NASA Procedural Requirements

COMPLIANCE IS MANDATORY

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

NASA Software Engineering Requirements

Responsible Office: Office of the Chief Engineer

NID 7150-113 NASA Interim Directive for Software License Management

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 Documents
1.3 Document Structure

Chapter 2. Responsibilities

2.1 Roles and Responsibilities
2.2 Principles Related to Tailoring Requirements

Chapter 3. Software Management Requirements

3.1 Software Life cycle Planning
3.2 Software Cost Estimation
3.3 Software Schedules
3.4 Software Project Specific Training
3.5 Software Classification and Planning Assessments
3.6 Software Assurance and Software IV&V
3.7 Safety-critical Software
3.8 Automatic Generation of Software Source Code
3.9 Use of Commercial, Government, Legacy, Heritage, and Modified Off-the-Shelf Software
3.10 Software Verification and Validation
3.11 Software Development Processes
3.12 Software Acquisition
3.13 Software Monitoring
3.14 Software Reuse
3.15 Open Source Software
3.16 Software Security

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 Measurement
5.5 Software Best Practices
5.6 Software Training

Chapter 6. Recommended Software Documentation Contents

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

List of Figures

Figure 1-1 NASA Software Classification Structure
Figure 1-2 Relationships of Governing Software Documents

Preface

P.1 Purpose

Software engineering is a core capability and a key enabling technology for NASA's missions and supporting infrastructure. This directive 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 the NASA Policy Directive (NPD) 7120.4.

P.2 Applicability

a. This directive is applicable to NASA Headquarters and NASA Centers, including Component Facilities and Technical and Service Support Centers. This language applies to the Jet Propulsion Laboratory, a Federally Funded Research and Development Center, other contractors, grant recipients, or parties to 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 directive provides requirements for NASA contracts, grant recipients, or agreements to the responsible NASA project managers and contracting officers 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 directive applies to software development, maintenance, retirement, operations, management, acquisition, and assurance activities. The requirements of this directive cover all software created, acquired, or maintained by or for NASA and apply to all of the Agency's investment areas containing software systems and subsystems. The applicability of these requirements 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 in Appendix D, in conjunction with the Requirements Mapping and Compliance Matrix in Appendix C. Some projects may contain multiple systems and subsystems having different software classes. Using the Requirements Mapping and Compliance Matrix, the applicable requirements and their associated rigor are adapted according to the classification and safety criticality of the software. Figure 1-1 shows the NASA software classification structure.

NASA-Wide Software Classifications

Class A	**Human-Rated Space Software Systems**
Class B	**Non-Human Space-Rated Software Systems or Large-Scale Aeronautics Vehicles**
Class C	**Mission Support Software or Aeronautic Vehicles, or Major Engineering/Research Facility Software**

(e.g., Classes A through C are mostly software developed or acquired for Highly Specialized IT systems)

Class D	**Basic Science/Engineering Design and Research and Technology Software**
Class E	**Design Concept and Research and Technology Software**
Class F	**General Purpose Computing, Business and IT Software (Multi-Center or Multi-Program/Project)**
Class G	**General Purpose Computing, Business and IT Software (Single Center or Project)**
Class H	**General Purpose Desktop Software**

Notes: It is not uncommon for a project to contain multiple systems and subsystems having different software classes.

Figure 1-1 NASA Software Classification Structure

c. This directive is not retroactively applicable to software development, maintenance, operations, management, acquisition, and assurance activities started before September 27, 2004 (i.e., existing systems and subsystems containing software for the International Space Station, Hubble, Chandra, etc.).

d. This directive does not supersede more stringent requirements imposed by individual NASA organizations and other Federal Government agencies.

e. In this directive, 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.

f. In this directive, "software engineering" is defined as the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, i.e., the application of engineering to software.

g. In this directive, "software" is defined as the computer programs, procedures, scripts, rules, and associated documentation and data pertaining to the development and operation of a computer system. This definition applies to software developed by NASA, software developed for NASA, software maintained by or for NASA, commercial off-the-shelf (COTS) software, government off-the-shelf (GOTS) software, modified off-the-shelf (MOTS) software, reused software, auto-generated code, embedded software, the software executed on processors embedded in programmable logic devices (see NASA-HDBK-4008, Programmable Logic Devices (PLD) Handbook), legacy, heritage, and open-source software components.

h. In this directive, 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

a. NPD 1200.1, NASA Internal Control.

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

c. NPD 2091.1, Inventions Made By Government Employees.

d. NPD 7120.6, Knowledge Policy on Programs and Projects.

e. NPR 2190.1, NASA Export Control Program.

f. NPR 2210.1, Release of NASA Software.

g. NPR 2800.1, Managing Information Technology.

h. NPR 2800.2, Electronic and Information Technology Accessibility.

i. NPR 2810.1, Security of Information Technology.

j. NPR 2830.1, NASA Enterprise Architecture Procedures.

k. NPR 2841.1, Identity, Credential, and Access Management.

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

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

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

o. NPR 7120.9, NASA Product Data and Life-Cycle Management (PDLM) for Flight Programs and Projects.

p. NPR 7120.10, Technical Standards for NASA Programs and Projects.

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

r. NPR 8000.4, Agency Risk Management Procedural Requirements.

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

t. NPR 8705.4, Risk Classification for NASA Payloads.

u. NPR 8715.3, NASA General Safety Program Requirements.

v. NPR 8735.1, Procedures for Exchanging Parts, Materials, Software, and Safety Problem Data Utilizing the Government-Industry Data Exchange Program (GIDEP) and NASA Advisories.

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

x. NPR 9250.1, Property, Plant, and Equipment and Operating Materials and Supplies.

y. NASA-STD-8719.13, NASA Software Safety Standard.

z. NASA-STD-8739.8, Software Assurance Standard.

P.5 Measurement/Verification

Compliance with this document is verified by submission to responsible NASA officials of the completed compliance matrix(ces), including any approved waivers and deviations (see Appendix C) and by internal and external controls. Internal controls are consistent with processes 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 7150.2A, NASA Software Engineering Requirements, dated November 19, 2009.

b. NID 7150-1, NASA Interim Directive (NID): NASA Software Engineering Requirements, dated December 16, 2013.

Chapter 1: Introduction

1.1 Overview

1.1.1 This directive imposes requirements on procedures, design considerations, activities, and tasks used to acquire, develop, assure, and maintain software created and acquired by or for NASA programs. This directive is a designed set of requirements for protecting the Agency's investment in software engineering products and to fulfill its responsibility to the citizens of the United States.

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 will find that many of the requirements are satisfied through programs, procedures, and processes that are in place.

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 community together to optimize resources and talents across Center boundaries. For engineers to effectively communicate and work seamlessly among 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; but 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 Requirement; NPR 7120.7, NASA Information Technology and Institutional Infrastructure 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 The NASA Chief Engineer is committed to instituting and updating these requirements to meet the Agency's current and future challenges in software engineering. Successful experiences will be 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.

1.2 Hierarchy of NASA Software-Related Documents

This section helps the reader understand the flow down of requirements with respect to software created and acquired by or for NASA. Figure 1-2 shows the software engineering perspective of the relationship between relevant documents. The shaded documents in the figure show documents that primarily address software engineering policy and requirements. The text that follows the figure provides a brief description of each type of document, listed according to its position in the figure.

