NASA Procedural Requirements

NPR 7150.2D
Effective Date: March 08, 2022
Expiration Date: March 08, 2027

COMPLIANCE IS MANDATORY FOR NASA EMPLOYEES

NASA Software Engineering Requirements

Responsible Office: Office of the Chief Engineer

Table of Contents

Preface

P.1 Purpose
P.2 Applicability
P.3 Authority
P.4 Applicable Documents and Forms
P.5 Measurement/Verification
P.6 Cancellation

Chapter 1. Introduction

1.1 Overview
1.2 Hierarchy of NASA Software-Related Engineering and Program/Project Documents
1.3 Document Structure

Chapter 2. Roles, Responsibilities, and Principles Related to Tailoring of the Requirements

2.1 Roles and Responsibilities
2.2 Principles Related to Tailoring of the Requirements

Chapter 3. Software Management Requirements

3.1 Software Life Cycle Planning
3.2 Software Cost Estimation
3.3 Software Schedules

3.4 Software Training
3.5 Software Classification Assessments
3.6 Software Assurance and Software Independent Verification & Validation
3.7 Safety-Critical Software
3.8 Automatic Generation of Software Source Code
3.9 Software Development Processes and Practices
3.10 Software Reuse
3.11 Software Cybersecurity
3.12 Software Bi-Directional Traceability

Chapter 4. Software Engineering (Life Cycle) Requirements

4.1 Software Requirements
4.2 Software Architecture
4.3 Software Design
4.4 Software Implementation
4.5 Software Testing
4.6 Software Operations, Maintenance, and Retirement

Chapter 5. Supporting Software Life Cycle Requirements

5.1 Software Configuration Management
5.2 Software Risk Management
5.3 Software Peer Reviews/Inspections
5.4 Software Measurements
5.5 Software Non-conformance or Defect Management

Chapter 6. Recommended Software Documentation Contents

6.1 Software Engineering Products
6.2 Software Engineering Product Content

Appendix A. Definitions
Appendix B. Acronyms
Appendix C. Requirements Mapping Matrix
Appendix D. Software Classifications
Appendix E. References

List of Figures

Figure 1. NASA Software Classification Structure

List of Tables

Table 1. Bi-directional traceability by software classification
Table 2. Requirements Mapping Matrix

Preface

P.1 Purpose

Software engineering is a core capability and key enabling technology for NASA"s missions and supporting infrastructure. This NASA Procedural Requirement (NPR) establishes the engineering requirements for software acquisition, development, maintenance, retirement, operations, and management consistent with the governance model contained in NASA Policy Directive (NPD) 1000.0, NASA Governance and Strategic Management Handbook. This NASA Procedural Requirements (NPR) supports the implementation of NPD 7120.4, NASA Engineering and Program/Project Management Policy.

P.2 Applicability

a. This NPR applies to NASA Headquarters (HQ) and NASA Centers, including Component Facilities and Technical and Service Support Centers. This language applies to the Jet Propulsion Laboratory (JPL) (a Federally Funded Research and Development Center (FFRDC)), other contractors, grant recipients, or parties to cooperative agreements and other agreements only to the extent specified or referenced in the appropriate contracts, grants, or agreements.

Note: The above statement alone is not sufficient to stipulate requirements for the contractor, grant recipient, or agreement. This NPR provides requirements for NASA contracts, grant recipients, or agreements to the responsible NASA project managers, contracting officers, and the contracting officers representatives that are made mandatory through contract clauses, specifications, or statements of work (SOWs) in conformance with the NASA Federal Acquisition Regulation (FAR) Supplement or by stipulating in the contracts, grants, or agreements which of the NPR requirements apply.

b. This NPR applies to the complete software development life cycle, including software planning, development, testing, maintenance, retirement, operations, management, acquisition, and assurance activities. The requirements of this directive cover such software created, acquired, or maintained by NASA or for NASA to the extent specified or referenced in an appropriate contract, grant, or cooperative agreement. The applicability of these requirements to specific systems and subsystems within the Agency's investment areas, programs, and projects is through the use of the NASA-wide definition of software classes, defined in Appendix D. Some projects may contain multiple software systems and software subsystems having different software classes. For this directive, software is defined in Appendix A, and includes software executing on processors embedded in programmable logic devices.

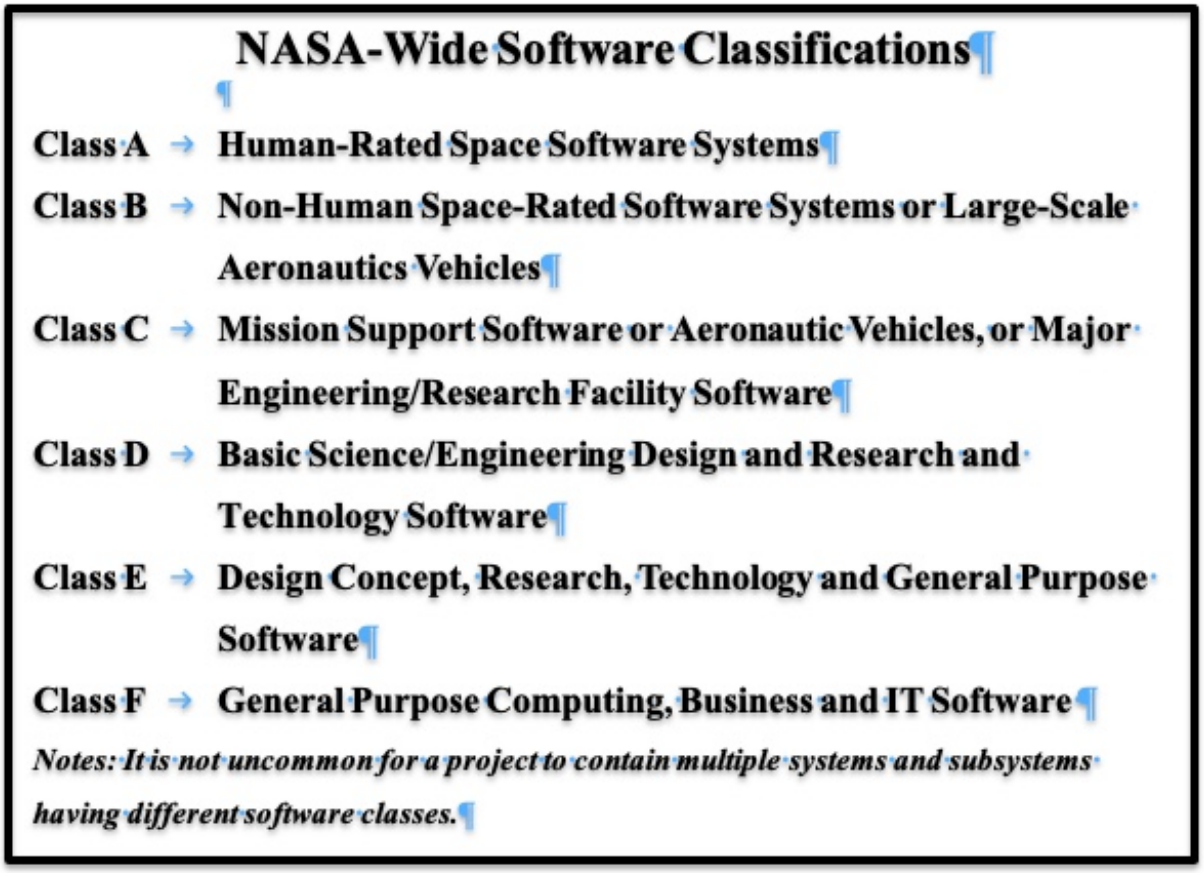

Figure 1. NASA software classification structure.

c. For existing Class A through E programs and projects, the software engineering requirements of this NPR apply to their current and future phases as determined by the responsible Mission Directorate as approved by the NASA Chief Engineer (or as delegated).

d. For existing Class F programs and projects, the software engineering requirements of this NPR apply to their current and future phases as determined by the Center Chief Information Officer (CIO) and approved by the NASA CIO (or delegate).

e. This NPR can be applied to other NASA investments at the discretion of the responsible manager or the NASA Associate Administrator. This NPR is not retroactively applicable to software development, maintenance, operations, management, acquisition, and assurance activities started before the effective date of this NPR (i.e., existing systems and subsystems containing software for the International Space Station, Hubble, Chandra, etc.).

f. This NPR does not supersede more stringent requirements imposed by individual NASA organizations and other Federal Government agencies or by Federal law.

g. In this NPR, all mandatory actions (i.e., requirements) are denoted by statements containing the term "shall," followed by a software engineering (SWE) requirement number. The terms "may" or "can" denote discretionary privilege or permission, "should" denotes a good practice and is recommended but not required, "will" denotes expected outcome, and "are/is" denotes descriptive material.

h. In this NPR, all document citations are assumed to be the latest version unless otherwise noted.

P.3 Authority

a. The National Aeronautics and Space Act, as amended, 51 U.S.C. § 20113(a).

b. NPD 1000.0, NASA Governance and Strategic Management Handbook.

c. NPD 1000.3, The NASA Organization.

d. NPD 1000.5, Policy for NASA Acquisition.

e. NPD 7120.4, NASA Engineering and Program/Project Management Policy.

P.4 Applicable Documents and Forms

a. NPD 1210.2, NASA Surveys, Audits, and Reviews Policy.

b. NPD 1600.2, NASA Security Policy.

c. NPD 2091.1, Inventions Made By Government Employees.

d. NPD 2800.1, Managing Information Technology.

e. NPR 1600.1, NASA Security Program Procedural Requirements.

f. NPR 2800.2, Information and Communication Technology Accessibility.

g. NPR 2810.1, Security of Information Technology.

h. NPR 7120.5, NASA Space Flight Program and Project Management Requirements.

i. NPR 7120.7, NASA Information Technology and Institutional Infrastructure Program and Project Management Requirements.

j. NPR 7120.8, NASA Research and Technology Program and Project Management Requirements.

k. NPR 8705.2, Human-Rating Requirements for Space Systems.

l. NPR 8705.4, Risk Classification for NASA Payloads.

m. NPR 8715.3, NASA General Safety Program Requirements.

n. NASA-STD-1006, Space System Protection Standard.

o. NASA-STD-8739.8, Software Assurance and Software Safety Standard.

p. NASA-HDBK-2203, NASA Software Engineering Handbook.

P.5 Measurement/Verification

Implementation of this directive is defined as implementing all the identified processes, activities, and requirements in accordance with the software classification and approved software tailoring. Compliance with this NPR is verified by submission of the completed Requirements Mapping Matrix(ces) to responsible NASA officials, including any approved tailoring (see Appendix C) and by internal and external controls. Internal controls processes are defined in NPD 1200.1, NASA

Internal Control. Internal controls include surveys, audits, and reviews conducted in accordance with NPD 1210.2, NASA Surveys, Audits, and Reviews Policy. External controls may include external surveys, audits, and reporting or contractual requirements.

P.6 Cancellation

a. NPR 7150.2C, NASA Software Engineering Requirements, dated August 02, 2019.

b. NASA Interim Directive 7150-113: NASA Interim Directive for Software License Management, dated June 13, 2017.

Chapter 1: Introduction

1.1 Overview

1.1.1 This directive imposes requirements on procedures, design considerations, activities, and tasks used to acquire, develop, maintain, operate, retire, and manage applicable software. This directive is a designed set of requirements for protecting the ’Agency’s investment in software engineering products and fulfilling our responsibility to the citizens of the United States (U.S.).

1.1.2 The requirements in this directive have been extracted from industry standards and proven NASA experience in software engineering. Centers and software developers may show that many of the requirements are satisfied through existing programs, procedures, and processes.

1.1.3 The Agency makes significant investments in software engineering to support the Agency’s investment areas: Space Flight, Aeronautics, Research and Technology, Information Technology (IT), and Institutional Infrastructure. NASA ensures that programs, projects, systems, and subsystems that use software follow a standard set of requirements. One of the goals of this directive is to bring the Agency’s engineering and software development and management communities together to optimize resources and talents across Center boundaries. For NASA to effectively communicate and work seamlessly across Centers, a common framework of generic requirements is needed. This directive fulfills this need for the Agency within the discipline of software engineering.

1.1.4 This directive does not require a specific software life cycle model. Where this NPR refers to phases and milestone reviews in the software life cycle, it uses the standard NASA life cycle models described in NPR 7120.5, NASA Space Flight Program and Project Management Requirements, NPR 7120.7, NASA Information Technology Program and Project Management Requirements, and NPR 7120.8, NASA Research and Technology Program and Project Management Requirements, as supported by milestone reviews described in NPR 7123.1, NASA Systems Engineering Processes and Requirements.

1.1.5 NASA is committed to instituting and updating these requirements to meet the Agency’s current and future challenges in software engineering. Successful experiences are codified in updated versions of this directive after experience has been gained through its use within the NASA software community, the collection of lessons learned from projects, and the implementation records of the Engineering Technical Authorities (ETAs).

1.2 Hierarchy of NASA Software-Related Engineering and Program/Project Documents

1.2.1 Agency-Level Software Policies and Requirements

NPD 7120.4, NASA Engineering and Program/Project Management Policy, is an overarching directive that establishes top-level policies for all software created, acquired, or maintained by or for NASA, including Commercial-off-the-shelf (COTS) software, Government-off-the-shelf (GOTS) software, and Modified-off-the-shelf (MOTS) software and open-source software, embedded software, reused software, legacy software, and heritage software. This directive supports the implementation of NPD 7120.4, and establishes the Agency set of software engineering

requirements for software acquisition, development, maintenance, retirement, operations, and management. It provides a set of software engineering requirements in generic terms for use by NASA, contractors, grant recipients, or parties to agreements. Additional Agency-level project management requirements and systems engineering requirements exist that influence and affect the software development activities on a project. In the event of a conflict between an NPD and NPR, the NPD takes precedence.

1.2.2 Agency-Level Multi-Center and Product Line Requirements (non-software specific)

Existing Agency-Level NPDs and NPRs elaborate, tailor, and in some cases add requirements to those above to address the needs of major multi-Center projects, specific product lines, and specific focus areas. Examples of representative NPRs in this category are NPR 8705.2, Human-Rating Requirements for Space Systems, NPR 8715.3, NASA General Safety Program Requirements, NPR 8735.2, Hardware Quality Assurance Program Requirements for Programs and Projects, NPR 7120.5, NPR7123.1, NPD 2800.1, Managing Information Technology, NPR 2800.2, Information and Communication Technology Accessibility, and NPR 2810.1, Security of Information Technology.

1.2.3 Center-Level Directives or Requirements (related to software)

Center-level directives or requirements are developed by NASA Centers to document their local software policies, requirements, and procedures. These directives are responsive to the higher-level requirements while addressing the specific application areas and the Center's mission within the Agency. In the event of a conflict between this NPR with a Center-level directive, the information provided in this NPR takes precedence.

1.3 Document Structure

1.3.1 Chapter 2 describes the roles, responsibilities, and institutional requirements relevant to the requirements in this directive. This chapter describes the responsibilities for maintaining and advancing organizational capability in software engineering practices to effectively meet the scientific and technological objectives of the Agency. It defines the roles and responsibilities of key officials in software engineering management, the software development and management processes, and the software life cycle management processes. Specific software classification applicability, if any, for the requirements in Chapter 2 are contained in the requirement wording. The requirements in Chapter 2 are not part of the Requirements Mapping Matrix in Appendix C. Approval of any tailoring of requirements designated in Chapter 2 can be done by the appropriate organization per the defined roles and responsibilities.

1.3.2 Chapter 3 establishes software management requirements. The software management activities define and control the many software aspects of a project from beginning to end. The software management activities include the required interfaces to other organizations, determination of deliverables, cost estimates, tracking of schedules, 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 or system plans.

1.3.3 Chapter 4 provides the software engineering life cycle requirements. This directive makes no recommendation for a specific software life cycle model. Each has its strengths and weaknesses, and no one model is best for every situation. Whether using the agile methods, spiral model, the iterative

model, waterfall, or any other development life cycle model, each has its own set of requirements, design, implementation, testing, release to operations, maintenance, and retirement. Although this directive does not impose a particular life cycle model on each software project, it does support a standard set of life cycle phases. Use of the different phases of a life cycle allows the various products of a project to be gradually developed and matured from initial concepts through the fielding of the product and to its final retirement.

