projects' chances of mission success. CMMI ratings can cover a team, a work group, a project, a division, or an entire organization. When evaluating software suppliers, it's important to make sure that the specific organization doing the software work on the project has the cited rating (as some parts of a company may be rated while others are not).

3.11.3 The project manager shall acquire, develop, and maintain software from an organization with a non-expired CMMI-DEV rating as measured by a CMMI Institute authorized or certified lead appraiser as follows: [SWE-032]

a. For Class A software: CMMI-DEV Maturity Level 3 Rating or higher for software, or CMMI-DEV Capability Level 3 Rating or higher in all CMMI-DEV Maturity Level 2 and 3 process areas for software.

b. For Class B software (except Class B software on NASA Class D payloads, as defined in NPR 8705.4): CMMI-DEV Maturity Level 2 Rating or higher for software, or CMMI-DEV Capability Level 2 Rating or higher for software in the following process areas:

- (1) Requirements Management.
- (2) Configuration Management.
- (3) Process and Product Quality Assurance.
- (4) Measurement and Analysis.

- (5) Project Planning.
- (6) Project Monitoring and Control.
- (7) Supplier Agreement Management (if applicable).

Note: Organizations that have completed Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) Class A appraisals against the CMMI-DEV model are to maintain their rating and have their results posted on the CMMI Institute Web site so that NASA can assess the current maturity/capability rating. Software development organizations need to be reappraised and keep an active appraisal rating posted on the CMMI® Institute Website during the time that they are responsible for the development and maintenance of the software.

Note: For Class B software, in lieu of a CMMI® rating by a development organization, the project will conduct an evaluation, performed by a qualified evaluator selected by the Center Engineering Technical Authority, of the seven process areas listed in SWE-032 and mitigate any risk, if deficient. This exception is intended to be used in those cases in which NASA wishes to purchase a product from the "best of class provider," but the best of class provider does not have the required CMMI® rating. When this exception is exercised, the Center Engineering Technical Authority is notified.

Note: For Class B software on NASA Class D Payloads and Class C software, it is highly recommended that providers have a Certified CMMI® Lead Appraiser conduct periodic informal evaluations (e.g., Appraisal Class Bs or Cs) against relevant process areas.

3.12 Software Acquisition

3.12.1 The requirements in this section are applicable for both NASA contracted software procurements (e.g., reuse of existing software, modification of existing software, contracted and subcontracted software, and/or development of new software) and in-house developments. Acquisition requirements are focused both inside the acquisition organization, to ensure the acquisition is conducted effectively, and outside the acquisition organization, as the organization conducts project monitoring and control of its suppliers. These acquisition requirements provide a foundation for acquisition process discipline and rigor that enables product and service development to be repeatedly executed with high levels of acquisition success. This section contains project software acquisition and contract requirements to ensure that the project has the data needed for the review of provided systems and/or services. The project is responsible for ensuring that these requirements apply when software activities are developed in-house, contracted directly, or subcontracted from a NASA prime contractor. These requirements are used in addition to, not in place of, the other requirements of this directive.

3.12.2 The project manager shall assess options for software acquisition versus development. [SWE-033]

Note: The assessment can include risk, cost, and benefits criteria for each of the options listed below:

- a. Acquire an off-the-shelf software product that satisfies the requirement.
- b. Develop the software product or obtain the software service internally.
- c. Develop the software product or obtain the software service through contract.
- d. Enhance an existing software product or service.
- e. Reuse an existing software product or service.

3.12.3 The project manager shall define and document the acceptance criteria and conditions for the software. [SWE-034]

3.12.4 The project manager shall establish a procedure for software supplier selection, including proposal evaluation criteria. [SWE-035]

3.12.5 The project manager shall determine which software processes, software documents, electronic products, software activities, and tasks are required for the project and software suppliers. [SWE-036]

Note: A list of typical software engineering products or electronic data products used on a software project is contained in Chapter 6 of this directive.

3.12.6 The project manager shall define the milestones at which the software supplier(s) progress will be reviewed and audited as a part of the acquisition activities. [SWE-037]

3.12.7 The project manager shall document software acquisition planning decisions. [SWE-038]

3.12.8 The project manager shall require the software supplier(s) to provide insight into software development and test activities; at a minimum, the software supplier(s) will be required to allow the project manager or designate to: [SWE-039]

- a. Monitor product integration.
- b. Review the verification activities to ensure adequacy.
- c. Review trades studies and source data.
- d. Audit the software development process.
- e. Participate in software reviews and systems and software technical interchange meetings.

3.12.9 The project manager shall require the software supplier(s) to provide NASA with software products and software process tracking information, in electronic format, including software development and management metrics. [SWE-040]

3.12.10 The project manager shall require the software supplier(s) to provide NASA with electronic access to the source code developed for the project in a modifiable format, including MOTS software and non-flight software (e.g., ground test software, simulations, ground analysis software, ground control software, science data processing software, and hardware manufacturing software). [SWE-042]

Note: The electronic access requirements for the source code, software products, and software process tracking information implies that NASA gets electronic copies of the items for use by NASA at NASA facilities.