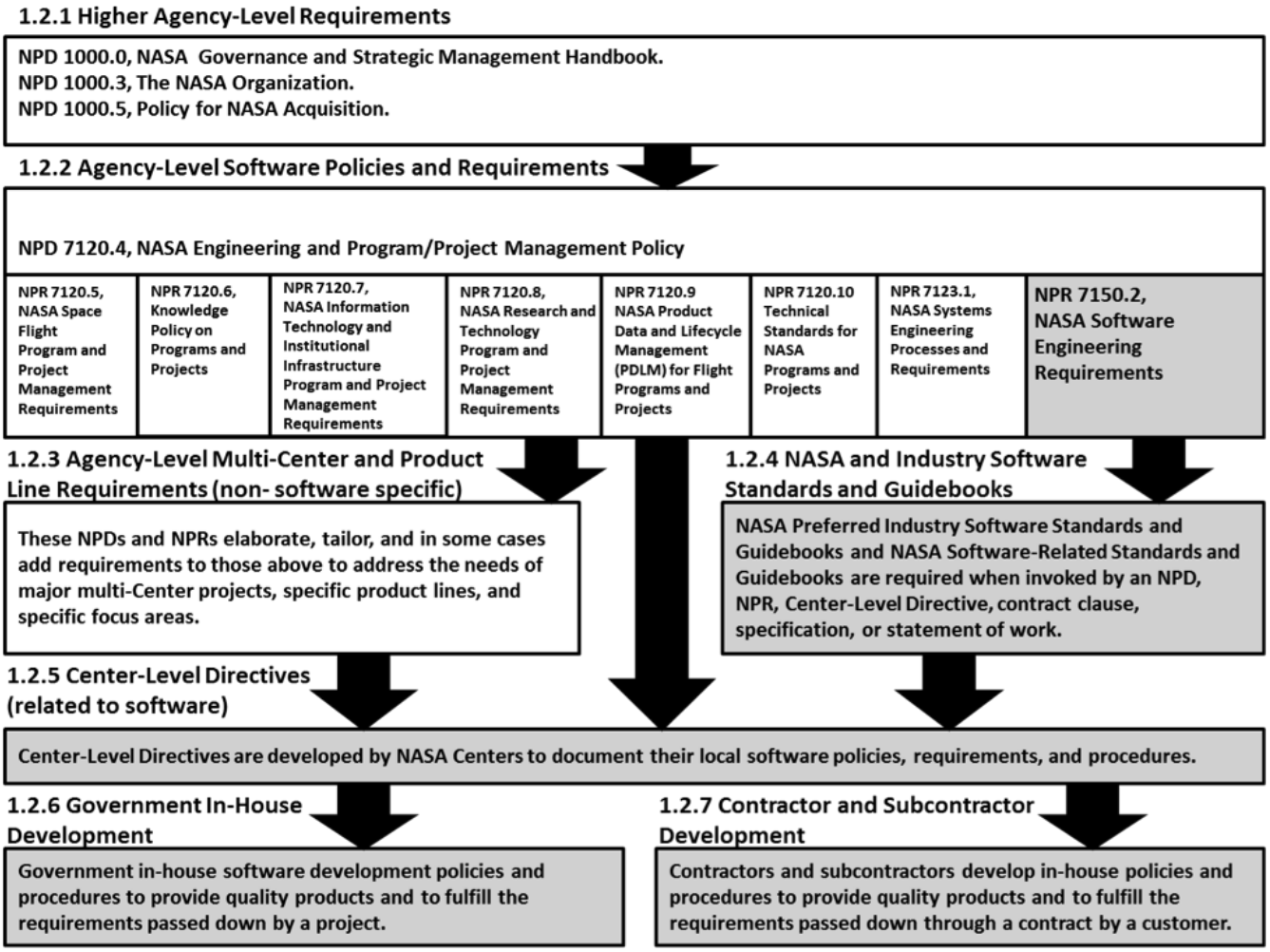

FIGURE 1-2 Relationships of Governing Software Documents

1.2.1 Higher Agency-Level Requirements

NPD 1000.0 is the highest ranking NASA directive. NPD 1000.0 sets forth the principles by which NASA will manage the Agency, describes the means for doing so, and identifies the specific requirements that drive NASA's strategic planning process, leading to products such as the Strategic Plan and the Annual Performance and Accountability Report. NPD 1000.3, The NASA Organization, defines the basic roles and responsibilities necessary to conduct the mission and business of NASA. It is the official repository for defining NASA's organizational architecture. NPD 1000.5 provides the overall policy framework of NASA's disciplined, comprehensive strategic acquisition process with appropriate references to other key processes and directives. This acquisition process complies with NASA obligations as a Federal agency and is tailored to each of NASA's major areas of investment to ensure the efficient, effective use of the resources entrusted to the Agency. In the event of a conflict among the top-level directives, the information provided in the highest ranking directive takes precedence. In the event of conflict among the top-level directives and one or more lower-level NPDs and/or NPRs, the information provided in the top-level directive(s) takes precedence. These policies may include very high-level requirements relevant to software and information technology that are elaborated in lower-level policies and procedural requirements.

1.2.2 Agency-Level Software Policies and Requirements

NPD 7120.4, NASA Engineering and Program/Project Management Policy, is an overarching document that establishes top-level policies for all software created, acquired, and maintained by or for NASA, including COTS, GOTS, and MOTS software and open-source, embedded, reused, legacy, and heritage software. This directive supports the implementation of NPD 7120.4. NPR 7150.2 establishes the set of software engineering requirements established by the Agency for software acquisition, development, maintenance, retirement, operations, and management. It provides a set of software engineering requirements in generic terms to be applied throughout NASA and its contractor community. Software engineering is a core capability and a key enabling technology for NASA's missions and supporting infrastructure. Additional Agency-level project management requirements (NPR 7120.5; NPD 7120.6, Knowledge Policy on Programs and Projects; NPR 7120.7, Information Technology Requirements; NPR 7120.8; NPR 7120.9, NASA Product Data and Life-Cycle Management (PDLM) for Flight Programs and Projects; and NPR 7120.10, Technical Standards for NASA Programs and Projects); and systems

engineering requirements (NPR 7123.1) exist that influence and affect the software development activities on a project. In the event of a conflict between an NPD and an NPR, the information provided in the NPD takes precedence.

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

These 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; and NPR 8735.2, Management of Government Quality Assurance Functions for NASA Contracts.

1.2.4 NASA and Industry Software Standards and Guidebooks

NASA-preferred industry software standards and guidebooks and NASA software-related standards and guidebooks are required when invoked by an NPD, NPR, Center-level directive, contract clause, specification, or statement of work.

1.2.5 Center-Level Directives (related to software)

Center-level directives 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 an NPD or NPR with a Center-level directive, the information provided in the NPD or NPR takes precedence.

1.2.6 Government In-House Development

Government in-house software development policies and procedures are developed to provide quality software products that fulfill the requirements passed down by the project. Government in-house software development policies and procedures are typically designed to meet the needs of the supported projects in an effective and efficient manner.

1.2.7 Contractor and Subcontractor Development

Policies and procedures are developed by contractors and subcontractors to provide quality software products and to fulfill the requirements passed down through a contract by a customer. Contractor and subcontractor policies and procedures are typically designed to satisfy different customers in an effective, efficient manner.