1.3.4 Chapter 5 provides supporting software life cycle requirements. Unlike development processes, support processes are not targeted primarily at a specific phase of the project life cycle but typically occur with similar intensity throughout the complete project or product life cycle. For example, normal configuration management baselines (e.g., requirements, code, and products) happen across the life cycle, as does cybersecurity. Support processes are software management and engineering processes that support the entire software life cycle: Software Configuration Management, Risk Management, Peer Reviews, Inspections, Software Measurement, and Non-conformance and Defect Management.

1.3.5 Chapter 6 provides a list of the recommended software records.

1.3.6 Appendix A provides definitions.

1.3.7 Appendix B provides acronyms used in this directive.

1.3.8 Appendix C contains the Requirements Mapping Matrix.

1.3.9 Appendix D contains software classifications.

1.3.10 Appendix E contains software references for this directive.

Chapter 2. Roles, Responsibilities, and Principles Related to Tailoring of the Requirements

2.1 Roles and Responsibilities Associated with this Directive

2.1.1 The NASA Office of the Chief Engineer (OCE).

2.1.1.1 The NASA OCE shall lead and maintain a NASA Software Engineering Initiative to advance software engineering practices. [SWE-002]

2.1.1.2 The NASA OCE shall periodically benchmark each Center's software engineering capability against requirements in this directive. [SWE-004]

> *Note: Capability Maturity Model® Integration (CMMI®) for Development (CMMI®-DEV) appraisals are the preferred benchmarks for objectively measuring progress toward software engineering process improvement at NASA Centers.*

2.1.1.3 The NASA OCE shall periodically review the project requirements mapping matrices. [SWE-152]

2.1.1.4 The NASA OCE shall authorize appraisals against selected requirements in this NPR to check compliance. [SWE-129]

2.1.1.5 The NASA OCE and Center training organizations shall provide training to advance software engineering practices. [SWE-100]

2.1.1.6 The NASA OCE shall maintain an Agency-wide process asset library of applicable best practices and process templates for all size projects. [SWE-098]

2.1.2 NASA Chief, Safety and Mission Assurance (SMA).

2.1.2.1 The NASA Chief, SMA manages Agency software assurance policy and software safety policies and is the Technical Authority (TA) for any requirements in this directive.

2.1.2.2 The NASA Chief, SMA shall lead and maintain a NASA Software Assurance and Software Safety Initiative to advance software assurance and software safety practices. [SWE-208]

2.1.2.3 The NASA Chief, SMA shall periodically benchmark each Center's software assurance and software safety capabilities against the NASA-STD-8739.8, NASA Software Assurance and Software Safety Standard. [SWE-209]

2.1.2.4 The NASA Chief, SMA shall periodically review the project's requirements mapping matrices. [SWE-212]

2.1.2.5 The NASA Chief, SMA shall authorize appraisals against selected requirements in this NPR to check compliance. [SWE-221]

2.1.2.6 The NASA Chief, SMA shall provide for software assurance training. [SWE-222]

2.1.2.7 The NASA Chief, SMA shall make the final decision on all proposed tailoring of SWE-141, the Independent Verification and Validation (IV&V) requirement. [SWE-223]

2.1.3 NASA Chief, Office of Chief Information Officer (OCIO)

2.1.3.1 The NASA OCIO Senior Agency Information Security Officer (SAISO) will conduct security assessments and appraisals on selected requirements in this NPR to check compliance.

2.1.3.2 The NASA OCIO SAISO will participate in any software development reviews, as needed.

2.1.4 Chief Health and Medical Officer (CHMO)

2.1.4.1 The CHMO is the TA for any requirements which impact health and medical aspects.

2.1.4.2 The CHMO has approval authority of tailoring of software with health and medical implications as documented in NPR 7120.11, Health and Medical Technical Authority Implementation.

2.1.5 Center Director

2.1.5.1 In this directive, the phrase “the Center Directors shall...” means that the roles and responsibilities of the Center Directors may be further delegated within the organization consistent with the scope and scale of the system.

2.1.5.2 Center Director, or designee, shall maintain, staff, and implement a plan to continually advance the Center’s in-house software engineering capability and monitor the software engineering capability of NASA’s contractors. [SWE-003]

Note: The recommended practices and guidelines for the content of a Center Software Engineering Improvement Plan are defined in NASA-HDBK-2203, NASA Software Engineering Handbook. Each Center has a current Center Software Engineering Improvement Plan on file in the NASA Chief Engineer’s office.

2.1.5.3 Center Director, or designee, shall establish, document, execute, and maintain software processes per the requirements in this directive. [SWE-005]

2.1.5.4 Center Director, or designee, shall comply with the requirements in this directive that are marked with an “X” in Appendix C. [SWE-140]

Note: The responsibilities for approving changes in the requirements for a project is listed for each requirement in the requirement mapping matrix. When the requirement and software class are marked with an “X,” the projects will record the risk and rationale for any requirement that is not completely implemented by the project. The projects can document their related mitigations and risk acceptance in the approved Requirements Mapping Matrix. Project relief from the applicable cybersecurity requirements, Section 3.11, Software Cybersecurity, has to include an agreement from the SAISO or Center CISO, as designated by the SAISO. The NASA Agency CIO, or Center CIO designee, has institutional authority on all Class F software projects.

2.1.5.5 The Center Director, or designee, shall report on the status of the Center's software engineering discipline, as applied to its projects, upon request by the OCE, OSMA, or OCHMO. [SWE-095]

2.1.5.6 Center Director, or designee, shall maintain a reliable list of their Center's programs and projects containing Class A, B, C, and D software. The list should include: [SWE-006]

a. Project/program name and Work Breakdown Structure (WBS) number.

b. Software name(s) and WBS number(s).

c. Software size estimate (report in Kilo/Thousand Source Lines of Code (KSLOCs)).

d. The phase of development or operations.

e. Software Class or list of the software classes being used on the project.

f. Software safety-critical status.

g. For each Computer Software Configuration Item (CSCI)/Major System containing Class A, B, or C software, provide:

(1) The name of the software development organization.

(2) Title or brief description of the CSCI/Major System.

(3) The estimated total KSLOCs, the CSCI/Major System, represents.

(4) The primary programming languages used.

(5) The life cycle methodology on the software project.

(6) Name of responsible software assurance organization(s).

2.1.5.7 For Class A, B, and C software projects, the Center Director, or designee, shall establish and maintain a software measurement repository for software project measurements containing at a minimum: [SWE-091]

a. Software development tracking data.

b. Software functionality achieved data.

c. Software quality data.

d. Software development effort and cost data.

2.1.5.8 For Class A, B, and C software projects, the Center Director, or designee, shall utilize software measurement data for monitoring software engineering capability, improving software quality, and to track the status of software engineering improvement activities. [SWE-092]

2.1.5.9 Center Director, or designee, will maintain and implement software training to advance its in-house software engineering capabilities.

2.1.5.10 For Class A, B, and C software projects, each Center Director, or designee, shall establish and maintain software cost repository(ies) that contains at least the following measures: [SWE-142]

a. Planned and actual effort and cost.

b. Planned and actual schedule dates for major milestones.

c. Both planned and actual values for key cost parameters that typically include software size, requirements count, defects counts for maintenance or sustaining engineering projects, and cost model inputs.

d. Project descriptors or metadata that typically includes software class, software domain/type, and requirements volatility.

2.1.5.11 Each Center Director, or designee, shall contribute applicable software engineering process assets in use at his/her Centers to the Agency-wide process asset library. [SWE-144]

2.1.5.12 The designated ETA(s) shall define the content requirements for software documents or records. [SWE-153].

> *Note: The recommended practices and guidelines for the content of different types of software activities (whether stand-alone or condensed into one or more project level or software documents or electronic files) are defined in NASA-HDBK-2203. The Center defined content should address prescribed content, format, maintenance instructions, and submittal requirements for all software related records. The designated TA for software approves the required software content for projects within their scope of authority. Electronic submission of data deliverables is preferred. "Software records should be in accordance with NPR 7120.5, NPD 2810.1, NASA Information Security Policy, NPD 2800.1, and NPR 2810.1."*

2.1.5.13 The Center Director, or designee, shall ensure that the Government has clear rights in the software, a Government purpose license, or other appropriate license or permission from third party owners prior to providing the software for internal NASA software sharing or reuse. [SWE-215]

2.1.5.14 The Center Director, or designee, shall ensure that all software listed on the internal software sharing or reuse catalog(s) conforms to NASA software engineering policy and requirements. [SWE-216]

2.1.5.15 The Center Director, or designee, (e.g., the Civil Servant Technical Point of Contact (POC) for the software product) shall perform the following actions: [SWE-217]

a. Keep a list of all contributors to the software product.

b. Ensure that the software product contains appropriate disclaimer and indemnification provisions (e.g., in a "README" file) stating that the software may be subject to U.S. export control restrictions, and it is provided "as is" without any warranty, express or implied, and that the recipient waives any claims against, and indemnifies and holds harmless, NASA and its contractors and subcontractors.

2.1.5.16 The Center Director or designee (e.g., the Civil Servant Technical POC for the software product) shall perform the following actions for each type of internal NASA software transfer or reuse: [SWE-214]

a. A NASA civil servant to a NASA civil servant:

(1) Verify the requesting NASA civil servant has requested and completed an Acknowledgment (as set forth in the note following paragraph 3.10.2e).

(2) Provide the software to the requesting NASA civil servant.

b. A NASA civil servant to a NASA contractor:

(1) Verify a NASA civil servant (e.g., a Contracting Officer (CO) or Contracting Officer Representative (COR)) has confirmed the NASA contractor requires such software for the performance of Government work under their NASA contract and that such performance of work will be a Government purpose. Center Intellectual Property Counsel should be consulted for any questions regarding what is or is not a Government purpose.

(2) Verify a NASA civil servant (e.g., a CO or COR) has confirmed an appropriate Government Furnished Software clause (e.g., 1852.227-88, “Government-furnished computer software and related technical data”) is in the subject contract (or, if not, that such clause is first added); or the contractor may also obtain access to the software in accordance with the external release requirements of NPR 2210.1, Release of NASA Software.

(3) Verify NASA contractor is not a foreign person (as defined by 22 CFR §120.16).

(4) Verify there is a requesting NASA Civil servant (e.g., a CO or COR), and the requesting NASA civil servant has executed an Acknowledgment (as set forth in the note following paragraph 3.10.2e).

(5) After items (1), (2), (3), and (4) are complete, provide the software to the requesting NASA civil servant. The requesting NASA civil servant is responsible for furnishing the software to the contractor pursuant to the subject contract’s terms.

c. A NASA civil servant to any NASA grantees, Cooperative Agreement Recipients or any other agreement partners or to any other entity under U.S. Government Agency Release, Open source Release, Public Release, U.S. Release, Foreign Release:

(1) If the release is to any NASA grantees, Cooperative Agreement Recipients, or any other agreement (e.g., Space Act Agreement) partners or to any other entity under U.S. Government Agency Release, an Open source Release, a Public Release, a U.S. Release, or a Foreign Release, the software release is completed in accordance with the external release requirements of NPR 2210.1, Release of NASA Software – Revalidated w/change 1.

2.1.6 Center SMA Director

2.1.6.1 In this document, the phrase “the Center SMA Director will…” means that the roles and responsibilities of the Center SMA Directors may be further delegated within the organization consistent with the scope and scale of the system. The Center SMA Director designates SMA TAs for programs, facilities, and projects, providing direction, functional oversight, and assessment for all Agency safety, reliability, maintainability, and quality engineering and assurance activities, including Software Assurance.

2.1.6.2 The Center SMA Director will assure the project completes thorough hazard analyses which include software.

Note: The project manager is responsible for assuring Software Safety Hazard Analyses is performed on their project. The PM is responsible for the development of the project's software hazard analyses and its independent review. Any differences in software safety's independent software safety critical determinations will be worked through the ETA and the SMA TA.

2.1.6.3 The Center SMA Director, will review the project's IV&V 'Project Execution Plan (IPEP) to ensure it meets NASA IV&V criteria as defined in NASA-STD-8739.8.

2.1.6.4 The Center SMA Director will support the project to ensure that acquired, developed, and maintained software, as required by SWE-032, is developed by an organization with a non-expired CMMI®-DEV rating as measured by a CMMI® Institute Certified Lead Appraiser.

2.1.6.5 The Center SMA Director will support the Center organizations in obtaining and maintaining the NASA organization's CMMI®-DEV ratings.

2.1.6.6 The Center SMA Director, or designee, will ensure that the project's Requirements Mapping Matrix implementation approach does not impact SMA on the project.

2.1.6.7 The Center SMA Director will ensure that any disagreements between software engineering or the project office and software assurance are identified, reported, tracked, and if not resolved, elevated.

2.1.6.8 The designated SMA TA(s) will review, ensure, and concur on software products and processes throughout the project acquisition, development, delivery, operations, and maintenance.

2.1.7 Contracting Officers

2.1.7.1 Contracting Officers, as defined in FAR 2.101, or Agreement Managers as defined in NAII 1050.3, NASA Partnership Guide, in conjunction with Program/Project Managers shall ensure that the appropriate FAR, NFS, and other provisions/clauses based on this requirements document and NASA-STD-8739.8 are included for all NASA contracts, Space Act Agreements, cooperative agreements, partnership agreements, grants, or other agreements pursuant to which software is being acquired, developed, modified, operated, or managed for NASA. [SWE-218]

2.1.8 Technical Authorities

2.1.8.1 The TA(s) or Institutional Authority(s) for requirements in this NPR will be defined per NPR 7120.5, Section 3.3.

Note: Refer to Appendix C (column titled "Authority") for requirements and their associated Technical or Institutional Authority. NASA HQ will designate the TA for SWE-032 and SWE-141.

2.1.8.2 The technical and institutional authorities for requirements in this directive shall: [SWE-126]

a. Assess projects' requirements mapping matrices and tailoring from requirements in this directive by:

(1) Checking the accuracy of the project's classification of software components against the definitions in Appendix D.

(2) Evaluating the project's Requirements Mapping Matrix for commitments to meet applicable requirements in this directive, consistent with software classification.

(3) Confirming that requirements marked "Not-Applicable" in the project's Requirements Mapping Matrix are not relevant or not capable of being applied.

(4) Determining whether the project's risks, mitigations, and related requests for relief from requirements designated with "X" in Appendix C are reasonable and acceptable.

(5) Approving/disapproving requests for relief from requirements designated with "X" in Appendix C, which falls under this Authority's scope of responsibility.

(6) Facilitating the processing of projects' requirements mapping matrices and tailoring decisions from requirements in this directive, which falls under the responsibilities of a different Authority (see column titled "Authority" in Appendix C).

(7) Include the SAISO (or delegate) in all software reviews to ensure software cybersecurity is included throughout software development, testing, maintenance, retirement, operations, management, acquisition, and assurance activities.

(8) Ensuring that approved requirements mapping matrices, including any tailoring rationale against this directive, are archived as part of retrievable project records.

> *Note: To effectively assess projects' requirements mapping matrices, the designated Center Engineering Technical and Institutional Authorities for this NPR are recognized NASA software engineering experts or utilize recognized NASA software engineering experts in their decision processes. NASA-HDBK-2203 contains valuable information on each requirement, links to relevant NASA Lessons Learned, and guidance on tailoring. Center organizations or branches may also share frequently used tailoring and related common processes.*

b. Indicate the Technical Authority or Technical Authorities approval by signature(s) in the Requirements Mapping Matrix itself, when the Requirements Mapping Matrix is used to tailor from the applicable "X" requirement(s).