3.13 Software Monitoring

3.13.1 The project manager shall require the software supplier to track software changes and non-conformances and provide the data for the project's review. [SWE-043]

3.13.2 The project manager shall participate in any joint NASA/supplier audits of the software development process and software configuration management process. [SWE-045]

3.13.3 The project manager shall require the software supplier(s) to provide a software schedule for the project's review and schedule updates as requested. [SWE-046]

3.13.4 The project manager shall require the software supplier(s) to make electronically available the software traceability data for the project's review. [SWE-047]

3.14 Software Reuse

3.14.1 Software reuse entails capitalizing on existing software and systems to create new products. Successful reuse requires the integration of reuse-related activities into the life cycle to create reusable assets for current and future software and systems. Unless reuse is explicitly planned into life-cycle processes, an organization will not be able to repeatedly exploit reuse opportunities in multiple software projects or products. Systematic reuse is the practice of reuse according to a consistent, repeatable process. Practicing systematic reuse requires a focus on the use of engineering principles for all reuse assets involved in development. The major benefits that systematic reuse can deliver are as follows:

- a. Increase software productivity.
- b. Shorten software development and maintenance time.
- c. Reduce duplication of effort.
- d. Move personnel, tools, and methods more easily among projects.
- e. Reduce software development and maintenance costs.
- f. Produce higher quality software products.
- g. Increase software and system dependability.

3.14.2 The project manager shall specify reusability requirements that apply to its software development activities to enable future reuse of the software, including models used to generate the software. [SWE-147]

3.14.3 The project manager shall evaluate software for potential reuse by other projects across the Agency and contribute reuse candidates to the Agency Software Catalog. [SWE-148]

Note: The Agency Software Catalog is maintained as a part of the NASA Technology Transfer Portal. Each software code listed in the catalog is evaluated for access requirements and restrictions per the software release process (see <http://technology.nasa.gov/> and NPR 2210.1).

3.15 Open Source

3.15.1 Open Source Software (OSS) is commercial off-the-shelf software (COTS) that is licensed to allow distribution, use, and redistribution of the software source code, including modifications. There are many different types of OSS licenses, though any software license that has been approved by the Open Source Initiative (OSI) allows, at a minimum, use, modification and redistribution of the source code for any purpose. Most OSS licenses allow the software to become closed source and do not require that the source code and any modifications be redistributed, while other OSS licenses require that the source code and any modifications be made available to whomever the end product is distributed to. Many OSS projects are supported by multiple commercial organizations directly, and because the software is available for modification, a particular software project can also be supported by new vendors or directly by NASA where appropriate. Leveraging OSS in NASA software requires understanding of the architecture and implementation of the OSS, its technical merit, and a legal review of its use related to licensing and intellectual property.

3.15.2 The project manager shall ensure that when an OSS component is acquired or used, the following conditions are satisfied: [SWE-149]

- a. The requirements that are to be met by the software component are identified.
- b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).
- c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed.
- d. Future support for the software product is planned and adequate for project needs.
- e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.

Note: It is important to understand that under copyright law, OSS is a form of commercial software that needs to be treated with the same respect as any other commercial software. For this reason, it is important to understand both the specifics of the open source license in question and how the project intends to use and redistribute any modified OSS. It is the project's responsibility for both commercial and OSS to verify that the Government receives sufficient rights in any source or executable code, libraries, or "building blocks" (COTS/GOTS/MOTS & OSS) to meet the project's needs along with any anticipated further Government applications. This may include verifying that the license does not contain any undesired requirements or restrictions on redistribution, modification and release, etc. Seek guidance from your Center Office of Chief Counsel for help in making these determinations.

3.15.3 The project manager shall require the software supplier(s) to notify the project, in the response to the solicitation, as to whether or not open source software will be included in code developed for the project. [SWE-041]

3.16 Software Security

3.16.1 A central and critical aspect of the computer security problem is a software problem. Software defects with security ramifications include implementation bugs such as buffer overflows and design flaws such as inconsistent error handling. The following requirements in section 3.16 are for space flight software only. Security requirements for the acquisition, development, integration, and modification of ground software systems are found in NPR 2810.1.

3.16.2 The project manager shall ensure that security risks in space flight software systems are identified and security risk mitigations are planned for these systems in the Project Protection Plan. [SWE-154]

3.16.3 The project manager shall implement the identified software security risk mitigations addressed in the Project Protection Plan. [SWE-155]

3.16.4 The project manager shall ensure and record that all systems including space flight software are evaluated for security risks, including risks posed by the use of COTS, GOTS, MOTs, Open Source, and reused software. [SWE-156]

3.16.5 The project manager shall ensure that software systems with space communications capabilities are protected against un-authorized access. [SWE-157]

3.16.6 The project manager shall ensure that the space flight software systems are assessed for possible security vulnerabilities and weaknesses. [SWE-158]

3.16.7 The project manager shall verify and validate the required software security risk mitigations to ensure that security objectives identified in the Project Protection Plan for space flight software are satisfied in their implementation. [SWE-159]

Note: include assessments for security vulnerabilities during Peer Review/Inspections of software requirements and design and undergo automated security static analysis as well as coding standard static analyses of software code to find potential security vulnerabilities.

| [TOC](#) | [Preface](#) | [Chapter1](#) | [Chapter2](#) | [Chapter3](#) | [Chapter4](#) | [Chapter5](#) | [Chapter6](#) | [AppendixA](#) |
[AppendixB](#) | [AppendixC](#) | [AppendixD](#) | [AppendixE](#) | [ALL](#) |

| [NODIS Library](#) | [Program Formulation\(7000s\)](#) | [Search](#) |

DISTRIBUTION:
NODIS

This Document Is Uncontrolled When Printed.

Check the NASA Online Directives Information System (NODIS) Library
to Verify that this is the correct version before use: <http://nodis3.gsfc.nasa.gov>