1.3 Document Structure

a. Chapter 2 describes roles and responsibilities relevant to the requirements in this directive.

b. Chapter 3 establishes software management requirements.

c. Chapter 4 provides software engineering life-cycle requirements.

d. Chapter 5 provides supporting software life-cycle requirements.

e. Chapter 6 provides recommended software records content.

f. Appendix A provides definitions.

g. Appendix B provides acronyms used in this directive.

h. Appendix C contains the Requirements Mapping and Compliance Matrix.

i. Appendix D contains software classifications.

j. Appendix E contains software references for this directive.

Chapter 2. Responsibilities

Software engineering is a core capability and a key enabling technology necessary for the support of NASA's Mission Directorates. Ensuring the quality, safety, and reliability of NASA software is of paramount importance in achieving mission success. 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 the software engineering management process. The roles and responsibilities of senior NASA management, along with fundamental principles of governance, are defined in NPD 1000.0 and further described in NPD 1000.3. These requirements are applicable to all NASA Centers. Specific software classification applicability, if any, for the requirements in Chapter 2 is contained in the requirement wording. The majority of requirements in Chapter 2 are not part of the Compliance Matrix in Appendix C. Any tailoring of requirements designated in Chapter 2 can be approved by the appropriate engineering management per the defined roles and responsibilities.

2.1 Roles and Responsibilities

2.1.1 The NASA Chief Engineer (CE)

The NASA CE establishes policy, oversight, and assessment of the NASA engineering and program/project management processes; implements the Engineering Technical Authority process; and serves as principal advisor to the Administrator and other senior officials on matters pertaining to the technical capability and readiness of NASA programs and projects to execute according to plans. The CE directs the NASA Engineering and Safety Center (NESC) and ensures that programs/projects respond to requests from the NESC for data and information needed to make independent technical assessments and then respond to NESC assessments. The CE leads the mission and program/project performance assessment for the Baseline Performance Review (BPR); ensures that space asset protection functional support is provided to NASA missions and management, including at a minimum, preparation of program threat summaries and project protection plans; and co-chairs the Safety and Mission Success Review (SMSR) with the Office of Safety and Mission Assurance (OSMA).

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

2.1.1.2 The NASA CE shall periodically benchmark each Center's software engineering capability against its Center Software Engineering Improvement Plan. [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 Office of the Chief Engineer (OCE) shall periodically review project compliance 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 and fund training to advance software engineering practices and software acquisition. [SWE-100]

2.1.1.6 The NASA OCE shall maintain an Agency-wide process asset library of applicable best practices. [SWE-098]

2.1.2 Chief, Safety and Mission Assurance (SMA)

The Chief, SMA ensures the existence of robust safety and mission assurance processes and activities through the development, implementation, assessment, and functional oversight of Agency-wide safety, reliability, maintainability, quality, and risk management policies and procedures. The Chief, SMA serves as principal advisor to the Administrator and other senior officials on Agency-wide safety, reliability, maintainability, and quality; performs independent program and project compliance verification audits; implements the SMA Technical Authority process; monitors, collects, and assesses Agency-wide safety and mission assurance financial and performance results; oversees the prompt investigation of NASA mishaps and assures the appropriate closure; and co-chairs the SMSR with the OCE.

2.1.2.1 The NASA Chief, SMA will lead, maintain, and fund a NASA Software Assurance Initiative to advance software assurance practices.

2.1.2.2 The NASA Chief, SMA will periodically benchmark each Center's software assurance capabilities against the NASA Software Assurance Standard.

2.1.2.3 The NASA Chief, SMA will periodically review project compliance matrices.

2.1.2.4 The NASA Chief, SMA will authorize appraisals against selected requirements in this NPR to check compliance.

2.1.2.5 The NASA Chief, SMA training organizations will provide and fund software assurance training.

2.1.2.6 The NASA Chief, SMA will make the final decision on all waivers to SWE-141, the independent verification and validation (IV&V) requirement.

2.1.3 Center Directors

2.1.3.1 In this document, 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.

2.1.3.2 Center Directors, or designees, 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.

2.1.3.3 Center Directors, or designees, shall establish, document, execute, and maintain software processes. [SWE-005]

2.1.3.4 Center Directors, or designees, shall comply with the requirements in this directive that are marked with an "X" in Appendix C. [SWE-140]

Note: Project relief from an applicable "X" requirement can be granted only by the designated Technical Authority called out in the column titled "Technical Authority" in Appendix C. The projects also document their related mitigations and risk acceptance in the approved compliance matrix. When the requirement and software class are marked with an "X," the projects record the risk and rationale for any requirements that are completely relieved in the compliance matrix.

2.1.3.5 The designated Center Engineering Technical Authority(s) for requirements in this NPR that can be waived or deviated at the Center level shall be NASA civil servants (or JPL/CalTech employees) approved by the Center Director. [SWE-122]

Note: Center Directors designate an Engineering Technical Authority for software from their engineering organization for software Classes A through E and from their Center CIO organization for Classes G and H. The designation of an Engineering Technical Authority(ies) is documented in the Technical Authority Implementation Plan. The NASA CIO designates the Engineering Technical Authority for Class F software. Refer to Appendix C (column titled "Technical Authority") for requirements and their associated Technical Authority.

2.1.3.6 Serving as Technical Authorities for requirements in this directive, Center Directors, or designees shall:

a. Assess projects' compliance matrices, tailoring, waivers, and deviations from requirements in this directive by: [SWE-126]

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

(2) Evaluating the project's compliance 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 compliance 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) Coordinate with the Center S&MA organization that the compliance matrix implementation approach does not impact safety and mission assurance on the project.

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

(7) Facilitating the processing of projects' tailoring/compliance matrices, tailoring, waivers, or deviations from requirements in this directive, which fall under the responsibilities of a different Technical Authority (see column titled "Technical Authority" in Appendix C).

(8) Ensuring that approved compliance matrices, including any waivers and deviations against this directive, are

archived as part of retrievable project records.

Note: To effectively assess projects' compliance matrices, the designated Center Engineering Technical Authorities for this NPR are recognized NASA software engineering experts or utilize recognized NASA software engineering experts in their decision processes. Additionally, it is a best practice to obtain a risk assessment from the Center's Safety and Mission Assurance organization for any software waivers/deviations prior to Technical Authority approval. 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 their approval by signature(s) in the compliance matrix itself, when the compliance matrix is used to waive/deviate from applicable "X" requirement(s). [SWE-145]

Note: The compliance matrix documents the requirements that the project plans to meet, "not applicable" requirements, and any tailoring approved by designated Technical Authorities with associated justification. If a project wants to waive or deviate from a requirement marked as Headquarters Technical Authority, then the project is required to get NASA Headquarters approval (e.g., NASA Chief Engineer (CE), NASA Chief, Safety and Mission Assurance (CSMA), and/or NASA Chief Health and Medical Officer (CHMO)) on a formal waiver/deviation request or on a software compliance matrix.

2.1.3.7 The Center Director or designee shall periodically report on the status of the Center's software engineering discipline, as applied to its projects, to the NASA Office of Chief Engineer and relevant Technical Authorities as requested. [SWE-095]

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

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. Phase of development or operations.

e. Safety Critical Software (Yes or No).

f. Software Class or list of the software classes being development on the project.

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 KSLOC the CSCI/Major System represents.

(4) The primary programing languages used.

(5) Primary life-cycle methodology being used on the software project.

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

2.1.3.9 For Class A, B, C, and safety critical software projects, the Center Director 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.