> *Note: The Requirements Mapping Matrix documents the requirements that the project plans to meet, "not applicable" requirements, and any tailoring approved by designated Authorities with associated justification. If a project wants to tailor a requirement marked as HQ TA, then the project is required to get NASA HQ approval (e.g., OCE, OSMA, OCIO, or OCHMO) on a tailored request or a software Requirements Mapping Matrix.*

2.2 Principles Related to Tailoring Requirements

2.2.1 Software requirements tailoring is the process used to seek relief from NPR requirements consistent with program or project objectives, acceptable risk, and constraints. To accommodate the

wide variety of software systems and subsystems, application of these requirements to specific software development efforts may be tailored where justified and approved. To effectively maintain control over the application of requirements in this directive and to ensure proposed tailoring from specific requirements are appropriately mitigated, NASA established TA governance. Tailoring from requirements in this directive are governed by the following requirements, as well as those defined in NPD 1000.3, The NASA OrganizationNPD 2800.1, NPR 2810.1, NPR 7120.5, NPR 7120.7, NPR 7120.8, NPR 7120.11 and NPR 8715.3 for all of the Agency's investment areas. The Technical and Institutional Authority for each requirement in this NPR is documented in the "Authority" column of Appendix C. The responsible program, project, or operations manager need to formally accept the tailoring risk. Tailoring decided at the Center level are to consult the Center ETA, Center SMA TA, Center Health and Medical TA, and the NASA CIO's Center IT Authority designee as defined in the requirements mapping matrix. The OSMA has co-approval on any tailoring decided at the HQ level that involves software. The Office of the Chief Medical Officer (OCHMO) has co-approval on any tailoring decided that involves software with health and medical implications. The SAISO, or designee, has co-approval on any tailoring of the cybersecurity requirements in Section 3.11. For tailoring involving human safety risk, the actual risk taker(s) (or official spokesperson[s] and appropriate supervisory chain) need to formally agree to assume the risk. '

2.2.2 This directive establishes a baseline set of requirements to reduce software engineering risks on NASA projects and programs. Appendix C defines the default applicability of the requirements based on software classification. Each project has unique circumstances, and tailoring can be employed to modify the requirements set appropriate for the software engineering effort. Tailoring of requirements is based on key characteristics of the software engineering effort, including acceptable technical and programmatic risk posture, Agency priorities, size, and complexity. Requirements can be tailored more broadly across a group of similar projects, a program, an organization, or other collection of similar software development efforts in accordance with NPR 7120.5, Section 3.5.5.

2.2.3 In this directive, the phrase "the project manager shall..." means the roles and responsibilities of the project manager may be further delegated within the organization to the scope and scale of the system.

2.2.4 Requirements in this directive are invoked by software classifications as defined in Appendix C:

a. "X" – Indicates an invoked requirement by this directive consistent with software classification (ref. SWE-139).

b. Blank – Optional/Not invoked by this directive.

2.2.5 The approval of the Authority designated in Appendix C is required for all tailoring of requirements designated as "X." The implementation approach used to meet each requirement is typically determined by the appropriate software engineering management in conjunction with the project.

Note: The request for relief from a requirement includes the rationale, a risk evaluation, and reference to all material that justifies supporting acceptance. The organization submitting the tailoring request informs the next higher level of involved management in a timely manner of the tailoring request. The dispositioning organization reviews the request

with the other organizations that could be impacted or have a potential risk (i.e., to safety, quality, cybersecurity, health) with the proposed changes; and obtains the concurrence of those organizations.

2.2.6 Requests for software requirements relief at either the Center or HQ TA level (i.e., partial or complete relief) may be submitted in the streamlined form of a Requirements Mapping Matrix. The required signatures from engineering, NASA CIO, and SMA authorities are to be obtained. A required signature from designated SAISO is required for relief of cybersecurity requirements. If the Requirements Mapping Matrix is completed and approved in accordance with NPR 7120.5's direction on Authority and this directive, it meets the requirements for requesting tailoring.

2.2.7 The engineering, CIO, and SMA authorities shall review and agree with any tailored NPR 7150.2 requirements per the requirements mapping matrix authority column. [SWE-150]

2.2.8 If a system or subsystem development evolves to meet a higher or lower software classification as defined in Appendix D, then the project manager shall update their plan(s) and initiate modifications to any supplier contracts to fulfill the applicable requirements per the Requirements Mapping Matrix in Appendix C with approved tailoring. [SWE-021]

Chapter 3: Software Management Requirements

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 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, decommissioning, and supporting activities and processes. The software effort is scoped, the development processes defined, measurements defined, and activities are documented in software planning documents.

3.1.2 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 a 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.

f. Source code available external to NASA.

See the NASA Software Engineering Handbook for additional detail.

3.1.3 The project manager shall develop, maintain, and execute software plans, including security 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. The project should include, or reference in the software development plans, procedures for coordinating the software development and design, and the system or project development life cycle.*

3.1.4 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 are taken, recorded, and managed.

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

3.1.6 The project manager shall establish and maintain the software processes, software documentation plans, list of developed electronic products, deliverables, and list of tasks for the software development that are required for the project's software developers, as well as the action required (e.g., approval, review) of the Government upon receipt of each of the deliverables. [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. The software activities should include plans for software product verification and validation activities, software assurance, methods, environments, and criteria for the project.

3.1.7 The project manager shall define and document the milestones at which the software developer(s) progress will be reviewed and audited. [SWE-037]

3.1.8 The project manager shall require the software developer(s) to periodically report status and provide insight into software development and test activities; at a minimum, the software developer(s) will be required to allow the project manager and software assurance personnel to: [SWE-039]

a. Monitor product integration.

b. Review the verification activities to ensure adequacy.

c. Review trade studies and source data.

d. Audit the software development processes and practices.

e. Participate in software reviews and technical interchange meetings.

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

3.1.10 The project manager shall require the software developer(s) to provide NASA with electronic access to the source code developed for the project in a modifiable format. [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. This requirement should include MOTS software, ground test software, simulations, ground analysis software, ground control software,

science data processing software, hardware manufacturing software, and Class E and Class F software.

3.1.11 The project manager shall comply with the requirements in this NPR that are marked with an "X" in Appendix C consistent with their software classification. [SWE-139]

3.1.12 Where approved, the project manager shall document and reflect the tailored requirement in the plans or procedures controlling the development, acquisition, and deployment of the affected software. [SWE-121]

3.1.13 Each project manager with software components shall maintain a requirements mapping matrix or multiple requirements mapping matrices against requirements in this NPR, including those delegated to other parties or accomplished by contract vehicles or Space Act Agreements. [SWE-125]

Note: A project may have multiple software engineering requirements mapping matrices if needed for multiple software components on a given project.

Note: Project relief from an applicable "X" requirement can be granted only by the designated TAs, Engineering, and SMA, or, for security issues, the NASA CIO. The record of their approval of the tailored requirements in a Requirements Mapping Matrix will be indicated by the Authority signature or signatures in the Requirements Mapping Matrix. The projects will document their related mitigations and risk acceptance in the approved Requirements Mapping Matrix. When the requirement and software class are marked with an "X," the projects record the risk and rationale for any requirements that are entirely or partially relieved in the Requirements Mapping Matrix. The CIO has institutional authority on all Class F software projects.

3.1.14 The project manager shall satisfy the following conditions when a COTS, GOTS, MOTS, OSS, 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, transfer rights, and conditions of use (e.g., required copyright, author, and applicable license notices within the software code, or a requirement to redistribute the licensed software only under the same license (e.g., GNU GPL, ver. 3, license)) have been addressed and coordinated with Center Intellectual Property Counsel.

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.

3.2 Software Cost Estimation

3.2.1 To better estimate the cost of development, the project manager shall establish, document, and maintain: [SWE-015]

a. Two cost estimate models and associated cost parameters for all Class A and B software projects that have an estimated project cost of $2 million or more.

b. One software cost estimate model and associated cost parameter(s) for all Class A and Class B software projects that have an estimated project cost of less than $2 million.

c. One software cost estimate model and associated cost parameter(s) for all Class C and Class D software projects.

d. One software cost estimate model and associated cost parameter(s) for all Class F software projects.

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., programmatic assumptions/constraints, 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, including end state risk and threat assessments for cybersecurity.

e. Includes the cost of the required 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.

3.2.3 The project manager shall submit software planning parameters, including size and effort estimates, milestones, and characteristics, to the Center measurement repository at the conclusion of major milestones. [SWE-174]

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 dependencies for software development activities.

d. Identifies and accounts for dependencies with other projects and cross-program dependencies.

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

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

3.4 Software 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 Assessment

s

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, and F 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 in conjunction with the Requirements Mapping 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 assurance may perform an independent software classification, or concur with engineering's software classification decision. Software engineering and software assurance technical authorities need to agree on the classification of each system and subsystem containing software. If there is a disagreement between the technical authorities, then the dissenting opinion process for your center should be followed.

3.5.2 The project manager shall maintain records of each software classification determination, each software Requirements Mapping Matrix, and the results of each software independent classification assessments for the life of the project. [SWE-176]

3.6 Software Assurance and Software Independent Verification & Validation

3.6.1 The project manager shall plan and implement software assurance, software safety, and IV&V (if required) per NASA-STD-8739.8, Software Assurance and Software Safety Standard. [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. Software Assurance directions, requirements, and guidance can be found in the NASA-STD-8739.8.*

3.6.2 For projects reaching Key Decision Point A, 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, Risk Classification for NASA Payloads.

c. Projects selected explicitly by the Mission Directorate Associate Administrator (MDAA) to have software IV&V.

3.6.3 If software IV&V is required for a project, the project manager, in consultation with NASA IV&V, shall ensure an IPEP is developed, approved, maintained, and executed in accordance with IV&V requirements in NASA-STD-8739.8. [SWE-131]

> *Note: The IV&V Advisory Board will review the scope of NASA IV&V activities on an annual basis as part of the budget planning process.*

3.6.4 If software IV&V is performed on a project, the project manager shall ensure that IV&V is provided access to development artifacts, products, source code, and data required to perform the IV&V analysis efficiently and effectively. [SWE-178]

> *Note: The artifacts and products should be provided electronically in original format (i.e.,*

non-pdf) and, where possible, direct read-only electronic access to project document repositories and data stores should be provided. Appropriate security products should be completed and transferred as part of the overall package.

3.6.5 If software IV&V is performed on a project, the project manager shall provide responses to IV&V submitted issues and risks and track these issues and risks to closure. [SWE-179]

3.7 Safety-Critical Software

3.7.1 The project manager, in conjunction with the SMA organization, shall determine if each software component is considered to be safety-critical per the criteria defined in NASA-STD-8739.8. [SWE-205]

3.7.2 If a project has safety-critical software, the project manager shall implement the safety-critical software requirements contained in NASA-STD-8739.8. [SWE-023]

3.7.3 If a project has safety-critical software or mission-critical software, the project manager shall implement the following items in the software: [SWE-134]

a. The software is initialized, at first start and restarts, to a known safe state.

b. The software safely transitions between all predefined known states.

c. Termination performed by software functions is performed to a known safe state.

d. Operator overrides of software functions require at least two independent actions by an operator.

e. Software rejects commands received out of sequence when execution of those commands out of sequence can cause a hazard.

f. The software detects inadvertent memory modification and recovers to a known safe state.

g. The software performs integrity checks on inputs and outputs to/from the software system.

h. The 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. The software responds to an off-nominal condition within the time needed to prevent a hazardous event.

k. The software provides error handling.

l. The software can place the system into a safe state.

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

3.7.4 If a project has safety-critical software, the project manager shall ensure that there is 100

percent code test coverage using the Modified Condition/Decision Coverage (MC/DC) criterion for all identified safety-critical software components. [SWE-219]

Note: In MC/DC coverage, every condition in a decision is tested independently to reach full coverage. Each condition will be executed twice, once with the results true and once with the results of false, but with no difference in the truth values of all other conditions in the decision. In addition, it will be shown that each condition independently affects the decision. Any deviations from 100 percent should be reviewed and waived with rationale by the TAs approval. It is recommended that someone independent of the developer of the code under test design and perform this testing to ensure requirement interpretation or incorrect assumptions do not escape this testing.

3.7.5 If a project has safety-critical software, the project manager shall ensure all identified safety-critical software components have a cyclomatic complexity value of 15 or lower. Any exceedance shall be reviewed and waived with rationale by the project manager or technical approval authority. [SWE-220]

Note: Cyclomatic complexity is a metric used to measure the complexity of a software program. This metric measures independent paths through the source code. The point of the requirement is to minimize risk, minimize testing, and increase reliability associated with safety-critical software code components, thus reducing the chance of software failure during a hazardous event. The software developer should assess all software safety-critical components with a cyclomatic complexity score over 15 for testability, maintainability, and code quality. For more guidance on this requirement, see NASA-HDBK-2203.

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. Description of the limits and the allowable scope for the use of the auto-generated software.

d. Verification and validation of auto-generated source code using the same software standards and processes as hand-generated 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.

3.8.2 The project manager shall require the software developers and custom software suppliers to provide NASA with electronic access to the models, simulations, and associated data used as inputs

for auto-generation of software. [SWE-206]

Note: The term electronic access includes access to the data from NASA facilities.

3.9 Software Development Processes and Practices

3.9.1 The CMMI® model is an industry-accepted model of software development practices. It is utilized to assess how well NASA projects are supported by software development organization(s) having the necessary skills, practices, and processes in place to produce reliable products within cost and schedule estimates. The CMMI® model 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 the industry state of the practice.

c. Measure and ensure compliance with the intent of the directive's process related requirements using an industry standard approach.

d. Assess internal and external software development organization's processes and practices.

e. Identify potential risk areas within a given organization's software development processes and practices.

3.9.2 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 Certified Lead Appraiser as follows: [SWE-032]

a. For Class A software: CMMI®-DEV Maturity Level 3 Rating or higher 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.

Note: Organizations need to complete an official CMMI® Institute defined appraisal against either the CMMI®-DEV model V1.3 or V2.0. Organizations are to maintain their rating and have their results posted on the CMMI® Institute Website, or provide an Appraisal Disclosure Statement so that NASA can assess the current maturity/capability rating. Software development organizations need to maintain their appraisal rating during the period they are responsible for the development and maintenance of the software. CMMI® ratings can cover a team, a group, a project, a division, or an entire organization.

For Class B software, an exception can be exercised for those cases in which NASA wishes to purchase a product from the "best in class provider," but the best in class provider does not have the required CMMI® rating. For Class B software, instead of a CMMI® rating by a development organization, the project will conduct an evaluation, performed by a qualified evaluator selected by the Center ETA, against the CMMI®-DEV Maturity Level 2 practices, and mitigate any risk, if deficiencies are identified in the evaluation. If this approach is used, the development organization

and project are responsible for correcting the deficiencies identified in the evaluation. When this exception is exercised, the OCE and Center ETA are notified of the proposition and provided the results of the evaluation. The project manager should seek guidance from the Office of Procurement (OP) for help in exercising the exception.

3.10 Software Reuse

3.10.1 The project manager shall specify reusability requirements that apply to its software development activities to enable future reuse of the software, including the models, simulations, and associated data used as inputs for auto-generation of software, for U.S. Government purposes. [SWE-147]

3.10.2 The project manager shall evaluate software for potential reuse by other projects across NASA and contribute reuse candidates to the appropriate NASA internal sharing and reuse software system. However, if the project manager is not a civil servant, then a civil servant will pre-approve all such software contributions; all software contributions should include, at a minimum, the following information: [SWE-148]

a. Software Title.

b. Software Description.

c. The Civil Servant Software Technical POC for the software product.

d. The language or languages used to develop the software.

e. Any third-party code contained therein, and the record of the requisite license or permission received from the third party permitting the Government's use and any required markings (e.g., required copyright, author, applicable license notices within the software code, and the source of each third-party software component (e.g., software URL & license URL)), if applicable.

> *Note: Currently, there are more than one Agency-wide software inventories and repositories, several options can be found in NASA-HDBK-2203. In order to obtain and reuse the internal software reuse candidates from these repositories, NASA civil servants may request a copy by requesting and completing a simple Acknowledgment of Receipt of the software form that identifies any restrictions on NASA's right to use the software, including limiting its use to governmental purposes only. The Civil Servant Software Technical POC for the software product will keep a list of all contributors to the software. Any software shared will contain appropriate disclaimer and indemnification provisions (e.g., in a "README" file) stating that the software may be subject to U.S. export control restrictions, and it is provided "as is" without any warranty, express, or implied and that the recipient waives any claims against, and indemnifies and holds harmless, NASA and its contractors and subcontractors (see paragraph 2.1.5.17).*

f. Release notes.

3.10.3 In accordance with NPD 2091.1, Inventions Made by Government Employees, NASA Civil Servant employees who make an invention embodied by software will submit to NASA a disclosure of such invention. Likewise, such inventions made by NASA contractors will be reported to NASA,

preferably through the NASA electronic New Technology Report (e-NTR) system, pursuant to the terms of their respective contract. Such disclosures are made through the NASA e-NTR system available at http://invention.nasa.gov/.

3.11 Software Cybersecurity

3.11.1 Software defects are a central and critical aspect of computer security vulnerabilities. Software defects with cybersecurity ramifications include implementation bugs such as buffer overflows and design flaws such as inconsistent error handling.

Note: Software security relies on high-quality code development and testing practices (clean code, modular structure, well-defined interfaces) – anything that reduces error rates and opportunities for misinterpretation or error; considers both the development and deployment/operational context for the software; has the ability to rapidly assess, triage, correct, and deploy security-related updates while the software is in deployment/operations.