2.1.3.10 For Class A, B, C, and safety critical software projects, the Center Director shall utilize software measurement data for monitoring software engineering capability, improving software quality, and tracking the status of software engineering improvement activities. [SWE-092]

2.1.3.11 Each Center Director shall maintain and implement software training plan(s) to advance its in-house software engineering capability and as a reference for its contractors. [SWE-101]

2.1.3.12 For Class A, B, and C software projects, each Center Director shall establish and maintain a software cost repository(ies) that contains at least one of 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.

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

2.1.3.14 The designated Engineering Technical Authority(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 Engineering Technical Authority for software approves the required software content for projects within their scope of authority. Electronic submission of data deliverables is preferred.

2.1.4 Center Safety and Mission Assurance (SMA)

2.1.4.1 The Center SMA ensures the existence of robust safety and mission assurance processes and activities through the development, implementation, assessment, and functional oversight of Center-wide safety, reliability, maintainability, quality, and risk management policies and procedures. The Center SMA serves as principal advisor to the Center Director on Center-wide safety, reliability, maintainability, and quality; performs independent program and project compliance verification audits; implements the SMA Technical Authority process; monitors, collects, and assesses Center-wide safety and mission assurance financial and performance results; and oversees the prompt investigation of Center mishaps and assures the appropriate closure.

2.1.4.2 The Center SMA will ensure that the project's software assurance organization performs an independent classification assessment.

2.1.4.3 The Center SMA will ensure that the project implements software assurance per NASA-STD-8739.8.

2.1.4.4 The Center SMA will ensure that the project determines the software safety criticality in accordance with NASA-STD-8719.13.

2.1.4.5 The Center SMA will ensure that when a project is determined to have safety-critical software, that the project implements the requirements of NASA-STD-8719.13.

2.1.4.6 The Center SMA will approve the project's Independent Verification and Validation (IV&V) provider's IV&V Project Execution Plan (IPEP). 2.1.4.7 The Center SMA will support the project to ensure that acquired, developed, and maintained software, as required by SWE-032, is from an organization with a non-expired CMMI-DEV rating as measured by a CMMI Institute authorized or certified lead appraiser.

2.1.4.8 The Center SMA will support the Center organizations in maintaining the NASA organization's CMMI-DEV ratings.

2.1.5 Program and Project Managers

2.1.5.1 The software management process requires the understanding and application of laws and additional NASA policy requirements that impact the development, release, and/or maintenance of software. The documents listed in this section are additional requirements that may have an effect on software development projects and are mentioned here for awareness and completeness.

2.1.5.2 The Program and Project Managers ensure that software invention requirements of NPD 2091.1 are implemented by the project.

2.1.5.3 The Program and Project Managers ensure that software technology transfer requirements of NPR 2190.1 are implemented by the project. The project ensures that there will be no access by foreign persons or export or transfer to foreign persons or destinations until an export control review is completed and access/release is approved in accordance with NPR 2190.1 and NPR 2210.1.

2.1.5.4 The Program and Project Managers ensure that software external release requirements of NPR 2210.1 are implemented by the project.

2.1.5.5 The Program and Project Managers ensure that the information security requirements of NPR 2810.1 and NPR 2841.1 are implemented by the project.

2.1.5.6 The Program and Project Managers ensure that software is accessible to individuals with disabilities in

accordance with NPR 2800.2.

2.1.5.7 The Program and Project Managers ensure that software acquisitions or developments that meet NASA's capitalization criteria be capitalized per NPR 9250.1.

2.1.5.8 The Program and Project Managers ensure the human-rated software specific requirements of NPR 8705.2 are fulfilled.

2.1.5.9 The Program and Project Managers ensure the implementation of NPR 8735.1 for software in Category 1 and 2 programs and projects (see NPR 7120.5, Space Flight Program and Project Management Requirements and NPR 7120.8, NASA Research and Technology Program and Project Management Requirements) and for payloads with risk classification levels A-D (see NPR 8705.4, Risk Classification for NASA Payloads).

2.1.5.10 The Program and Project Managers ensure that IT strategy, investment, implementation, and operations decisions are integrated per NPR 2800.1.

2.1.5.11 The Program and Project Managers ensure that IT investments made at the project level align with the Agency Enterprise Architecture per NPR 2830.1.

2.1.5.12 The Program and Project Managers ensure compliance with intellectual property requirements and copyright laws.

2.1.5.13 When IV&V is required for a project as per Section 3.6 of this document, the project manager will ensure that IV&V is performed by the NASA IV&V Program, unless an alternate IV&V provider is agreed to by the CSMA.

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 variants from specific requirements are appropriately mitigated, NASA established Technical Authority governance. Waivers and deviations from requirements in this directive are governed by the following requirements, as well as those established in NPD 1000.3, NPR 7120.5, NPR 7120.7, and NPR 7120.8 for all of the Agency's investment areas. The Technical Authority for each requirement in this NPR is documented in the "Technical Authority" column of Appendix C. The NASA CSMA has co-approval on any waiver or deviation decided at the Headquarters level that involves software. The NASA CHMO has co-approval on any waiver or deviation decided at the Headquarters level that involves software with health and medical implications. Waivers or deviations decided at the Center level are to follow similar protocol when software criticality or health and medical issues are involved.

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 and safety criticality. Tailoring is the process used to adjust or seek relief from a prescribed requirement to accommodate the needs of a specific task or activity (e.g., program or project). The tailoring process results in the generation of waivers or deviations depending on the timing of the request (see Appendix A for relevant definitions). 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 document, 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 Where approved, the project manager shall document and reflect the tailored requirement in the plans or procedures controlling the development, acquisition, and/or deployment of the affected software. [SWE-121]

2.2.5 Each project manager with software components shall maintain a compliance matrix or multiple compliance 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 compliance matrices if needed for multiple software components on a given project.

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

Note: Project relief from an applicable "X" requirement can be granted only by the designated Technical

Authority called out in the column titled "Technical Authority" in Appendix C. The projects also document their related mitigations and risk acceptance in the approved compliance matrix. When the requirement and software class are marked with an "X," the projects record the risk and rationale for any requirements that are completely relieved in the compliance matrix.

2.2.7 Requirements in this directive are invoked by Software Classifications 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.

2.2.8 The approval of the Technical 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.

2.2.9 Requests for software requirements relief at either the Center or Headquarters Technical Authority level (i.e., partial or complete relief) may be submitted in the streamlined form of a compliance matrix. The required signatures from the responsible organizations and designated Technical Authorities, engineering and safety and mission assurance, are to be obtained. If the compliance matrix is completed and approved in accordance with NPR 7120.5's direction on Technical Authority and this directive, it meets the requirements for requesting tailoring and serves as a waiver or deviation.

2.2.10 Technical Authorities for requirements in this NPR shall review any tailored requirements whenever changes in project software plans or technical scope are made. [SWE-150]

2.2.11 The tailoring process (which can occur at any time in the program or project's life cycle) results in deviations or waivers to requirements depending on the timing of the request. Deviations and waivers of the requirements in this NPR can be submitted separately to the requirements owner or via the appropriate compliance matrix.

Chapter 3: Software Management Requirements

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

3.1 Software Life Cycle Planning

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

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

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

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

a. Corrective actions are taken, recorded, and managed to closure.

b. Changes to commitments (e.g., software plans) that have been agreed to by the affected groups and individuals.

3.2 Software Cost Estimation

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

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

a. Covers the entire software life cycle.

b. Is based on selected project attributes (e.g., assessment of the size, functionality, complexity, criticality, reuse code, modified code, and risk of the software processes and products).

c. Is based on the cost implications of the technology to be used and the required maturation of that technology.

d. Incorporates risk and uncertainty.

e. Includes the cost for software assurance support.

f. Includes other direct costs.

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

3.3 Software Schedules

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

a. Coordinates with the overall project schedule.

b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest

of the system.

c. Reflects the critical path for the software development activities.

d. Adhere to the guidance provided in NASA/SP-2010-3403, NASA Scheduling Management Handbook.

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

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

3.4 Software Project Specific Training

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

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

3.5 Software Classification and Planning Assessments

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

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

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

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

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

Note: Software Safety Critical Assessment Tool, in NASA-HDBK-2203, can be used to determine the software safety criticality. Engineering and software assurance must reach agreement on safety-critical determination of the software. Disagreements are elevated via both the Engineering Technical Authority and Safety and Mission Assurance Technical Authority chains.

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

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

3.6 Software Assurance and Software IV&V

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

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

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

a. Category 1 projects as defined in NPR 7120.5.

b. Category 2 projects as defined in NPR 7120.5 that have Class A or Class B payload risk classification per NPR

8705.4.

c. Projects specifically selected by the NASA CSMA to have software IV&V.

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

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

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

3.7 Safety-critical Software

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

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

a. Safety-critical software is initialized, at first start and at restarts, to a known safe state.

b. Safety-critical software safely transitions between all predefined known states.

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

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

e. Safety-critical software rejects commands received out of sequence, when execution of those commands out of sequence can cause a hazard.

f. Safety-critical software detects inadvertent memory modification and recovers to a known safe state.

g. Safety-critical software performs integrity checks on inputs and outputs to/from the software system.

h. Safety-critical software performs prerequisite checks prior to the execution of safety-critical software commands.

i. No single software event or action is allowed to initiate an identified hazard.

j. Safety-critical software responds to an off nominal condition within the time needed to prevent a hazardous event.

k. Software provides error handling of safety-critical functions.

l. Safety-critical software has the capability to place the system into a safe state.

m. Safety-critical elements (requirements, design elements, code components, and interfaces) are uniquely identified as safety-critical.

n. Requirements are incorporated in the coding methods, standards, and/or criteria to clearly identify safety-critical code and data within source code comments.

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

3.8 Automatic Generation of Software Source Code

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

a. Validation and verification of auto-generation tools.

b. Configuration management of the auto-generation tools and associated data.

c. Identification of the allowable scope for the use of auto-generated software.

d. Verification and validation of auto-generated source code.

e. Monitoring the actual use of auto-generated source code compared to the planned use.

f. Policies and procedures for making manual changes to auto-generated source code.

g. Configuration management of the input to the auto-generation tool, the output of the auto-generation tool, and modifications made to the output of the auto-generation tools.

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

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

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

a. The requirements to be met by the software component are identified.

b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).

c. Proprietary rights, usage rights, ownership, warranty, licensing rights, and transfer rights have been addressed.

d. Future support for the software product is planned and adequate for project needs.

e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.

f. The project has a plan to perform periodic assessments of vendor reported defects to ensure the defects do not impact the selected software components.

> *Note: The project responsible for procuring off-the-shelf software is responsible for documenting, prior to procurement, a plan for verifying and validating the software to the same level that would be required for a developed software component. The project ensures that the COTS, GOTS, MOTS, reused, and auto generated code software components and data meet the applicable requirements in this directive assigned to its software classification as shown in Appendix C. Open source requirements are in Section 3.15.*

3.10 Software Verification and Validation

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

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

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

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

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

3.11 Software Development Processes

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

a. Measure software development organizations against an industry-wide set of best practices that address software development and maintenance activities applied to products and services.

b. Measure and compare the maturity of an organization's product development and acquisition processes with industry state of the practice.

c. Measure and ensure compliance with the intent of the NPR 7150.2 process related requirements using an industry standard approach.

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

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

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

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

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

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

(1) Requirements Management.

(2) Configuration Management.

(3) Process and Product Quality Assurance.

(4) Measurement and Analysis.

(5) Project Planning.

(6) Project Monitoring and Control.

(7) Supplier Agreement Management (if applicable).

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

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

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

3.12 Software Acquisition

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

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

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

a. Acquire an off-the-shelf software product that satisfies the requirement.

b. Develop the software product or obtain the software service internally.

c. Develop the software product or obtain the software service through contract.

d. Enhance an existing software product or service.

e. Reuse an existing software product or service.

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

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

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

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

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

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

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

a. Monitor product integration.

b. Review the verification activities to ensure adequacy.

c. Review trades studies and source data.

d. Audit the software development process.

e. Participate in software reviews and systems and software technical interchange meetings.

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

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

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

3.13 Software Monitoring

3.13.1 The project manager shall require the software supplier to track software changes and non-conformances

and provide the data for the project's review. [SWE-043]

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

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

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

3.14 Software Reuse

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

a. Increase software productivity.

b. Shorten software development and maintenance time.

c. Reduce duplication of effort.

d. Move personnel, tools, and methods more easily among projects.

e. Reduce software development and maintenance costs.

f. Produce higher quality software products.

g. Increase software and system dependability.

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

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

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

3.15 Open Source

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

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

a. The requirements that are to be met by the software component are identified.

b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions).

c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed.

d. Future support for the software product is planned and adequate for project needs.

e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.

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

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

3.16 Software Security

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

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

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

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

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

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

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

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

Chapter 4: 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. Without recommending a life cycle, the requirements for each of these steps are provided below.

4.1 Software Requirements

4.1.1 The requirements phase is one of the most important 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 important part of the life-cycle process. This can be accomplished via rapid prototyping and customer-involved reviews of iterative and final software requirements.

4.1.2 Requirements Development

4.1.2.1 The project manager shall establish, capture, record, approve, and maintain software requirements, including the software quality requirements, 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.*

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

4.1.2.3 The project manager shall perform, record, and maintain bidirectional traceability between the software requirement and the higher-level requirement. [SWE-052]

4.1.3 Requirements Management

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

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

4.1.3.3 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 largely 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 strategies created by the NASA Space Asset Protection Program to protect mission architectures.

4.2.3 The project manager shall develop and record the software architecture. [SWE-057]

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 to 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 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: 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.

4.3.2 The project manager shall develop, record, and maintain the software design. [SWE-056]

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

4.3.4 The project manager shall perform, record, and maintain bidirectional traceability between the following: [SWE-059]

a. Software requirements and software architecture.

b. Software architecture and software design.

c. Software requirements and software design.

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, adhere to, and verify software coding methods, standards, and/or criteria. [SWE-061]

4.4.4 The project manager shall verify the software code by using the results from static analysis tool(s). [SWE-135]

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

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

4.4.7 The project manager shall perform, record, and maintain bidirectional traceability from software design to the software code. [SWE-064]

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

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, additional software test and integration requirements exist in NPR 8705.2 beyond those listed below.

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 report(s).

4.5.3 The project manager shall perform software testing. [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. Testing could include software integration testing, systems integration testing, validation testing, end-to-end testing, acceptance testing, white and black box testing, decision and path analysis, statistical testing, stress testing, performance testing, regression testing, qualification testing, simulation, and others. The use of automated software testing tools is also to be considered in software testing. Test breadth and accuracy can be increased through the use of test personnel independent of the software design and implementation teams, software peer reviews and inspections of software test procedures and software test results, and employing impartial test witnesses.