3.11.2 The project manager shall perform a software cybersecurity assessment on the software components per the Agency security policies and the project requirements, including risks posed by the use of COTS, GOTS, MOTS, OSS, or reused software components. [SWE-156]

3.11.3 The project manager shall identify cybersecurity risks, along with their mitigations, in flight and ground software systems and plan the mitigations for these systems. [SWE-154]

Note: Project Protection Plans describe the program's approach for planning and implementing the requirements for information, physical, personnel, industrial, and counterintelligence/counterterrorism security, and for security awareness/education requirements in accordance with NPR 1600.1, NASA Security Program Procedural Requirements, NPD 1600.2, the NASA Security Policy, NPD 2810.1, and NPR 2810.1. Include provisions in the plan to protect personnel, facilities, mission-essential infrastructure, and critical program information from potential threats and vulnerabilities that may be identified during the threat and vulnerability assessment process.

3.11.4 The project manager shall implement protections for software systems with communications capabilities against unauthorized access per the requirements contained in the NASA-STD-1006, Space System Protection Standard. [SWE-157]

3.11.5 The project manager shall test the software and record test results for the required software cybersecurity mitigation implementations identified from the security vulnerabilities and security weaknesses analysis. [SWE-159]

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

3.11.6 The project manager shall identify, record, and implement secure coding practices. [SWE-207]

3.11.7 The project manager shall verify that the software code meets the project's secure coding standard by using the results from static analysis tool(s). [SWE-185]

Note: If a static analysis tool will not work with the selected coding standard, other methods are acceptable, including manual inspection.

3.11.8 The project manager shall identify software requirements for the collection, reporting, and storage of data relating to the detection of adversarial actions. [SWE-210]

Note: Monitoring of key software observables (e.g., number of failed login attempts, performance changes, internal communication changes) is needed to detect adversarial actions that threaten mission success. When an adversarial action occurs, it should be reported. Raw event data should be further analyzed to determine whether an anomalous event represents an attack, and if so, the nature of the attack.

3.12 Software Bi-Directional Traceability

3.12.1 The project manager shall perform, record, and maintain bi-directional traceability between the following software elements: [SWE-052]

Table 1. Bi-directional traceability by software classification

Bi-directional Traceability	Class A, B, and C	Class D	Class F
Higher-level requirements to the software requirements	X		X
Software requirements to the system hazards	X	X	
Software requirements to the software design components	X		
Software design components to the software code	X		
Software requirements to the software verification(s)	X	X	X
Software requirements to the software non-conformances	X	X	X

Note: The project manager will maintain bi-directional traceability between the software requirements and software-related system hazards, including hazardous controls, hazardous mitigations, hazardous conditions, and hazardous events.

Chapter 4: Software Engineering Life Cycle Requirements

4.1 Software Requirements

4.1.1 The requirements phase is one of the most critical phases of software engineering. Studies show that the top problems in the software industry are due to poor requirements elicitation, inadequate requirements specification, and inadequate management of changes to requirements. Requirements provide the foundation for the entire life cycle, as well as for the software product. Requirements also provide a basis for planning, estimating, and monitoring. Requirements are based on customer, user, and other stakeholder needs and design and development constraints. The development of requirements includes elicitation, analysis, documentation, verification, and validation. Ongoing customer validation of the requirements to ensure the end products meet customer needs is an integral part of the life cycle process. Customer validation can be accomplished via rapid prototyping and customer-involved reviews of iterative and final software requirements.

4.1.2 The project manager shall establish, capture, record, approve, and maintain software requirements, including requirements for COTS, GOTS, MOTS, OSS, or reused software components, as part of the technical specification. [SWE-050]

> *Note: The software technical requirements definition process is used to transform the baselined stakeholder expectations into unique, quantitative, and measurable technical software requirements that can be used for defining a design solution for the software end products and related enabling products. This process also includes validation of the requirements to ensure that the requirements are well-formed (clear and unambiguous), complete (agrees with customer and stakeholder needs and expectations), consistent (conflict free), and individually verifiable and traceable to a higher level requirement. Recommended content for a software specification can be found in NASA-HDBK-2203.*

4.1.3 The project manager shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements, safety and reliability analyses, and the hardware specifications and design. [SWE-051]

4.1.4 The project manager shall include software related safety constraints, controls, mitigations, and assumptions between the hardware, operator, and software in the software requirements documentation. [SWE-184]

4.1.5 The project manager shall track and manage changes to the software requirements. [SWE-053]

4.1.6 The project manager shall identify, initiate corrective actions, and track until closure inconsistencies among requirements, project plans, and software products. [SWE-054]

4.1.7 The project manager shall perform requirements validation to ensure that the software will perform as intended in the customer environment. [SWE-055]

4.2 Software Architecture

4.2.1 Experience confirms that the quality and longevity of a software-reliant system is primarily determined by its architecture. The software architecture underpins a system's software design and code; it represents the earliest design decisions, ones that are difficult and costly to change later. The transformation of the derived and allocated requirements into the software architecture results in the basis for all software development work.

4.2.2 A software architecture:

a. Formalizes precise subsystem decompositions.

b. Defines and formalizes the dependencies among software work products within the integrated system.

c. Serves as the basis for evaluating the impacts of proposed changes.

d. Maintains rules for use by subsequent software engineers that ensure a consistent software system as the work products evolve.

e. Provides a stable structure for use by future groups through the documentation of the architecture, its views and patterns, and its rules.

f. Follows guidelines created by the NASA Space Asset and the Enterprise Protection Program to protect mission architectures.

g. Documents the valid and invalid modes or states of operation within the software requirements.

4.2.3 The project manager shall transform the requirements for the software into a recorded software architecture. [SWE-057]

> *Note: A documented software architecture that describes: the software's structure; identifies the software qualities (i.e., performance, modifiability, and security); identifies the known interfaces between the software components and the components external to the software (both software and hardware); identifies the interfaces between the software components and identifies the software components. Reference NASA's Software Architecture Review Board (SARB) paper NTRS ID 20160005787, "Quality Attributes for Mission Flight Software: A Reference for Architects."*

4.2.4 The project manager shall perform a software architecture review on the following categories of projects: [SWE-143]

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.

4.3 Software Design

4.3.1 Software design is the process of defining the software architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements. The software architecture is the fundamental organization of a system embodied in its components, their relationships to each other and the environment, and the principles guiding its design and evolution. The software architectural design is concerned with creating a strong overall structure for software entities that fulfill the allocated system and software-level requirements. Typical views captured in an architectural design include the decomposition of the software subsystem into design entities, computer software configuration items, definitions of external and internal interfaces, dependency relationships among entities and system resources, and finite state machines. The design should be further refined into lower-level entities that permit the implementation by coding in a programming language. Typical attributes that are documented for lower-level entities include the identifier, type, purpose, function, constraints, subordinates, dependencies, interface, resources, processing, and data. Rigorous specification languages, graphical representations, and related tools have been developed to support the evaluation of critical properties at the design level. Projects are encouraged to take advantage of these improved design techniques to prevent and eliminate errors as early in the life cycle as possible. Software, developed or purchased, has additional requirements to comply with from Section 508 of the Rehabilitation Act, as defined inNPR 2800.2.

4.3.2 The project manager shall develop, record, and maintain a software design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested. [SWE-058]

4.4 Software Implementation

4.4.1 Software implementation consists of implementing the requirements and design into code, data, and records. Software implementation also consists of following coding methods and standards. Unit testing is also usually a part of software implementation (unit testing can also be conducted during the testing phase).

4.4.2 The project manager shall implement the software design into software code. [SWE-060]

4.4.3 The project manager shall select, define, and adhere to software coding methods, standards, and criteria. [SWE-061]

4.4.4 The project manager shall use static analysis tools to analyze the code during the development and testing phases to, at a minimum, detect defects, software security, code coverage, and software complexity. [SWE-135]

> *Note: Although no maximum cyclomatic complexity score is required for non-safety critical software, all software projects should regularly collect and maintain complexity metrics and use them to manage risk, either when high-complexity code must be modified, or proactively to improve the overall quality and maintenance of the code base. For safety critical software, the analysis should take into account the requirements for cyclomatic complexity and code coverage as defined in 3.7.5 and 3.7.4 respectively.*

4.4.5 The project manager shall unit test the software code. [SWE-062]

Note: For safety critical software, the unit testing should follow the requirement established in 3.7.4 of this document.

4.4.6 The project manager shall assure that the unit test results are repeatable. [SWE-186]

4.4.7 The project manager shall provide a software version description for each software release. [SWE-063]

4.4.8 The project manager shall validate and accredit the software tool(s) required to develop or maintain software. [SWE-136]

Note: All software development tools contain some number of software defects. Validation and accreditation of the critical software development and maintenance tools ensure that the tools being used during the software development life cycle do not generate or insert errors in the software executable components. Software tool accreditation is the certification that a software tool is acceptable for use for a specific purpose. Accreditation is conferred by the organization best positioned to make the judgment that the software tool in question is acceptable. The likelihood that work products will function properly is enhanced, and the risk of error is reduced if the tools used in the development and maintenance processes have been validated and accredited themselves.

4.5 Software Testing

4.5.1 The purpose of testing is to verify the software functionality and remove defects. Testing verifies the code against the requirements and the design to ensure that the requirements are implemented. Testing also identifies problems and defects that are corrected and tracked to closure before product delivery. Testing also validates that the software operates appropriately in the intended environment. Please note for Class A software there are additional software test requirements and software integration requirements as defined in NPR 8705.2.

4.5.2 The project manager shall establish and maintain: [SWE-065]

a. Software test plan(s).

b. Software test procedure(s).

c. Software test(s), including any code specifically written to perform test procedures.

d. Software test report(s).

4.5.3 The project manager shall test the software against its requirements. [SWE-066]

Note: A best practice for Class A, B, and C software projects is to have formal software testing planned, conducted, witnessed, and approved by an independent organization outside of the development team.

4.5.4 The project manager shall place software items under configuration management prior to testing. [SWE-187]

Note: This includes the software components being tested and the software components being used to test the software, including components such as support software, models, simulations, ground support software, COTS, GOTS, MOTS, OSS, or reused software components.

4.5.5 The project manager shall evaluate test results and record the evaluation. [SWE-068]

4.5.6 The project manager shall use validated and accredited software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment. [SWE-070]

Note: Information regarding specific V&V techniques and the analysis of models and simulations can be found in NASA-STD-7009, Standard for Models and Simulations, NASA-HDBK-7009, Handbook for Models and Simulations, or discipline-specific recommended practice guides.

4.5.7 The project manager shall update the software test and verification plan(s) and procedure(s) to be consistent with software requirements. [SWE-071]

4.5.8 The project manager shall validate the software system on the targeted platform or high-fidelity simulation. [SWE-073]

Note: Typically, a high-fidelity simulation has the exact processor, processor performance, timing, memory size, and interfaces as the target system.

4.5.9 The project manager shall ensure that the code coverage measurements for the software are selected, implemented, tracked, recorded, and reported. [SWE-189]

Note: This requirement can be met by running unit, integration, and validation tests; measuring the code coverage; and achieving the code coverage by additional (requirement based) tests, inspection, or analysis.

If the project does not get 100 percent structural coverage, it means one of four things and each requires action on the project manager's part:

• Requirement missing - the code that hasn't been covered is performing an essential activity, but no requirement indicates that this should be done;

• Test missing - the code that hasn't been covered relates to an existing requirement, but no test was implemented for it;

• Extraneous/dead code – the code that hasn't been covered is not traceable to any requirement and isn't needed by the software;

• Deactivated code - the code that hasn't been covered isn't traceable to any requirements for the

current system, but is intended to be executed in specific configurations.

The code coverage data and any rationale for uncovered code should be presented and reviewed at major project milestones.

4.5.10 The project manager shall verify code coverage is measured by analysis of the results of the execution of tests. [SWE-190]

Note: If it can be justified that the required percentage cannot be achieved by test execution, the analysis, inspection, or review of design can be applied to the non-covered code. The goal of the complementary analysis is to assess that the non-covered code behavior is as expected.

4.5.11 The project manager shall plan and conduct software regression testing to demonstrate that defects have not been introduced into previously integrated or tested software and have not produced a security vulnerability. [SWE-191]

4.5.12 The project manager shall verify through test the software requirements that trace to a hazardous event, cause, or mitigation technique. [SWE-192]

4.5.13 The project manager shall develop acceptance tests for loaded or uplinked data, rules, and code that affects software and software system behavior. [SWE-193]

Note: These acceptance tests should validate and verify the data, rules, and code for nominal and off-nominal scenarios.

4.5.14 The project manager shall test embedded COTS, GOTS, MOTS, OSS, or reused software components to the same level required to accept a custom developed software component for its intended use. [SWE-211]

4.6 Software Operations, Maintenance, and Retirement

4.6.1 Planning for operations, maintenance, and retirement are typically considered throughout the software life cycle. Operational concepts and scenarios are derived from customer requirements and validated in the operational or simulated environment. Software maintenance activities sustain the software product after the product is delivered to the customer until retirement.

4.6.2 The project manager shall plan and implement software operations, maintenance, and retirement activities. [SWE-075]

4.6.3 The project manager shall complete and deliver the software product to the customer with appropriate records, including as-built records, to support the operations and maintenance phase of the software's life cycle. [SWE-077]

4.6.4 The project manager shall complete, prior to delivery, verification that all software requirements identified for this delivery have been met or dispositioned, that all approved changes have been implemented, and that all defects designated for resolution prior to delivery have been resolved. [SWE-194]

4.6.5 The project manager shall maintain the software using standards and processes per the applicable software classification throughout the maintenance phase. [SWE-195]

4.6.6 The project manager shall identify the records and software tools to be archived, the location of the archive, and procedures for access to the products for software retirement or disposal. [SWE-196]

Chapter 5: Supporting Software Life Cycle Requirements

5.1 Software Configuration Management (SCM)

5.1.1 Software Configuration Management (SCM) is the process of applying configuration management throughout the software life cycle to ensure the completeness and correctness of software configuration items. SCM applies technical and administrative direction and surveillance to identify and record the functional and physical characteristics of software configuration items, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements. SCM establishes and maintains the integrity of the products of a software project throughout the software life cycle. Use of standard Center or organizational SCM processes and procedures is encouraged where applicable.

5.1.2 The project manager shall develop a software configuration management plan that describes the functions, responsibilities, and authority for the implementation of software configuration management for the project. [SWE-079]

5.1.3 The project manager shall track and evaluate changes to software products. [SWE-080]

5.1.4 The project manager shall identify the software configuration items (e.g., software records, code, data, tools, models, scripts) and their versions to be controlled for the project. [SWE-081]

> *Note: The items to be controlled include tools, items, or settings used to develop the software, which could impact the software. Examples of such items include compiler/assembler versions, makefiles, batch files, and specific environment settings.*

5.1.5 The project manager shall establish and implement procedures to: [SWE-082]

a. Designate the levels of control through which each identified software configuration item is required to pass.

b. Identify the persons or groups with authority to authorize changes.

c. Identify the persons or groups to make changes at each level.

> *Note: IEEE 828-2012, IEEE Standard for Configuration Management in Systems and Software Engineering describes configuration management processes to be established, how they are to be accomplished, who is responsible for doing specific activities, when they are to happen, and what specific resources are required. It addresses configuration management activities over a product's life cycle. Configuration management in systems and software engineering is a specialty discipline within the larger discipline of configuration management. Configuration management is essential to systems engineering and software engineering.*

5.1.6 The project manager shall prepare and maintain records of the configuration status of software configuration items. [SWE-083]

5.1.7 The project manager shall perform software configuration audits to determine the correct version of the software configuration items and verify that they conform to the records that define them. [SWE-084]

5.1.8 The project manager shall establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products. [SWE-085]

5.1.9 The project manager shall participate in any joint NASA/developer audits. [SWE-045]

5.2 Software Risk Management

The project manager shall record, analyze, plan, track, control, and communicate all of the software risks and mitigation plans. [SWE-086]

Note: Project managers should be aware of any risks that remain after mitigations have been completed or after a risk has been accepted.

5.3 Software Peer Reviews and Inspections

5.3.1 Software peer reviews and inspections are the in-process technical examination of work products by peers to find and eliminate defects early in the life cycle. Software peer reviews and inspections are performed following defined procedures covering the preparation for the review, the review itself is conducted, results are recorded, results are reported, and completion criteria is certified. When planning the composition of a software peer review or inspection team, consider including software testing, system testing, software assurance, software safety, software cybersecurity, and software IV&V personnel.