4.5.4 The project manager shall verify the requirement to the implementation of each software requirement. [SWE-067]

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

4.5.6 The project manager shall record defects identified during testing and track to closure. [SWE-069]

4.5.7 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 verification and validation techniques and the analysis of models and simulations can be found in NASA-STD-7009 and NASA-HDBK-7009.

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

4.5.9 The project manager shall provide and maintain bidirectional traceability from the software test procedures to the software requirements. [SWE-072]

4.5.10 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.

4.6 Software Operations, Maintenance, and Retirement

4.6.1 Planning for operations, maintenance, and retirement is 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]

Chapter 5: 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, typical configuration management baselines (e.g., requirements, code, and products) happen across the life cycle. Support processes are software management and engineering processes that typically support the entire software life cycle (e.g., configuration management).

5.1 Software Configuration Management (SCM)

5.1.1 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]

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 Standard for Configuration Management in Systems and Software Engineering, IEEE 828-2012, 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 to 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]

5.2 Software Risk Management

5.2.1 Identification and management of risks provide a basis for systematically examining changing situations over time to uncover and correct circumstances that impact the ability of the project to meet its objectives.

5.2.2 The project manager shall identify, analyze, plan, track, control, communicate, and record software risks and mitigation plans in accordance with NPR 8000.4. [SWE-086]

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, 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.

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 a recommended best practice 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.

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 required participants.

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

5.4 Software Measurement

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 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: IEEE Standard Adoption of ISO/IEC 15939 - Systems and Software Engineering - Measurement Process is a good generic model for developing a software measurement process for a project or Center. This international standard contains a set of activities and tasks that comprise a measurement process that meets the specific needs of organizations, enterprises, and projects. 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. This includes collecting software technical measurement data from the project's software supplier(s).

5.4.3 The project manager shall analyze software measurement data collected using documented project-specified and/or 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, Center and Headquarters SMA, and Center repositories. [SWE-094]

Chapter 6: Recommended Software Records Content

6.1 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 content of software records defined in NASA-HDBK-2203. Specific content within these records may not be applicable for every project. Use of NASA Center and contractor formats in document deliverables is acceptable if necessary 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, if safety-critical software.

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.

6.2 The recommendations for 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/or data sets are on the Software Process Across NASA (SPAN) Web site, accessible at https://span.nasa.gov/.

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.

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

Computer. 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 Systems and software engineering-Vocabulary)

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

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).

Deviation. A documented authorization releasing a program or project from meeting a requirement before the requirement is put under configuration control at the level the requirement will be implemented.

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 Systems and software engineering-Vocabulary)

Embedded Software. Software that is part of a larger system and performs some of the requirements of that system. (Source: ISO/IEC 24765 Systems and software engineering- Vocabulary)

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.

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

Government Off-the-Shelf Software. This refers to Government-created software, usually from another project. The software was not created by the current developers (see software reuse). Usually, 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.

Highly Specialized Information Technology. Highly Specialized IT is a part of, internal to, or embedded in a mission platform. The platform's function (e.g., avionics, guidance, navigation, flight controls, simulation, radar, etc.) is enabled by IT but not driven by IT itself (e.g., computer hardware and software to automate internal functions of a spacecraft or spacecraft support system such as spacecraft control and status, sensor signal and data processing, and operational tasking.) Highly Specialized IT acquisitions may include full development (where the information technology is a primary issue) to modification of existing systems (information architecture is firm and demonstrated in an operational environment) where information technology is not an issue. Real time is often critical -- and few opportunities exist to use Commercial Off The Shelf (COTS) or Government Off The Shelf (GOTS) beyond microprocessors and operating systems because these systems are largely unprecedented or largely unique applications. Certain IT considered Mission Critical because the loss of which would cause the stoppage of mission operations supporting real-time on-orbit mission operations is identified as "Highly Specialized" by the Directorate Associate Administrator. Highly Specialized IT is largely custom, as opposed to COTS or commodity IT systems or applications, and includes coding/applications that are integral parts of the research or science requirements, e.g., Shuttle Avionics Upgrade. Common engineering IT tools such as Product Life cycle Management (PLM) systems, Computer-Aided Design (CAD) systems, and collaborative engineering systems and environments are not Highly Specialized IT. Representative examples of Highly Specialized IT include: Avionics software, real-time control systems, onboard processors, Deep Space Network, spacecraft instrumentation software, wind tunnel control system, human physiology monitoring systems, ground support environment, experiment simulators, Mission Control Center, and Launch cameras. (Source: NPR2800.1, Managing Information Technology)

Independent Verification and Validation. Verification and validation performed by an organization that is

technically, managerially, and financially independent of the development organization. (Source: ISO/IEC 24765 systems and software engineering vocabulary)

Information Technology. Any equipment or interconnected system(s) or subsystem(s) of equipment that is used in the automatic acquisition, storage, analysis, evaluation, manipulation, management, movement, control, display, switching, interchange, transmission, or reception of data or information by the Agency (reference FAR 2.101). (Source: NPR2800.1, Managing Information Technology)

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 7123.1B)

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

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)

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

Modified Off-the-Shelf Software. 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/or design of an off-the-shelf software component constitutes "modification," but the common usage allows for some percentage of change before the off-the-shelf software is declared to be modified off-the-shelf (MOTS) software. This may include the changes to the application shell and/or glueware to add or protect against certain features and not to the off-the-shelf software system code directly. See off-the-shelf software.

Off-the-Shelf Software. Software not developed in-house or by a contractor for the specific project now underway. The software is generally developed for a purpose different from the current project. Used in practice as umbrella for COTS, GOTS, and MOTS.

Open-Source Software. Software where its human-readable source code is made broadly available without cost under an OSS license, which provides conditions on 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.

Operational Software. Software that has been accepted and deployed, has been delivered to its customer, or is deployed in its intended environment.

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/or 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.

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.

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 Software. See description in NASA-STD-8719.13.

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 characteristics of a system, entity, phenomena, or process using a computational model. (Source: NASA-STD-7009) Only for the purpose of this document, the term "simulation" refers to only those simulations that are implemented in software.

Software. Computer programs, procedures, scripts, rules, and associated documentation and data pertaining to the development and operation of a computer system. This definition applies to software developed by NASA, software developed for NASA, commercial-off-the-shelf (COTS) software, Government-off-the-shelf (GOTS) software, modified-off-the-shelf (MOTS) software, reused software, auto-generated code, embedded software, the software executed on processors embedded in Programmable Logic Devices (see NASA-HDBK-4008), 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, 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: IEEE 24765, Systems and software engineering-Vocabulary, paragraph 3.2760)

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

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, IEEE Standard for Software Reviews and Audits). 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. Each use may include all or part of the software product and may involve its modification. This term can be applied to any software product (such as requirements and architectures), not just to software code itself. Often, this is software previously written by an in-house development team and used on a different project. GOTS software would come under this category if the product is supplied from one Government project to another Government project.

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

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

Subsystem. A secondary or subordinate system within a larger system. (Source: ISO/IEC 24765, Systems and software engineering-Vocabulary)

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 or seek relief from a prescribed requirement to accommodate the needs of a specific task or activity (e.g., program or project). The tailoring process results in the generation of deviations and waivers depending on the timing of the request.

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, applicability of information, physical

variation, randomness or stochastic behavior, indeterminacy, judgment, and approximation. (Source: NPR 8000.4)

Unit Test. (1) Testing of individual routines and modules by the developer or an independent tester (ISO/IEC/IEEE 24765 Systems and software engineering--Vocabulary) (2) A test of individual programs or modules in order to ensure that there are no analysis or programming errors (ISO/IEC 2382-20 Information technology--Vocabulary--Part 20: System development, 20.05.05) (3) Test of individual hardware or software units or groups of related units. (ISO/IEC/IEEE 24765 Systems and software engineering--Vocabulary)

Waiver. A documented authorization releasing a program or project from meeting a requirement after the requirement is put under configuration control at the level the requirement will be implemented.

Wrapper. See glueware definition.

Appendix B. Acronyms

BPR	Baseline Performance Review
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 Development
CMU	Carnegie Mellon University
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
GOTS	Government-off-the-Shelf
HDBK	Handbook
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
IPEP	IV&V Project Execution Plan
IT	Information Technology
IV&V	Independent Verification and Validation
JPL	Jet Propulsion Laboratory, a Federally Funded Research and Development Center
KDP	Key Decision Point
KLSOC	Kilo/Thousand Source Lines of Code
MOTS	Modified off-the-Shelf
NASA	National Aeronautics and Space Administration
NESC	NASA Engineering and Safety Center
NPD	NASA Policy Directive
NPR	NASA Procedural Requirements
OCE	Office of the Chief Engineer
OSMA	Office of Safety and Mission Assurance

OSS	Open Source Software
PLD	Programmable Logic Devices
SCAMPISM	Standard CMMI® Appraisal Method for Process Improvement
SCM	Software Configuration Management
SEI	Software Engineering Institute
SMA	Safety and Mission Assurance
SMSR	Safety and Mission Success Review
SOW	Statement of Work
SPAN	Software Process Across NASA
SRR	Software Requirements Review
SWE	Software Engineering
WBS	Work Breakdown Structure

Appendix C. Requirements Mapping and Compliance Matrix

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

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

a. The unique requirement identifier.

b. The section reference.

c. The NPR 7150.2 requirement statement.

d. The Technical Authority Level responsible for assessing a project's compliance matrices, tailoring, waivers, and deviations from requirements in this NPR.

e. The requirement owner (the organization or individual responsible for the requirement).

f. 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 Technical Authority approval (ref. Chapter 2.2).

Blank - Optional/Not invoked by this NPR.

X (not OTS) - Does not apply to Off the Shelf (OTS), Commercial Software.

Center Director - Center Director or the Center Director's designated Engineering Technical Authority or Center Director's designated Safety and Mission Assurance Technical Authority.

Note 1 - Project is required to meet this requirement to the extent necessary to satisfy safety critical aspects of the software. All Safety-critical software has to be classified as Class D or Higher.

Note 2 - Applies to Class B software except for Class B software on NASA Class D payloads, as defined in NPR 8705.4. For Class B software, in lieu of a CMMI rating by a development organization, the project will conduct an evaluation, performed by a qualified evaluator selected by the Center Engineering Technical Authority, of the seven process areas listed in SWE-032 and mitigate any risk, if deficient. This exception is intended to be used in those cases in which NASA wishes to purchase a product from the "best of class provider," but the best of class provider does not have the required CMMI rating. When this exception is exercised, the Center Engineering Technical Authority should be notified.

Note 3 - For tailoring of NASA-STD-8739.8 and NASA-STD-8719.13, the Software Assurance Standard and the Software Safety Standard respectively, use the tailoring provided within those documents. They are both risk based and Software Class based tailoring.

Note 4 - The Technical Authority implementation responsibilities for Class F software is at the NASA Headquarters Chief Information Officer (CIO) level, the Technical Authority implementation responsibilities for Class G and H is at the Center CIO organization level or at the level defined in the Center Technical Authority implementation plan. All Safety-critical software has to be classified as Class D or higher.

					Software Class							Software Class		
Section	NPR SWE #	Requirement Text	Technical Authority	Responsibility	A	B	C	D	E	Class C or Class D Software and Safety Critical Software (see Note 1)		F	G	H
3.1.1	13	The project shall develop, maintain, and execute software plans that cover the entire software life cycle and as a minimum address the requirements of this NPR with approved tailoring.	Center Level	Project	X	X	X	X	X			X	X	
3.1.2	24	The project shall track the actual results and performance of software activities against the software plans including the following activities: a. Corrective actions are taken, recorded, and managed to closure when actual results and performance deviate from the software plans. b. Changes to commitments (e.g., software plans) are agreed to by the affected groups and individuals.	Center Level	Project	X	X	X	X				X	X	
3.2.1	15	The project shall establish, document, and maintain two cost estimates and associated cost parameters for all software class A and B projects (that have an estimated project cost of 2 million dollars or more) or one software cost estimate and associated cost parameter(s) for other software projects.	Center Level	Project	X	X	X	X						
3.2.2	151	The project 's software cost estimate or estimates shall satisfy the following conditions: a. Covers the entire software life cycle. b. Is based on selected project attributes (e.g., assessment of the size, functionality, complexity, criticality, and risk of the software processes and products). c. Is based on the cost implications of the technology to be used and the required maturation of that technology. d. Incorporates risk and uncertainty. e. Includes other direct costs.	Center Level	Project	X	X	X	X						
3.3.1	16	The project 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 path for the software development activities.	Center Level	Project	X	X	X	X				X	X	
3.3.2	18	The project shall regularly hold reviews of software activities, status, and results with the project stakeholders and track issues to resolution.	Center Level	Project	X	X	X	X				X	X	
3.3.3	19	The project shall select and document a software development life cycle or model that includes phase transition criteria for each life-cycle phase.	Center Level	Project	X	X	X	X				X (not OTS)	X	
3.4.1	17	The project shall plan, track, and ensure project specific software training for project personnel.	Center Level	Project	X	X	X			X		X	X	

					Software Class							Software Class		
Section	NPR SWE #	Requirement Text	Technical Authority	Responsibility	A	B	C	D	E	Class C or Class D Software and Safety Critical Software (see Note 1)		F	G	H
3.5.1	20	The project shall classify each system and subsystem containing software in accordance with the highest applicable software classification definitions for Classes A, B, C, D, E, F, G, and H software in Appendix D.	Center Level	Project	X	X	X	X	X			X	X	X
3.5.2	132	The project's software assurance organization shall perform an independent classification assessment.	Center Level	Project	X	X	X	X						
3.5.3	133	The project, in conjunction with the Safety and Mission Assurance organization, shall determine the software safety criticality in accordance with NASA-STD-8739.8.	Center Level	Project	X	X	X	X				X	X	X
3.5.4	21	If a system or subsystem evolves to a higher or lower software classification as defined in Appendix D or there is a change in the safety criticality of the software then the project shall update its plan to fulfill the applicable requirements per the Requirements Mapping Matrix in Appendix C and any approved tailoring.	Center Level	Project	X	X	X	X	X			X	X	X
3.6.1	22	The project shall implement software assurance per NASA-STD-8739.8.	Center Level	Project and Center S&MA (Note 3)	X	X	X	X						
3.6.2	141	For Projects started after the effective date of this revision, software Independent Verification and Validation (IV&V) shall be 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 specifically selected by the NASA Chief, Safety and Mission Assurance (S&MA) to have software IV&V.	HQ OSMA	Project and Center S&MA	Per selection criteria defined in the SWE-141 requirement									
3.6.3	131	If software Independent Verification and Validation (IV&V) is performed on a project, the IV&V provider shall develop an IV&V Project Execution Plan (IPEP).	Center Level	Project	X	X	X							
3.7.1	23	When a project is determined to have safety-critical software, the project shall implement the requirements of NASA-STD-8719.13.	Center Level	Project and Center S&MA (Note 3)	X	X				X		X	X	X
3.7.2	134	When a project is determined to have safety-critical software, the project shall implement the following items in the software: a. Safety-critical software is initialized, at first start and at restarts, to a known safe state. b. Safety-critical software safely transitions between all predefined known states. c. Termination performed by software of safety critical functions is performed to a known safe state. d. Operator overrides of safety-critical software functions require at least two independent actions by an operator. e. Safety-critical software rejects commands received out of sequence, when execution of those commands out of sequence can cause a hazard. f. Safety-critical software detects inadvertent memory modification and recovers to a known safe state. g. Safety-critical software performs integrity checks on inputs and outputs to/from the software system. h. Safety-critical software performs prerequisite checks prior to the execution of safety-critical software commands. i. No single software event or action is allowed to initiate an identified hazard.	Center Level	Project	X	X				X				