5.3.2 The project manager shall perform and report the results of software peer reviews or software inspections for: [SWE-087]

a. Software requirements.

b. Software plans, including cybersecurity.

c. Any design items that the project identified for software peer review or software inspections according to the software development plans.

d. Software code as defined in the software and or project plans.

e. Software test procedures.

Note: Software peer reviews or software inspections are recommended best practices for all safety and mission-success related software components. Recommended best practices and guidelines for software formal inspections are contained in NASA-STD-8739.9, Software Formal Inspection Standard.

5.3.3 The project manager shall, for each planned software peer review or software inspection: [SWE-088]

a. Use a checklist or formal reading technique (e.g., perspective-based reading) to evaluate the work products.

b. Use established readiness and completion criteria.

c. Track actions identified in the reviews until they are resolved.

d. Identify the required participants.

5.3.4 The project manager shall, for each planned software peer review or software inspection, record necessary measurements. [SWE-089]

5.4 Software Measurements

5.4.1 Software measurement is a primary tool for managing software processes and evaluating the quality of software products. Analysis of measures provides insight into the capability of the software organization and identifies opportunities for software process and product improvements. Software measurement programs at multiple levels are established to meet measurement objectives. The requirements below are designed to reinforce the use of measurement at the project, Center software organization, and NASA Chief Engineer levels to assist in managing projects, assuring quality, and improving software engineering practices. Measurement programs are designed to meet the following goals:

a. Improve future software planning and software cost estimation.

b. Describe and record information about a software product during its life cycle.

c. Assist usability and maintainability of a software product.

d. Monitor and control software life cycle processes.

e. Communicate information about the system, software product, or service.

f. Provide a history, including lessons learned, during the development and maintenance to support management and process improvement.

g. Provide evidence that the processes were followed.

h. Provide indicators of software quality.

i. Track the status of software engineering improvement and assurance programs.

j. Report the status of software engineering improvements and assurance programs to Center software organizations and Center SMA.

5.4.2 The project manager shall establish, record, maintain, report, and utilize software management and technical measurements. [SWE-090]

Note: The NASA-HDBK-2203 contains a set of candidate management indicators that may be used on a software development project. The NASA Chief Engineer may identify and document additional Center measurement objectives, software measurements, collection procedures and guidelines, and analysis procedures for selected software projects and software development organizations. The software measurement process includes collecting software technical measurement data from the project's software developer(s).

5.4.3 The project manager shall analyze software measurement data collected using documented project-specified and Center/organizational analysis procedures. [SWE-093]

5.4.4 The project manager shall provide access to the software measurement data, measurement analyses, and software development status as requested to the sponsoring Mission Directorate, the NASA Chief Engineer, the Center Technical Authorities, HQ SMA, and other organizations as appropriate. [SWE-094]

5.4.5 The project manager shall monitor measures to ensure the software will meet or exceed performance and functionality requirements, including satisfying constraints. [SWE-199]

Note: The metrics could include planned and actual use of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage device capacity, and communications/network equipment capacity, bus traffic, partition allocation) over time. As part of the verification of the software detailed design, the developer will update the estimation of the technical resource metrics. As part of the verification of the coding, testing, and validation, the technical resource metrics will be updated with the measured values and will be compared to the margins.

5.4.6 The project manager shall collect, track, and report software requirements volatility metrics. [SWE-200]

5.5 Software Non-conformance or Defect Management

5.5.1 The project manager shall track and maintain software non-conformances (including defects in tools and appropriate ground software). [SWE-201]

5.5.2 The project manager shall define and implement clear software severity levels for all software non-conformances (including tools, COTS, GOTS, MOTS, OSS, reused software components, and applicable ground systems). [SWE-202]

Note: At a minimum, classes should include loss of life or loss of vehicle, mission success, visible to the user with operational workarounds, and an 'other' class that does not meet previous criteria.

5.5.3 The project manager shall implement mandatory assessments of reported non-conformances for all COTS, GOTS, MOTS, OSS, and/or reused software components. [SWE-203]

Note: This includes operating systems, run-time systems, device drivers, code generators, compilers, math libraries, and build and Configuration Management (CM) tools. It should be performed pre-flight, with mandatory code audits for critical defects.

5.5.4 The project manager shall implement process assessments for all high-severity software non-conformances (closed loop process). [SWE-204]

Chapter 6: Recommended Software Records Content

6.1 Software Engineering Products

It is possible to prepare a plan, associated procedures, and reports, as well as numerous records, requests, descriptions, and specifications for each software development life cycle process. When deciding how to prepare any of these items, consider the users of the information first. Reviewing and understanding the requirements, needs, and background of users and stakeholders are essential to applying the recommendations for the content of software records defined in NASA-HDBK-2203. Specific content within these records may not apply to every project. Use of NASA Center and contractor formats in document deliverables is acceptable if the required content (as defined by the project) is addressed. Product records should be reviewed and updated as necessary. Typical software engineering products or electronic data include:

a. Software Development Plan/Software Management Plan.

b. Software Schedule.

c. Software Cost Estimate.

d. Software Configuration Management Plan.

e. Software Change Reports.

f. Software Test Plans.

g. Software Test Procedures.

h. Software Test Reports.

i. Software Version Description Reports.

j. Software Maintenance Plan.

k. Software Assurance Plan(s).

l. Software Safety Plan.

m. Software Requirements Specification.

n. Software Data Dictionary.

o. Software and Interface Design Description (Architectural Design).

p. Software Design Description.

q. Software User’s Manual.

r. Records of Continuous Risk Management for Software.

s. Software Measurement Analysis Results.

t. Record of Software Engineering Trade-off Criteria & Assessments (make/buy decision).

u. Software Acceptance Criteria and Conditions.

v. Software Status Reports.

w. Programmer's/Developer's Manual.

x. Software Reuse Report.

y. Software Model and Simulation Data and Documentation, including the Verification, Validation, and Credibility Plan for Software Model and Simulation.

6.2 Software Engineering Product Content

The recommendations for the content of software records are defined in NASA-HDBK-2203. The Software Engineering handbook also provides guidance regarding when these records should be drafted, baselined, and updated. Examples and templates for these records and datasets are on the Software Process Across NASA (SPAN) Web site, accessible at https://nen.nasa.gov/web/software/wiki .

Appendix A. Definitions

Accredit. The official acceptance of a software development tool, model, or simulation (including associated data) to use for a specific purpose.

Analysis. The post-processing or interpretation of the individual values, arrays, files of data, or execution information. It is a careful study of something to learn about its parts, what they do, and how they are related to each other.

Assure. When personnel makes certain that the specified software engineering, software management, and software assurance activities have been performed by others.

Bidirectional Traceability. Association among two or more logical entities that are discernible in either direction (to and from an entity). (ISO/IEC/IEEE 24765, Systems and software engineering – Vocabulary)

Code coverage. The percentage of the software that has been executed (covered) by the test suite.

Commercial Off-the-Shelf Software (COTS). The software product is available for purchase and use without the need to conduct development activities. COTS solutions, as opposed to custom-developed solutions, are typically readily available in the commercial marketplace and ready for use as purchased.

Computer. A functional unit that can perform substantial computations, including numerous arithmetic operations and logic operations.

Computer Software Configuration Item. An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Computer System. A system containing one or more computers and associated software. (Source: ISO/IEC/IEEE 24765)

Condition. (1) Measurable qualitative or quantitative attribute that is stipulated for a requirement and that indicates a circumstance or event under which a requirement applies (Systems and software engineering--Systems and software assurance--Part 1: Concepts and vocabulary ISO/IEC/IEEE 15026-1:2019, 3.1.5), (2) Description of a contingency to be considered in the representation of a problem or a reference to other procedures to be considered as part of the condition (Information processing -- Specification of single-hit decision tables, ISO 5806:1984, 3.6), and (3) Boolean expression containing no Boolean operators (Software and systems engineering -- Software testing -- Part 4: Test techniques, ISO/IEC/IEEE 29119-4:2015, 4.6).

Contracted Software. Software created for a project by a contractor or subcontractor.

Cybersecurity. The protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide confidentiality, integrity, and availability.

Cyclomatic Complexity. Cyclomatic complexity is a software metric used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a function’s source code.

Data. Information for computer processing (e.g., numbers, text, images, and sounds in a form that is suitable for storage in or processing by a computer).

Defect. Any occurrence in a software product that is determined to be incomplete or incorrect relative to the software requirements, expectations for the software, and/or program standards. (Source: NASA-STD-8739.9)

Embedded Computer System. A computer system that is part of a larger system and performs some of the requirements of that system. (Source: ISO/IEC/IEEE 24765)

Embedded Software. Software that is part of a larger system and performs some of the requirements of that system. (Source: ISO/IEC/IEEE 24765)

Ensure. To secure or guarantee, to make sure or certain.

Establish and Maintain. Formulation, documentation, use/deployment, and current maintenance of the object (usually a document, requirement, process, or policy) by the responsible project, organization, or individual.

Failure. The behavior of the software or system component when a fault is encountered, producing an incorrect or undesired effect of a specified severity. (Source: NASA-STD-8739.9)

Fault. The manifestation of an error in software that may cause a failure. (Source: NASA-STD-8719.24 Annex)

Freeware. Software that is proprietary and that is available for use at no monetary cost. In other words, freeware may be used without payment but may usually not be modified, re-distributed, or reverse-engineered without the author's permission.

Glueware. Software created to connect the off-the-shelf software/reused software with the rest of the system. It may take the form of software that modifies interfaces or add missing functionality, "firewalls" that isolate the off-the-shelf software, or software that check inputs and outputs to the off-the-shelf software and may modify to prevent failures.

Government Off-the-Shelf Software. Government Off-the-Shelf Software refers to Government-created software, usually from another project. The software was not created by the current developers (see software reuse). Usually, the source code is included and documentation, including test and analysis results, is available; (e.g., the Government is responsible for the Government off-the-shelf (GOTS) software to be incorporated into another system).

Independent Verification and Validation (IV&V). Verification and validation performed by an organization that is technically, managerially, and financially independent of the development organization. (Source: ISO/IEC/IEEE 24765) The NASA requirements for IV&V are defined in the NASA-STD-8739.8.

Information Technology. Any equipment or interconnected system or subsystem of equipment that is used in the automatic acquisition, storage, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information by an executive agency. IT also includes computers, ancillary equipment (including imaging peripherals, input, output, and storage devices necessary for security and surveillance), peripheral equipment designed to be controlled by the central processing unit of a computer; software; firmware; and similar procedures, services (including support services), and related resources, but does not include any equipment acquired by a Federal contractor incidental to a Federal contract. (Source: NPR 7120.7)

Insight. An element of Government surveillance that monitors contractor compliance using Government-identified metrics and contracted milestones. Insight is a continuum that can range from low intensity such as reviewing quarterly reports to high intensity such as performing surveys and reviews. (Source: NPR 7123.1)

Legacy and Heritage Software. Software products (architecture, code, requirements) written specifically for one project and then, without prior planning during its initial development, found to be useful for other projects. See software reuse.

Major Engineering/Research Facility. Used in this document to show research, development, test, or simulation facilities representing a significant NASA investment (facilities with a Current Replace Value equal to or greater than 50 million dollars) which contains software that supports programs and projects managed under NPR 7120.5, NPR 7120.7, or NPR 7120.8, and that have a Mission Dependency Index value equal to or greater than 70.

MC/DC. Modified condition/decision coverage (MC/DC) is a code coverage criterion used in software testing. MC/DC requires all of the below during testing: Each condition in a decision is shown to independently affect the outcome of the decision. Independence of a condition is shown by proving that only one condition changes at a time.

Mission Critical. Item or function that should retain its operational capability to assure no mission failure (i.e., for mission success - meeting all mission objectives and requirements for performance and safety). (Source: NPR 8715.3)

Mission Critical Software. Software that can cause, contribute to, or mitigate the loss of capabilities that are essential to the primary mission objectives.

Mobile Application. A mobile application is an application built using native code for the device or a software Web application that is distributed through the device specific marketplace. Web applications presented via a mobile browser are not considered mobile applications.

Model. A description or representation of a system, entity, phenomena, or process. (Source: NASA-STD-7009) Only for this document, the term “model” refers to models implemented in software.

Modified Off-the-Shelf Software (MOTS). When COTS or legacy and heritage software is reused, or heritage software is changed, the product is considered “modified.” The changes can include all or part of the software products and may involve additions, deletions, and specific alterations. An argument can be made that any alterations to the code and design of an off-the-shelf software component constitute “modification,” but the common usage allows for some percentage (less than five percent of the code changes) of change before the off-the-shelf software is declared to be modified off-the-shelf (MOTS) software. Modified Off-the-Shelf Software may include the changes to the application shell or glueware to add or protect against certain features and not to the off-the-shelf software system code directly. When less than 30 percent of the existing code changes, the product can be considered “modified.” If more than 30 percent of the code changes or if the new code is added, the software should be considered a new software development.

Off-the-Shelf Software. Software not developed in-house or by a contractor for the specific project now underway. The software is developed for a purpose different from the current project. Used in practice as an umbrella for COTS, GOTS, MOTS, OSS, freeware, shareware, trial software, demonstration software, legacy software, heritage software, and reuse software.

Open-Source Software. Software where its human-readable source code is made broadly available without cost under an OSS license, which provides conditions for use, reuse, modification/improvement, and redistribution; and often where the software development, management, and planning is done publicly, or easily observable by an individual or organization not previously connected with its open source project.

Primary Mission Objectives. Outcomes expected to be accomplished, which are closely associated with the reason the mission was proposed, funded, developed, and operated (e.g., objectives related to top-level requirements or their flow down).

Process Asset Library. A collection of process asset holdings that may be used by an organization or project. (Source: CMMI® for Systems Engineering/Software Engineering/Integrated Product and Process Development Supplier Sourcing.)

Program. A strategic investment by a Mission Directorate or Mission Support Office that has a defined architecture and technical approach, requirements, funding level, and a management structure that initiates and directs one or more projects. A program defines a strategic direction that the Agency has identified as critical.

Programmable Logic Device. A semiconductor device based on a matrix of configurable logic blocks connected via a configurable interconnect. The circuitry (combinational/sequential logic, memory/storage, input/output) in a PLD is configured to meet design requirements for a desired application after device manufacturing.

Project. A specific investment having defined goals, objectives, requirements, life cycle cost, a beginning, and an end. A project yields new or revised products or services that directly address NASA's strategic needs. They may be performed wholly in-house; by Government, industry, academia partnerships; or through contracts with private industry.

Project Manager. A generic term that represents the position in charge of the project. A project manager could be designated as a project lead, project principal investigator, project scientist, research director, project executive, or some other term, as defined in the project's governing document. A project manager is responsible for the formulation and implementation of the R&T project, per the governing document in coordination with the program manager. (NPR 7120.5 defines the roles and responsibilities for this position).

Requirements Volatility. The total number of requirements compared to requirement changes over time. It may include additions, changes, and reduction of requirements.

Risk Management. An organized, systematic decision-making process that efficiently identifies, analyzes, plans, tracks, controls, communicates, and documents risk to increase the likelihood of achieving program/project goals. (Source: NPR 8715.3)

Safety Critical. A term describing any condition, event, operation, process, equipment, or system that could cause or lead to severe injury, major damage, or mission failure if performed or built improperly, or allowed to remain uncorrected. (Source NPR 8715.3)

Scripts. A sequence of automated computer commands embedded in a program that tells the program to execute a specific procedure (e.g., files with monitoring, logic, or commands used by software to automate a process or procedure).

Simulation. The imitation of the behavioral characteristics of a system, entity, phenomenon, or

process. (Source: NASA-STD-7009) Only for the purpose of this document, the term “simulation” refers to only those simulations that are implemented in software.

Shareware. Software that is available free of charge and often distributed informally for evaluation, after which a fee may be requested for continued use.

Software. In this directive, “software” is defined as (1) computer programs, procedures, and associated documentation and data pertaining to the operation of a computer system (IEEE 828-2012, 2.1), (2) all or a part of the programs, procedures, rules, and associated documentation of an information processing system (ISO/IEC 19770-5:2015, Information technology, 3.34), (3) program or set of programs used to run a computer (ISO/IEC 26514:2008, Systems and software engineering–requirements for designers and developers of user documentation, 4.46) (4) all or part of the programs which process or support the processing of digital information (ISO/IEC 19770-1:2017, Information technology – IT asset management – Part 1: IT asset management systems--Requirements, 3.49), and (5) part of a product that is the computer program or the set of computer programs (ISO/IEC/IEEE 26513:2017, Systems and software engineering–requirements for testers and reviewers of information for users, 3.34). This definition applies to software developed by NASA, software developed for NASA, software maintained by or for NASA, COTS, GOTS, MOTS, OSS, reused software components, auto-generated code, embedded software, the software executed on processors embedded in programmable logic devices (see NASA-HDBK-4008), legacy, heritage, applications, freeware, shareware, trial or demonstration software, and open-source software components.

Software Architecture. The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the properties of those components, and the relationships between them. The term also refers to documentation of a system’s software architecture. Documenting software architecture facilitates communication between stakeholders, documents early decisions about high-level design, and allows reuse of design components and patterns between projects.

Software Assurance. The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. For NASA, this includes the disciplines of Software Quality (functions of Software Quality Engineering, Software Quality Assurance, and Software Quality Control), Software Safety, Software Reliability, Mission Software Cybersecurity Assurance, Software Verification and Validation, and IV&V.

Software Engineering. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software (i.e., the application of engineering to software). (Source: ISO/IEC/IEEE 24765)

Software Item. Source code, object code, control code, control data, or a collection of these items.

Software Maintenance. (1) Totality of activities required to provide cost-effective support to a software system. (2) Entitlement of additional rights (such as additional functionality, upgrade, or support) for a previously granted software entitlement. (3) A set of services a Publisher can sell to a Customer for the ongoing development and delivery of software bug fixes and product upgrades.

Software Peer Review and Inspection. A visual examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications. (Source: IEEE 1028). Refer to NASA-STD-8739.9 for guidelines for software peer reviews or inspections.

Software Reuse. A software product developed for one use but having other uses or one developed specifically to be usable on multiple projects or in multiple roles on one project. Examples include, but are not limited to, COTS products, acquirer-furnished software products, software products in reuse libraries, and pre-existing developer software products.

Software Suppliers. An organization or individual that enters into an agreement with the acquirer for the supply of a software product or service or individual or organization that enters into a contract with the acquirer for the supply of a software system, software product, or software service under the terms of the contract or an organization or part of an organization or individual that enters into an agreement with the application management organization for the supply of a software product or software service. Software Suppliers includes NASA in-house software development.

Software Validation. Confirmation that the product, as provided (or as it will be provided), fulfills its intended use. In other words, validation ensures that "you built the right thing." (Source: IEEE 1012, IEEE Standard for Software Verification and Validation)

Software Verification. Confirmation that products properly reflect the requirements specified for them. In other words, verification ensures that "you built it right." (Source: IEEE 1012)

Static Analysis. The process of evaluating a system or component based on its form, structure, content, or documentation. (Source: ISO/IEC/IEEE 24765)

Subsystem. A secondary or subordinate system within a larger system. (Source: ISO/IEC/IEEE 24765)

System. The combination of elements that function together to produce the capability required to meet a need. The elements include hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (Source: NPR 7123.1)

Tailoring. The process used to adjust a prescribed requirement to accommodate the needs of a specific task or activity (e.g., program or project). Tailoring may result in changes, subtractions, or additions to a typical implementation of the requirement.

Uncertainty. (1) The estimated amount or percentage by which an observed or calculated value may differ from the true value. (2) A broad and general term used to describe an imperfect state of knowledge or a variability resulting from a variety of factors including, but not limited to, lack of knowledge, the applicability of information, physical variation, randomness or stochastic behavior, indeterminacy, judgment, and approximation. (Source: NPR 8000.4, Agency Risk Management Procedural Requirements).

Unit Test. (1) Testing of individual routines and modules by the developer or an independent tester (ISO/IEC/IEEE 24765). (2) A test of individual programs or modules in order to ensure that there are no analysis or programming errors (ISO/IEC 2382-20). (3) Test of individual hardware or software units or groups of related units. (ISO/IEC/IEEE 24765)

Validation. (1) Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled (ISO/IEC 25000:2014 Systems and software Engineering--Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Guide to SQuaRE, 4.41) (ISO/IEC/IEEE 12207:2017 Systems and software engineering--Software life cycle processes, 3.1.71) (ISO/IEC/IEEE 15288:2015 Systems and software engineering--System life cycle processes, 4.1.53) (ISO/IEC TS 24748-1:2016 Systems and software engineering--Life cycle management--Part 1: Guide for life cycle management, 2.61), (2)

process of providing evidence that the system, software, or hardware and its associated products satisfy requirements allocated to it at the end of each life cycle activity, solve the right problem (e.g., correctly model physical laws, implement business rules, and use the proper system assumptions), and satisfy intended use and user needs (IEEE 1012-2016, IEEE Standard for Software Verification and Validation, 3.1.35), (3) the assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. (A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide) -- Fifth Edition), and (4) process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements (IEEE 1012-2016, IEEE Standard for Software Verification and Validation, 3.1) Note: Validation in a system life cycle context is the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals, and objectives. The right system has been built. Validation demonstrates that the system can be used by the users for their specific tasks. “Validated” is used to designate the corresponding status. [ISO 9000] Multiple validations can be carried out if there are different intended uses.

Appendix B. Acronyms

CAD/CAM	Computer-Aided Design/and Computer-Aided Manufacturing
CE	Chief Engineer
CHMO	Chief Health and Medical Officer
CIO	Chief Information Officer
CMMI®	Capability Maturity Model® Integration
CMMI®-DEV	Capability Maturity Model® Integration® (CMMI®) for evelopment
CMU	Carnegie Mellon University
CO	Contracting Officer
COR	Contracting Officer Representative
COTS	Commercial-Off-the-Shelf
CSCI	Computer Software Configuration Item
CSMA	Chief, Safety and Mission Assurance
EDL	Entry, Descent, and Landing
ETA	Engineering Technical Authority
EVA	Extra Vehicular Activity
FAR	Federal Acquisition Regulations
FFRDC	Federally Funded Research and Development Center
GOTS	Government-Off-the-Shelf
HDBK	Handbook
IEEE	Institute of Electrical and Electronics Engineers
IPEP	IV&V Project Execution Plan
IT	Information Technology
IV&V	Independent Verification and Validation
JPL	Jet Propulsion Laboratory
KSLOC	Kilo/Thousand Source Lines of Code
MC/DC	Modified Condition/Decision Coverage
MDAA	Mission Directorate Associate Administrator
MOTS	Modified Off-the-Shelf
NASA	National Aeronautics and Space Administration
NPD	NASA Policy Directive

NPR	NASA Procedural Requirements
NTR	New Technology Report
OCE	Office of the Chief Engineer
OCHMO	Office of Chief Health and Medical Officer
OCIO	Office of Chief Information Officer
OP	Office of Procurement
OSMA	Office of Safety and Mission Assurance
OSS	Open-Source Software
PLD	Programmable Logic Devices
POC	Point of Contact
SAISO	Senior Agency Information Security Officer
SCM	Software Configuration Management
SEI	Software Engineering Institute
SMA	Safety and Mission Assurance
SOW	Statement of Work
SPAN	Software Process Across NASA
SWE	Software Engineering
TA	Technical Authority
US	United States
WBS	Work Breakdown Structure

Appendix C. Requirements Mapping Matrix

C.1 The rationale for the requirements is contained in the NASA-HDBK-2203. Programs/Projects may substitute a matrix that documents their mapping with their particular Center's implementation of NPR 7150.2, if applicable. See NASA-HDBK-2203 for requirements mapping matrices organized by class, tailoring field for each requirement, tailoring rationale, and approval signature lines.

C.2 The Requirements Mapping Matrix documents the program/project's mappings or intent to comply with the requirements of this NPR or justification for tailoring. The matrix lists:

a. The section reference.

b. The unique requirement identifier.

c. The NPR 7150.2 requirement statement.

d. The Authority Level responsible for assessing a project's requirements mapping matrices and any requested tailoring from requirements in this NPR. The CIO, or the designee, has institutional authority on all Class F software projects and has joint responsibility on the cybersecurity requirements in Section 3.11.

e. The applicability of the requirements in this NPR to specific systems and subsystems within the Agency's investment areas, programs, and projects is determined through the use of the NASA-wide definition of software classes.

C.3 Tailoring Guidance

X - Indicates an invoked requirement by this NPR consistent with Software Classification (ref. SWE-139). May be tailored with TA approval (ref. Chapter 2.2).

Blank - Optional/Not invoked by this NPR.

Center - Center Director or the Center Director's designated ETA, the Center Director's designated SMA TA, and the CHMO designated for Health and Medical TA. The NASA CIO, or the designee, has institutional authority on all Class F software projects and has joint responsibility with the ETA on the cybersecurity requirements in Section 3.11 per the direction in the Requirements Mapping Matrix.

CIO - The NASA CIO, or the designee Center CIO, has institutional authority on all Class F software projects and has joint responsibility with the ETA on the cybersecurity requirements in section 3.11, per the direction in the Requirements Mapping Matrix.

Each requirement marked 'X' for the project's software classification(s) should be addressed in the Requirements Mapping Matrix. All requirements can be tailored per the guidance in this directive. Requirements that are not applicable to a given project, such as the IV&V requirements, should be tailored out in the Requirements Mapping Matrix with justification.

Table 2. Requirements Mapping Matrix

Section	SWE #	Requirement Text	Class A-E Authority	A	B	C	D	E		Class F Authority	F
3.0		Software Management Requirements									
3.1		Software Life Cycle Planning									
3.1.2	033	The project manager shall assess options for software acquisition versus development.	Center	X	X	X	X	X		CIO	X
3.1.3	013	The project manager shall develop, maintain, and execute software plans, including security plans, that cover the entire software life cycle and, as a minimum, address the requirements of this directive with approved tailoring.	Center	X	X	X	X	X		CIO	X
3.1.4	024	The project manager shall track the actual results and performance of software activities against the software plans. 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 are taken, recorded, and managed.	Center	X	X	X	X			CIO	X
3.1.5	034	The project manager shall define and document the acceptance criteria for the software.	Center	X	X	X	X			CIO	X
3.1.6	036	The project manager shall establish and maintain the software processes, software documentation plans, list of developed electronic products, deliverables, and list of tasks	Center	X	X	X	X			CIO	X

		for the software development that are required for the project's software developers, as well as the action required (e.g., approval, review) of the Government upon receipt of each of the deliverables.								
3.1.7	037	The project manager shall define and document the milestones at which the software developer(s) progress will be reviewed and audited.	Center	X	X	X	X		CIO	X
3.1.8	039	The project manager shall require the software developer(s) to periodically report status and provide insight into software development and test activities; at a minimum, the software developer(s) will be required to allow the project manager and software assurance personnel to: a. Monitor product integration. b. Review the verification activities to ensure adequacy. c. Review trades studies and source data. d. Audit the software development processes and practices.e. Participate in software reviews and technical interchange meetings.	Center	X	X	X	X		CIO	X
3.1.9	040	The project manager shall require the software developer(s) to provide NASA with software products, traceability, software change tracking information and nonconformances in electronic format, including software development and management metrics.	Center	X	X	X	X		CIO	X

3.1.10	042	The project manager shall require the software developer(s) to provide NASA with electronic access to the source code developed for the project in a modifiable format.	Center	X	X	X	X	X		CIO	X
3.1.11	139	The project manager shall comply with the requirements in this NPR that are marked with an “X” in Appendix C consistent with their software classification.	Center	X	X	X	X	X		CIO	X
3.1.12	121	Where approved, the project manager shall document and reflect the tailored requirement in the plans or procedures controlling the development, acquisition, and deployment of the affected software.	Center	X	X	X	X	X		CIO	X
3.1.13	125	Each project manager with software components shall maintain a requirements mapping matrix or multiple requirements mapping matrices against requirements in this NPR, including those delegated to other parties or accomplished by contract vehicles or Space Act Agreements.	Center	X	X	X	X	X		CIO	X
3.1.14	027	The project manager shall satisfy the following conditions when a COTS, GOTS, MOTS, OSS, or reused software component is acquired or used: 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	Center	X	X	X	X			CIO	X

		rights, ownership, warranty, licensing rights, transfer rights, and conditions of use (e.g., required copyright, author, and applicable license notices within the software code, or a requirement to redistribute the licensed software only under the same license (e.g., GNU GPL, ver. 3, license)) have been addressed and coordinated with Center Intellectual Property Counsel. 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.									
3.2		**Software Cost Estimation**									
3.2.1	015	To better estimate the cost of development, the project manager shall establish, document, and maintain: a. Two cost estimate models and associated cost parameters for all Class A and B software projects that have an estimated project cost of $2 million or more. b. One software cost estimate model and associated cost parameter(s) for all Class A and Class B software projects that have an estimated project cost of less than $2 million.	Center	X	X	X	X				X

		c. One software cost estimate model and associated cost parameter(s) for all Class C and Class D software projects. d. One software cost estimate model and associated cost parameter(s) for all Class F software projects.								
3.2.2	151	The project manager's software cost estimate(s) shall satisfy the following conditions: a. Covers the entire software life cycle. b. Is based on selected project attributes (e.g., programmatic assumptions/constraints, 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, including end state risk and threat assessments for cybersecurity. e. Includes the cost of the required software assurance support. f. Includes other direct costs.	Center	X	X	X	X		CIO	X
3.2.3	174	The project manager shall submit software planning parameters, including size and effort estimates, milestones, and characteristics, to the Center measurement repository at the conclusion of major milestones.	Center	X	X	X	X			
3.3		Software Schedules								

3.3.1	016	The project manager shall document and maintain a software schedule that satisfies the following conditions: 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 dependencies for software development activities. d. Identifies and accounts for dependencies with other projects and cross-program dependencies.	Center	X	X	X	X			CIO	X
3.3.2	018	The project manager shall regularly hold reviews of software schedule activities, status, performance metrics, and assessment/analysis results with the project stakeholders and track issues to resolution.	Center	X	X	X				CIO	X
3.3.3	046	The project manager shall require the software developer(s) to provide a software schedule for the project's review and schedule updates as requested.	Center	X	X	X	X			CIO	X
3.4		**Software Training**									
3.4.1	017	The project manager shall plan, track, and ensure project specific software training for project personnel.	Center	X	X	X				CIO	X
3.5		Software Classification Assessments									

3.5.1	020	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, and F software in Appendix D.	Center	X	X	X	X	X		CIO	X
3.5.2	176	The project manager shall maintain records of each software classification determination, each software Requirements Mapping Matrix, and the results of each software independent classification assessments for the life of the project.	Center	X	X	X	X	X		CIO	X
3.6		Software Assurance and Software Independent Verification & Validation									
3.6.1	022	The project manager shall plan and implement software assurance, software safety, and IV&V (if required) per NASA-STD-8739.8, Software Assurance and Software Safety Standard.	Center	X	X	X	X	X			
3.6.2	141	For projects reaching Key Decision Point A, the program manager shall ensure that software IV&V is performed on the following categories of projects: 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 selected explicitly by the NASA Chief, Safety and Mission Assurance to have	HQ OSMA	X	X						

		software IV&V.									
3.6.3	131	If software IV&V is performed on a project, the project manager shall ensure an IPEP is developed, approved, maintained, and executed in accordance with the IV&V criteria defined in NASA-STD-8739.8.	Center	X	X						
3.6.4	178	If software IV&V is performed on a project, the project manager shall ensure that IV&V is provided access to development artifacts, products, source code, and data required to perform the IV&V analysis efficiently and effectively.	Center	X	X						
3.6.5	179	If software IV&V is performed on a project, the project manager shall provide responses to IV&V submitted issues and risks, and track these issues and risks to closure.	Center	X	X						
3.7		**Safety-critical Software**									
3.7.1	205	The project manager, in conjunction with the SMA organization, shall determine if each software component is considered to be safety-critical per the criteria defined in NASA-STD-8739.8.	Center	X	X	X	X	X			
3.7.2	023	If a project has safety-critical software, the project manager shall implement the safety-critical software requirements contained in NASA-STD-8739.8.	Center	X	X	X	X				

3.7.3	134	If a project has safety-critical software or mission-critical software, the project manager shall implement the following items in the software: a. The software is initialized, at first start and restarts, to a known safe state. b. The software safely transitions between all predefined known states. c. Termination performed by software functions is performed to a known safe state. d. Operator overrides of software functions require at least two independent actions by an operator. e. Software rejects commands received out of sequence when execution of those commands out of sequence can cause a hazard. f. The software detects inadvertent memory modification and recovers to a known safe state. g. The software performs integrity checks on inputs and outputs to/from the software system. h. The 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. The software responds to an off-nominal condition within the time needed to prevent a hazardous event. k. The software provides error handling.	Center	X	X	X	X				

		handling. l. The software can place the system into a safe state.								
3.7.4	219	If a project has safety-critical software, the project manager shall ensure that there is 100 percent code test coverage using the Modified Condition/Decision Coverage (MC/DC) criterion for all identified safety-critical software components.	Center	X	X	X	X			
3.7.5	220	If a project has safety-critical software, the project manager shall ensure all identified safety-critical software components have a cyclomatic complexity value of 15 or lower. Any exceedance shall be reviewed and waived with rationale by the project manager or technical approval authority.	Center	X	X	X	X			
3.8		**Automatic Generation of Software Source Code**								
3.8.1	146	The project manager shall define the approach to the automatic generation of software source code including: a. Validation and verification of auto-generation tools. b. Configuration management of the auto-generation tools and associated data. c. Description of the limits and the allowable scope for the use of the auto-generated software. d. Verification and validation of auto-generated source code using the same software standards and processes as hand-generated code. e. Monitoring the actual use of auto-generated source code compared to the planned use.	Center	X	X	X			CIO	X

		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.								
3.8.2	206	The project manager shall require the software developers and custom software suppliers to provide NASA with electronic access to the models, simulations, and associated data used as inputs for auto-generation of software.	Center	X	X	X	X		CIO	X
3.9		**Software Development Processes and Practices**								
3.9.2	032	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 Certified Lead Appraiser as follows: a. For Class A software: CMMI®-DEV Maturity Level 3 Rating or higher 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.	HQ OCE and HQ OSMA	X	X					
3.10		**Software Reuse**								
3.10.1	147	The project manager shall specify reusability requirements that apply to its software development activities to enable future reuse	Center	X	X	X	X		CIO	X

		activities to enable future reuse of the software, including the models, simulations, and associated data used as inputs for auto-generation of software, for U.S. Government purposes.									
3.10.2	148	The project manager shall evaluate software for potential reuse by other projects across NASA and contribute reuse candidates to the appropriate NASA internal sharing and reuse software system. However, if the project manager is not a civil servant, then a civil servant will pre-approve all such software contributions; all software contributions should include, at a minimum, the following information: a. Software Title. b. Software Description. c. The Civil Servant Software Technical POC for the software product. d. The language or languages used to develop the software. e. Any third party code contained therein and the record of the requisite license or permission received from the third party permitting the Government's use and any required markings (e.g., required copyright, author, applicable license notices within the software code, and the source of each third-party software component (e.g., software URL & license URL)), if applicable. f. Release notes.	Center	X	X	X	X	X		CIO	X
3.11		**Software Cybersecurity**									

3.11.2	156	The project manager shall perform a software cybersecurity assessment on the software components per the Agency security policies and the project requirements, including risks posed by the use of COTS, GOTS, MOTS, OSS, or reused software components.	Center and Center CIO	X	X	X	X	X		CIO	X
3.11.3	154	The project manager shall identify cybersecurity risks, along with their mitigations, in flight and ground software systems and plan the mitigations for these systems.	Center and Center CIO	X	X	X	X				
3.11.4	157	The project manager shall implement protections for software systems with communications capabilities against unauthorized access per the requirements contained in the Space System Protection Standard, NASA-STD-1006.	Center and Center CIO	X	X	X	X				
3.11.5	159	The project manager shall test the software and record test results for the required software cybersecurity mitigation implementations identified from the security vulnerabilities and security weaknesses analysis.	Center and Center CIO	X	X	X	X			CIO	X
3.11.6	207	The project manager shall identify, record, and implement secure coding practices.	Center and Center CIO	X	X	X	X				
3.11.7	185	The project manager shall verify that the software code meets the project's secure coding standard by using the results from static analysis tool(s).	Center and Center CIO	X	X	X	X			CIO	X

3.11.8	210	The project manager shall identify software requirements for the collection, reporting, and storage of data relating to the detection of adversarial actions.	Center	X	X	X			CIO	X
3.12		**Software Bi-Directional Traceability**								
3.12.1	052	The project manager shall perform, record, and maintain bi-directional traceability between the following software elements: (See Table in 3.12.1)	Center	X	X	X	X		CIO	X
4.0		**Software Engineering (Life Cycle) Requirements**								
4.1		**Software Requirements**								
4.1.2	050	The project manager shall establish, capture, record, approve, and maintain software requirements, including requirements for COTS, GOTS, MOTS, OSS, or reused software components, as part of the technical specification.	Center	X	X	X	X		CIO	X
4.1.3	051	The project manager shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements, safety and reliability analyses, and the hardware specifications and design.	Center	X	X	X				
4.1.4	184	The project manager shall include software related safety constraints, controls, mitigations and assumptions between the hardware, operator, and software in the software requirements documentation.	Center	X	X	X				

4.1.5	053	The project manager shall track and manage changes to the software requirements.	Center	X	X	X	X			CIO	X
4.1.6	054	The project manager shall identify, initiate corrective actions, and track until closure inconsistencies among requirements, project plans, and software products.	Center	X	X	X	X			CIO	X
4.1.7	055	The project manager shall perform requirements validation to ensure that the software will perform as intended in the customer environment.	Center	X	X	X	X			CIO	X
4.2		**Software Architecture**									
4.2.3	057	The project manager shall transform the requirements for the software into a recorded software architecture.	Center	X	X	X					
4.2.4	143	The project manager shall perform a software architecture review on the following categories of projects: 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.	Center	X	X	X					
4.3		**Software Design**									
4.3.2	058	The project manager shall develop, record, and maintain a software design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested.	Center	X	X	X					
4.4		**Software Implementation**									

4.4.2	060	The project manager shall implement the software design into software code.	Center	X	X	X				CIO	X
4.4.3	061	The project manager shall select, define, and adhere to software coding methods, standards, and criteria.	Center	X	X	X	X			CIO	X
4.4.4	135	The project manager shall use static analysis tools to analyze the code during the development and testing phases to, at a minimum, detect defects, software security, code coverage, and software complexity.	Center	X	X	X	X			CIO	X
4.4.5	062	The project manager shall unit test the software code.	Center	X	X	X	X			CIO	X
4.4.6	186	The project manager shall assure that the unit test results are repeatable.	Center	X	X	X	X			CIO	X
4.4.7	063	The project manager shall provide a software version description for each software release.	Center	X	X	X	X			CIO	X
4.4.8	136	The project manager shall validate and accredit the software tool(s) required to develop or maintain software.	Center	X	X	X					
4.5		**Software Testing**									
4.5.2	065	The project manager shall establish and maintain: a. Software test plan(s). b. Software test procedure(s). c. Software test(s), including any code specifically written to perform test procedures. d. Software test report(s).	Center	X	X	X	X			CIO	X
4.5.3	066	The project manager shall test the software against its requirements.	Center	X	X	X	X			CIO	X

4.5.4	187	The project manager shall place software items under configuration management prior to testing.	Center	X	X	X				CIO	X
4.5.5	068	The project manager shall evaluate test results and record the evaluation.	Center	X	X	X	X			CIO	X
4.5.6	070	The project manager shall use validated and accredited software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment.	Center	X	X	X					
4.5.7	071	The project manager shall update the software test and verification plan(s) and procedure(s) to be consistent with software requirements.	Center	X	X	X	X			CIO	X
4.5.8	073	The project manager shall validate the software system on the targeted platform or high-fidelity simulation.	Center	X	X	X					
4.5.9	189	The project manager shall ensure that the code coverage measurements for the software are selected, implemented, tracked, recorded, and reported.	Center	X	X	X	X				
4.5.10	190	The project manager shall verify code coverage is measured by analysis of the results of the execution of tests.	Center	X	X	X					
4.5.11	191	The project manager shall plan and conduct software regression testing to demonstrate that defects have not been introduced into previously integrated or tested software and have not produced a security vulnerability.	Center	X	X	X				CIO	X

4.5.12	192	The project manager shall verify through test the software requirements that trace to a hazardous event, cause, or mitigation technique.	Center	X	X	X	X			
4.5.13	193	The project manager shall develop acceptance tests for loaded or uplinked data, rules, and code that affects software and software system behavior.	Center	X	X					CIO	X
4.5.14	211	The project manager shall test embedded COTS, GOTS, MOTS, OSS, or reused software components to the same level required to accept a custom developed software component for its intended use.	Center	X	X	X				
4.6		Software Operations, Maintenance, and Retirement								
4.6.2	075	The project manager shall plan and implement software operations, maintenance, and retirement activities.	Center	X	X	X	X			CIO	X
4.6.3	077	The project manager shall complete and deliver the software product to the customer with appropriate records, including as-built records, to support the operations and maintenance phase of the software's life cycle.	Center	X	X	X	X			CIO	X
4.6.4	194	The project manager shall complete, prior to delivery, verification that all software requirements identified for this delivery have been met or dispositioned, that all approved changes have been implemented and that all defects designated for resolution prior to delivery	Center	X	X	X	X			CIO	X

		have been resolved.								
4.6.5	195	The project manager shall maintain the software using standards and processes per the applicable software classification throughout the maintenance phase.	Center	X	X	X	X		CIO	X
4.6.6	196	The project manager shall identify the records and software tools to be archived, the location of the archive, and procedures for access to the products for software retirement or disposal.	Center	X	X	X	X		CIO	X
5.0		**Supporting Software Life Cycle Requirements**								
5.1		**Software Configuration Management**								
5.1.2	079	The project manager shall develop a software configuration management plan that describes the functions, responsibilities, and authority for the implementation of software configuration management for the project.	Center	X	X	X	X		CIO	X
5.1.3	080	The project manager shall track and evaluate changes to software products.	Center	X	X	X	X		CIO	X
5.1.4	081	The project manager shall identify the software configuration items (e.g., software records, code, data, tools, models, scripts) and their versions to be controlled for the project.	Center	X	X	X	X		CIO	X
5.1.5	082	The project manager shall establish and implement procedures to: a. Designate the levels of control through which each identified software	Center	X	X	X	X		CIO	X

		identified software configuration item is required to pass. b. Identify the persons or groups with authority to authorize changes. c. Identify the persons or groups to make changes at each level.								
5.1.6	083	The project manager shall prepare and maintain records of the configuration status of software configuration items.	Center	X	X	X	X			CIO	X
5.1.7	084	The project manager shall perform software configuration audits to determine the correct version of the software configuration items and verify that they conform to the records that define them.	Center	X	X	X	X			CIO	X
5.1.8	085	The project manager shall establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products.	Center	X	X	X	X			CIO	X
5.1.9	045	The project manager shall participate in any joint NASA/developer audits.	Center	X	X	X				CIO	X
5.2		**Software Risk Management**									
5.2	086	The project manager shall record, analyze, plan, track, control, and communicate all of the software risks and mitigation plans.	Center	X	X	X				CIO	X
5.3		**Software Peer Reviews/Inspections**									
5.3.2	087	The project manager shall perform and report the results of software peer reviews or software inspections for: a. Software requirements. b. Software plans, including	Center	X	X	X				CIO	X

		cybersecurity. c. Any design items that the project identified for software peer review or software inspections according to the software development plans. d. Software code as defined in the software and or project plans. e. Software test procedures.									
5.3.3	088	The project manager shall, for each planned software peer review or software inspection: a. Use a checklist or formal reading technique (e.g., perspective based reading) to evaluate the work products. b. Use established readiness and completion criteria. c. Track actions identified in the reviews until they are resolved. d. Identify the required participants.	Center	X	X	X					
5.3.4	089	The project manager shall, for each planned software peer review or software inspection, record necessary measurements.	Center	X	X	X				CIO	X
5.4		**Software Measurements**									
5.4.2	090	The project manager shall establish, record, maintain, report, and utilize software management and technical measurements.	Center	X	X	X					
5.4.3	093	The project manager shall analyze software measurement data collected using documented project-specified and Center/organizational analysis procedures.	Center	X	X	X					

5.4.4	094	The project manager shall provide access to the software measurement data, measurement analyses, and software development status as requested to the sponsoring Mission Directorate, the NASA Chief Engineer, the Center TAs, HQ SMA, and other organizations as appropriate.	Center	X	X	X					
5.4.5	199	The project manager shall monitor measures to ensure the software will meet or exceed performance and functionality requirements, including satisfying constraints.	Center	X	X	X					
5.4.6	200	The project manager shall collect, track, and report software requirements volatility metrics.	Center	X	X						
5.5		**Software Non-conformance or Defect Management**									
5.5.1	201	The project manager shall track and maintain software non-conformances (including defects in tools and appropriate ground software).	Center	X	X	X	X			CIO	X
5.5.2	202	The project manager shall define and implement clear software severity levels for all software non-conformances (including tools, COTS, GOTS, MOTS, OSS, reused software components, and applicable ground systems).	Center	X	X	X				CIO	X
5.5.3	203	The project manager shall implement mandatory assessments of reported non-conformances for all COTS, GOTS, MOTS, OSS, and/or reused software components.	Center	X	X	X					

components.

5.5.4	204	The project manager shall implement process assessments for all high severity software non-conformances (closed loop process).	Center	X	X						

Appendix D. Software Classifications

D.1 The applicability of requirements in this directive to specific systems and subsystems containing software is determined through the use of the NASA-wide software classification structure. 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. Classes A through E cover engineering-related software. Class F cover Business and Information Technology Infrastructure Software. Using the Requirements Mapping Matrix, the number of applicable requirements and their associated rigor are scaled back for lower software classes. Situations in which a project contains separate systems and subsystems having different software classes are not uncommon.

D.2 For a given system or subsystem, software is expected to be uniquely defined within a single class. If more than one software class appears to apply, then assign the higher of the classes to the system/subsystem. Any potential discrepancies in classifying software within Classes A through-- E are to be resolved using the definitions and the five underlying factors listed in the previous paragraph. Engineering and SMA provide dual TA chains for resolving classification issues. The NASA Chief Engineer is the ultimate TA for software classification disputes concerning definitions in this NPR.

Class A: Human Rated Space Software Systems

a. Definition:

1. Human Space Flight Software Systems*: Ground and flight software systems developed or operated by or for NASA needed to perform a primary mission objective of human space flight and directly interact with human space flight systems. Limited to software required to perform "vehicle, crew, or primary mission function," as defined by software that is:

(a) Required to operate the vehicle or space asset (e.g., spacesuit, rover, or outpost), including commanding of the vehicle or asset.

(b) Required to sustain a safe, habitable1 environment for the crew.

(c) Required to achieve the primary mission objectives.

(d) Required to directly prepare resources (e.g., data, fuel, power) that are consumed by the above functions.

*Includes software involving launch, on-orbit, in space, surface operations, entry, descent, and landing.

1 Current standards that address habitability and environmental health, including atmospheric composition and pressure, air, and water quality and monitoring, acceleration, acoustics, vibration, radiation, thermal environment, combined environmental effects, and human factors, are documented in NASA-Standard-3001 Volume 1, Space Flight Human-System Standard: Crew Health, NASA-Standard-3001 Volume 2, Space Flight Human-System Standard: Human Factors, Habitability, and Environmental Health, FAA HFDS - Human Factors Design Standard.

b. Examples:

Examples of Class A software (human-rated space flight) include, but are not limited to, the mission phases listed below.

1. During Launch:

Abort modes and selection; separation control; range safety; crew interface (display and controls); crew escape; critical systems monitoring and control; guidance, navigation, and control; and communication and tracking.

2. On-Orbit/In Space:

Extravehicular activity (EVA); control of electrical power; payload control (including suppression of hazardous satellite and device commands); critical systems monitoring and control; guidance, navigation, and control; life support systems; crew escape; rendezvous and docking; failure detection, isolation and recovery; communication and tracking; and mission operations.

3. On Ground:

Pre-launch and launch operations; Mission Control Center (and Launch Control Center) front-end processors; spacecraft commanding; vehicle processing operations; re-entry operations; flight dynamics simulators used for ascent abort calls; and launch and flight controller stations for human-crewed spaceflight.

4. Entry, Descent, and Landing (EDL):

Command and control; aero-surface control; power; thermal; fault protection; and communication and tracking.

5. Surface Operations:

Planet/lunar surface EVA and communication and tracking.

c. Exclusions:

Class A does not include:

1. Software that happens to fly in space but is superfluous to mission objectives (e.g., software contained in an iPod carried on board by an astronaut for personal use); or

2. Software that exclusively supports aeronautics, research and technology, and science conducted without space flight applications; or

3. Systems (e.g., simulators, emulators, facilities) used to test Class A systems containing software in a development environment.

Class B: Non-Human Space Rated Software Systems or Large Scale Aeronautics Vehicles

a. Definitions:

1. Space Systems involve flight and ground software that should perform reliably to accomplish primary mission objectives or major function(s) in non-human space rated systems. Included is software involving launch, on orbit, in space, surface operations, entry, descent, and landing. These

systems are limited to software that is:

(a) Required to operate the vehicle or space asset (e.g., orbiter, lander, probe, flyby spacecraft, rover, launch vehicle, or primary instrument) such as commanding of the vehicle or asset;

(b) Required to achieve the primary mission objectives; or

(c) Required to directly prepare resources (data, fuel, power) that are consumed by the above functions.

2. Aeronautic Vehicles include large scale2 aeronautic software systems unique to NASA in which the software:

(a) Is integral to the control of an airborne vehicle;

(b) Monitors and controls the cabin environment; or

(c) Monitors and controls the vehicle's emergency systems.

This definition includes software for vehicles classified as "test," "experimental," or "demonstration" that meets the above definition for Class B software. Also included are systems in a test or demonstration where the software's known and scheduled intended use is to be part of a Class A or B software system.

2 Large-scale (life cycle cost exceeding $250M) fully integrated technology development system – see NPR 7120.8.

b. Examples:

Examples of Class B software include, but are not limited to:

1. Space, Launch, Ground, EDL, and Surface Systems:

Propulsion systems; power systems; guidance navigation and control; fault protection; thermal systems; command and control ground systems; planetary/lunar surface operations; hazard prevention; primary instruments; science sequencing engine; simulations that create operational EDL parameters; subsystems that could cause the loss of science return from multiple instruments; flight dynamics and related data; and launch and flight controller stations for non-human spaceflight.

2. Aeronautics Vehicles (Large Scale NASA Unique):

Guidance, navigation, and control; flight management systems; autopilot; propulsion systems; power systems; emergency systems (e.g., fire suppression systems, emergency egress systems, emergency oxygen supply systems, traffic/ground collision avoidance system); and cabin pressure and temperature control.

c. Exclusions:

Class B does not include:

1. Software that exclusively supports non-primary instruments on non-human space rated systems (e.g., low-cost, non-primary, university supplied instruments); or

2. Systems (e.g., simulators, emulators, facilities) used in testing Class B systems containing software in a development environment; or

3. Software for NASA Class D payloads, as defined in NPR 8705.4.

Class C: Mission Support Software or Aeronautic Vehicles, or Major Engineering/Research Facility Software

a. Definitions:

1. Space Systems include the following types of software:

(a) Flight or ground software necessary for the science return from a single (non-primary) instrument;

(b) Flight or ground software used to analyze or process mission data;

(c) Other software for which a defect could adversely impact the attainment of some secondary mission objectives or cause operational problems;

(d) Software used for the testing of space assets;

(e) Software used to verify system requirements of space assets by analysis; or

(f) Software for space flight operations not covered by Class A or B software.

2. Aeronautic Vehicles include systems for non-large scale aeronautic software systems in which the software:

(a) Is integral to the control of an airborne vehicle;

(b) Monitors and controls the cabin environment; or

(c) Monitors and controls the vehicle’s emergency system.

Also included are systems on an airborne vehicle (including large-scale vehicles) that acquire, store, or transmit the official record copy of flight or test data.

3. Major Engineering/Research Facility is systems that operate a major facility for research, development, test, or evaluation (e.g., facility controls and monitoring, systems that operate facility-owned instruments, apparatus, and data acquisition equipment).

4. Sounding Rockets and Sounding Rocket payloads.

5. Software for NASA Class D payloads, as defined in NPR 8705.4.

b. Examples:

Examples of Class C software include, but are not limited to:

1. Space Systems:

Software that supports prelaunch integration and test; mission data processing and analysis; analysis software used in trend analysis and calibration of flight engineering parameters; primary/major science data collection storage and distribution systems (e.g., Distributed Active Archive Centers); simulators, emulators, or facilities used to test Class A, B, or C software in development; integration and test environments; software used to verify system-level requirements associated with Class A, B, or C software by analysis (e.g., guidance, navigation, and control system performance verification by

analysis); simulators used for mission training; software employed by network operations and control (which is redundant with systems used at tracking complexes); command and control of non-primary instruments; ground mission support software used for secondary mission objectives, real-time analysis, and planning (e.g., monitoring, consumables analysis, mission planning); CubeSat mission software; SmallSat mission software; sounding rocket software and sounding rocket experiments or payload software; and all software on NASA Class D payloads, as defined in NPR 8705.4 to examples of Class C software.

2. Aeronautics Vehicles:

Guidance, navigation, and control; flight management systems; autopilot; propulsion systems; power systems; emergency systems (e.g., fire suppression systems, emergency egress systems, emergency oxygen supply systems, traffic/ground collision avoidance system); cabin pressure and temperature control; in-flight telescope control software; aviation data integration systems; automated flight planning systems and primary/major science data collection storage and distribution systems (e.g., Distributed Active Archive Centers); sounding rockets and sounding rocket experiments or payload software; flight software for free-flying unmanned aerial vehicles (UAVs) in public airspace or over controlled ranges; Balloon Flight software and balloon flight experiment software, and all software on NASA Class D pay loads, as defined in NPR 8705.4.

3. Major Engineering/Research Facility:

Major Center facilities; data acquisition and control systems for wind tunnels, vacuum chambers, and rocket engine test stands; ground-based software used to operate a major facility telescope; and major aeronautic applications facilities (e.g., air traffic management systems; high fidelity motion based simulators).

c. Exclusions:

Class C does not include:

Systems unique to research, development, test, or evaluation activities in a major engineering/research facility or airborne vehicle in which the system is not part of the facility or vehicle and does not impact the operation of the facility or vehicle.

Class D: Basic Science/Engineering Design and Research and Technology Software

a. Definitions:

1. Basic Science/Engineering Design includes:

(a) Ground software that performs secondary science data analysis;

(b) Ground software tools that support engineering development;

(c) Ground software or software tools used for informal testing of software systems;

(d) Ground software tools that support mission planning or formulation;

(e) Ground software that operates a research, development, test, or evaluation laboratory (i.e., not a major engineering/research facility); or

(f) Ground software that provides decision support for non-mission critical situations.

2. Airborne Vehicle Systems include:

(a) Software whose anomalous behavior would cause or contribute to a failure of system function resulting in a minor failure condition for the airborne vehicle (e.g., DO-178C, Software Considerations in Airborne Systems and Equipment Certification, “Class D”);

(b) Software whose anomalous behavior would cause or contribute to a failure of system function with no effect on airborne vehicle operational capability or pilot workload (e.g., DO-178C, “Class E”); or

(c) Ground software tools that perform research associated with airborne vehicles or systems.

3. Major Engineering/Research Facility related software includes research software that executes in a major engineering/research facility but is independent of the operation of the facility.

b. Examples:

Examples of Class D software include, but are not limited to:

1. Basic Science and Engineering Design:

Engineering design and modeling tools (e.g., computer-aided design and computer-aided manufacturing (CAD/CAM), thermal/structural analysis tools); project assurance databases (e.g., problem reporting, analysis, and corrective action system, requirements management databases); propulsion integrated design tools; integrated build management systems; inventory management tools; probabilistic engineering analysis tools; test stand data analysis tools; test stand engineering support tools; experimental flight displays evaluated in a flight simulator; forecasts and assimilated data products; and tools used to develop design reference missions to support early mission planning.

2. Airborne Vehicles:

Software tools for designing advanced human-automation systems; experimental synthetic-vision display; and cloud-aerosol light detection and ranging installed on an aeronautics vehicle; flight software for physically constrained UAVs such as UAVs on tethers, within cages, or used in indoor labs; and experimental UAV payloads with minor consequences of failure.

c. Exclusions:

Class D does not include:

1. Software that can impact primary or secondary mission objectives or cause loss of data that is generated by space systems;

2. Software that operates a major engineering/research facility;

3. Software that operates an airborne vehicle; or

4. Flight software (i.e., software that meets the flight portions of Class A, B, or C Software Classifications).

Class E: Design Concept, Research, Technology, and General Purpose Software

a. Definitions:

1. Software developed to explore a design concept or hypothesis but not used to make decisions for an operational Class A, B, or C system or a to-be-built Class A, B, or C system.

2. Software used to perform minor analyses of science or experimental data.

3. A defect in Class E software may affect the productivity of a single user or small group of users but generally will not affect mission objectives or system safety. Class E software cannot be safety-critical software. If the software is classified as safety-critical software, then it has to be classified as Class D or higher.

4. Class E software runs in a general-purpose computing environment or a board top environment. Class E software does not support ground tests, flight tests, or operations.

b. Examples:

Examples of Class E software include, but are not limited to, parametric models to estimate performance or other attributes of design concepts; software to explore correlations between data sets; line of code counters; file format converters; and document template builders. Class E can include prototypes of flight and ground systems, developed at minimal cost, in the spirit of "exploring a design concept." Once the design concept is demonstrated, and a program agrees to incorporate it for flight or ground operational use, or for an in-flight test of the technology, then the software should be upgraded to its appropriate classification, based on the operational (or in-flight test) use case. Class E software includes, but is not limited to, software such as word processing applications, spreadsheet applications, and presentation applications.

c. Exclusions:

Class E does not include:

1. Flight systems (i.e., software that meets the flight portions of Class A, B, C, or D Software Classifications);

2. Software developed by or for NASA to directly support an operational system (e.g., human-rated space system, robotics spacecraft, space instrument, airborne vehicle, major engineering/research facility, mission support facility, and primary/major science data collection storage and distribution systems);

3. Software developed by or for NASA to be flight qualified to support an operational system;

4. Software that directly affects primary or secondary mission objectives;

5. Software that can adversely affect the integrity of engineering/scientific artifacts;

6. Software used in technical decisions concerning operational systems, or systems being developed for operation;

7. Software that has an impact on operational vehicles; or

8. Software that is safety-critical.

Business and Information Technology Infrastructure Software

Class F: General Purpose Computing, Business, and IT Software

a. Definition:

General purpose computing Business and IT software used in support of the Agency, multiple Centers, multiple programs/projects, single Centers/projects, or locally deployed General Purpose Infrastructure To-Be Component of the NASA Enterprise Architecture. These software applications are generally used to support voice, wide-area network, local area network, video, data centers, cloud computing, information management, business and IT application services (e.g., Finance, Logistics, Human Capital, Procurement), messaging and collaboration, and public Web. A defect in Class F software is likely to affect the productivity of multiple users across a single geographic location or several geographic locations and may affect mission objectives or system safety. Mission objectives can be cost, schedule, or technical objectives for any work that the Agency or a Center performs.

b. Examples:

Examples of Class F software include, but are not limited to, Agency-wide enterprise applications (e.g., WebTADS, SAP, CONCURGov, ePayroll, Business Warehouse), Center-specific software, or specific Web applications, including mobile applications; Agency-wide educational outreach software; software in support of the NASA-wide area network; and the NASA Web portal.

Appendix E. References

a. CMMI® Development V2.0 model, see https://cmmiinstitute.com/.

b. CMU/SEI-2010-TR-033 CMMI® for Development, Version 1.3 Software Engineering Institute, Carnegie Mellon University, 2010. See https://cmmiinstitute.com/.

c. DO-178C, Software Considerations in Airborne Systems and Equipment Certification.

d. FAR 52.212-4, Contract Terms and Conditions - Commercial Items.

e. FAR 2005-014, Federal Acquisition Regulation; Smart BUY Pages 61603 – 61605.

f. Federal Information Security Modernization Act of 2014 (FISMA), 44 U.S.C. § 3551, et seq.

g. Federal Information Technology Acquisition Reform Act (FITARA).

h. Government-Wide, US Government Accountability Office, GAO-14-413.

i. IEEE 828-2012, IEEE Standard for Configuration Management in Systems and Software Engineering.

j. IEEE 1012, IEEE Standard for Software Verification and Validation, https://standards.nasa.gov/.

k. IEEE 1028, IEEE Standard for Software Reviews and Audits, https://standards.nasa.gov/.

l. ISO 5806, Information processing -- Specification of single-hit decision tables.

m. ISO/IEC 2382-20, Information technology–Vocabulary–Part 20: System development, 20.05.05.

n. ISO/IEC 19770-1:2017, Information technology – IT asset management – Part 1: IT asset management systems—Requirements.

o. ISO/IEC 19770-5:2015, Information technology.

p. ISO/IEC 26514:2008, Systems and software engineering–requirements for designers and developers of user documentation.

q. ISO/IEC/IEEE 15026-1:2019, Systems and software engineering—Systems and software assurance—Part 1: Concepts and vocabulary.

r. ISO/IEC/IEEE 24765, Systems and software engineering – Vocabulary.

s. ISO/IEC/IEEE 26513:2017, Systems and software engineering–requirements for testers and reviewers of information for users.

t. ISO/IEC/IEEE 29119-4:2015, Software and systems engineering -- Software testing -- Part 4: Test techniques.

u. ISO/IEC 15939, Systems and Software Engineering—Measurement Process.

v. ISO/IEC 19770, International Standards for Software Asset Management Processes.

w. ISO/IEC 24765, Systems and Software Engineering – Vocabulary.

x. NASA-STD-3001, Volumes I-II, Space Flight Human System Standard.

y. NASA-STD-7009, Standard for Models and Simulations, https://standards.nasa.gov/.

z. NASA-STD-8739.9, Software Formal Inspection Standard, https://standards.nasa.gov/.

aa. NASA FAR Supplement 1852.227-86, Commercial Computer Software License in software agreements.

bb. NASA Guidelines for Use of IT Contracts for Supporting End User Services (Update to MFR#137 and MFR#7).

cc. NASA IV&V Management System, https://www.nasa.gov/centers/ivv/ims/home/index.html.

dd. NASA Software Engineering Website, https://nen.nasa.gov/web/software.

ee. NASA Software Process Across NASA (SPAN) Website, https://nen.nasa.gov/web/software/wiki.

ff. NASA/SP-2010-3403, NASA Scheduling Management Handbook.

gg. NASA-HDBK-4008, Programmable Logic Devices (PLD) Handbook, https://standards.nasa.gov/.

hh. NASA-HDBK-7009, NASA Handbook for Models and Simulations: An Implementation Guide for NASA-STD-7009, https://standards.nasa.gov/.

ii. NFS 1813.301-79, Supporting Federal Policies, Regulations, and NASA Procedural Requirements.

jj. NFS 1852.237-72, Access to Sensitive Information.

kk. NFS 1852.237-73 Release of Sensitive Information.

ll. NIST 800-146, Cloud Computing Synopsis and Recommendations.

mm. NIST 800-37, Risk Management Framework.

nn. NIST 800-53, Security and Privacy Controls for Federal Information Systems and Organizations.

oo. NPD 1200.1, NASA Internal Control.

pp. NPD 2540.1, Personal Use of Government Office Equipment Including Information Technology, https://nodis3.gsfc.nasa.gov/.

qq. NPD 2810.1, NASA Information Security Policy.

rr. NPR 2190.1, NASA Export Control Program, https://nodis3.gsfc.nasa.gov/.

ss. NPR 2210.1, Release of NASA Software, https://nodis3.gsfc.nasa.gov/.

tt. NPR 2830.1, NASA Enterprise Architecture Procedures, https://nodis3.gsfc.nasa.gov/.

uu. NPR 2841.1, Identity, Credential, and Access Management, https://nodis3.gsfc.nasa.gov/.

vv. NPR 7120.7, NASA Information Technology Program and Project Management Requirements.

ww. NPR 7120.10, Technical Standards for NASA Programs and Projects, https://nodis3.gsfc.nasa.gov/.

xx. NPR 7120.11, Health and Medical Technical Authority Implementation.

yy. NPR 7123.1, NASA Systems Engineering Processes and Requirements.

zz. NPR 8000.4, Agency Risk Management Procedural Requirements.

aaa. NPR 8735.2, Management of Government Quality Assurance Functions for NASA Contracts.

bbb. NPR 9250.1, Capital Asset Identification and Treatment, https://nodis3.gsfc.nasa.gov/.

ccc. NTRS ID 20160005787, “Quality Attributes for Mission Flight Software: A Reference for Architects.”

ddd. OMB Memo M-04-08, “Maximizing Use of SmartBuy and Avoiding Duplication Agency Activities with the President’s 24 E-Gov Initiatives.”

eee. OMB Memo M-04-16, “Software Acquisition.”

fff. OMB Memo M-15-14, “Management and Oversight of Federal Information Technology.”

ggg. OMB Memo M-16-12, “Category Management Policy 16-1: Improving the Acquisition and Management of Common Information Technology: Software Licensing.”