					Software Class							Software Class		
Section	NPR SWE #	Requirement Text	Technical Authority	Responsibility	A	B	C	D	E	Class C or Class D Software and Safety Critical Software (see Note 1)		F	G	H
3.8.1	146	The project shall define the approach to the automatic generation of software source code including: a. Validation and verification of auto-generation tools. b. Configuration management (CM) of the auto-generation tools and associated data. c. Identification of the allowable scope for the use of auto-generated software. d. Verification and validation of auto-generated source code. e. Monitoring the actual use of auto-generated source code compared to the planned use. f. Policies and procedures for making manual changes to auto-generated source code. g. Configuration management of the input to the auto-generation tool, the output of the auto-generation tool, and modifications made to the output of the auto-generation tools.	Center Level	Project	X	X	X			X		X	X	
3.9.1	27	The project shall satisfy the following conditions when a COTS, GOTS, MOTS, or reused software component is acquired or used: a. The requirements that are to be met by the software component are identified. b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions). c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed. d. Future support for the software product is planned and adequate for project needs. e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.	Center Level	Project	X	X	X			X		X	X	
3.10.1	28	The project shall plan software verification activities, methods, environments, and criteria for the project.	Center Level	Project	X	X	X	X				X	X	
3.10.2	29	The project shall plan the software validation activities, methods, environments, and criteria for the project.	Center Level	Project	X	X	X	X				X	X	
3.10.3	30	The project shall record, address, and track to closure the results of software verification activities.	Center Level	Project	X	X	X	X				X	X	
3.10.4	31	The project shall record, address, and track to closure the results of software validation activities.	Center Level	Project	X	X	X	X				X	X	
3.11.1	32	The project shall acquire, develop, and maintain software from an organization with a non-expired Capability Maturity Model® Integration for Development (CMMI-DEV) rating as measured by a CMMI Institute authorized or certified lead appraiser as follows: For Class A software: CMMI-DEV Maturity Level 3 Rating or higher for software, or CMMI-DEV Capability Level 3 Rating or higher in all CMMI-DEV Maturity Level 2 and 3 process areas for software. For Class B software (except Class B software on NASA Class D payloads, as defined in NPR 8705.4): CMMI-DEV Maturity Level 2 Rating or higher for software, or CMMI-DEV Capability Level 2 Rating or higher for software in the following process areas: a. Requirements Management. b. Configuration Management. c. Process and Product Quality Assurance. d. Measurement and Analysis. e. Project Planning. f. Project Monitoring and Control. g. Supplier Agreement Management (if applicable).	HQ CE	Project	X	X (Note 2)								
3.12.1	33	The project shall assess options for software acquisition versus development.	Center Level	Project	X	X	X	X				X	X	
3.12.2	34	The project shall define and document the acceptance criteria and conditions for the software.	Center Level	Project	X	X	X	X				X	X	
3.12.3	35	The project shall establish a procedure for software supplier selection, including proposal evaluation criteria.	Center Level	Project	X	X	X			X		X		
3.12.4	36	The project shall determine which software processes, software documents, electronic products, software activities, and tasks are required for the project and software suppliers.	Center Level	Project	X	X	X	X				X (not OTS)	X	

					Software Class							Software Class		
Section	NPR SWE #	Requirement Text	Technical Authority	Responsibility	A	B	C	D	E	Class C or Class D Software and Safety Critical Software (see Note 1)		F	G	H
3.12.5	37	The project shall define the milestones at which the software supplier(s) progress will be reviewed and audited as a part of the acquisition activities.	Center Level	Project	X	X	X	X				X (not OTS)	X	
3.12.6	38	The project shall document software acquisition planning decisions.	Center Level	Project	X	X	X	X				X (not OTS)	X	
3.12.7	39	The project shall require the software supplier(s) to provide insight into software development and test activities; at a minimum the following activities are required: a. Monitoring integration. b. Review of the verification adequacy. c. Review of trade study data and results. d. Auditing the software development process. e. Participation in software reviews and systems and software technical interchange meetings.	Center Level	Project	X	X	X	X				X	X	
3.12.8	40	The project shall require the software supplier(s) to provide NASA with software products and software process tracking information, in electronic format, including software development and management metrics.	Center Level	Project	X	X	X			X		X	X	
3.12.9	42	The project shall require the software supplier(s) to provide NASA with electronic access to the source code developed for the project in a modifiable format, including MOTS software and non-flight software (e.g., ground test software, simulations, ground analysis software, ground control software, science data processing software, and hardware manufacturing software).	Center Level	Project	X	X	X			X		X	X	
3.13.1	43	The project shall require the software supplier to track all software changes and non-conformances and provide the data for the project's review.	Center Level	Project	X	X	X			X		X	X	
3.13.2	45	The project shall participate in any joint NASA/contractor audits of the software development process and software configuration management process.	Center Level	Project	X	X	X			X		X	X	
3.13.3	46	The project shall require the software supplier(s) to provide a software schedule for the project's review and schedule updates as requested.	Center Level	Project	X	X	X	X				X	X	
3.13.4	47	The project shall require the software supplier(s) to make available, electronically, the software traceability data for the project's review.	Center Level	Project	X	X	X			X		X	X	
3.14.1	147	The project shall specify reusability requirements that apply to its software development activities to enable future reuse of the software (including models used to generate the software).	Center Level	Project	X	X	X					X	X	
3.14.2	148	The project shall evaluate software for potential reuse by other projects across the Agency and contribute reuse candidates to the Agency Software Catalog.	Center Level	Project	X	X	X	X				X	X	
3.15.1	149	The project shall ensure that when an open source software component is acquired or used, the following conditions are satisfied: a. The requirements that are to be met by the software component are identified. b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions). c. Proprietary, usage, ownership, warranty, licensing rights, and transfer rights have been addressed. d. Future support for the software product is planned and adequate for project needs. e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use.	Center Level	Project	X	X	X	X				X	X	
3.15.2	41	The project shall require the software supplier(s) to notify the project, in the response to the solicitation, as to whether open source software will be included in code developed for the project.	Center Level	Project	X	X	X			X		X	X	
4.1.1.1	50	The project shall establish, capture, record, approve, and maintain software requirements, including the software quality requirements, as part of the technical specification.	Center Level	Project	X	X	X	X	X			X	X	X
4.1.1.2	51	The project shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements and the hardware specifications and design.	Center Level	Project	X	X	X			X		X	X	
4.1.1.3	52	The project shall perform, record, and maintain bidirectional traceability between the software requirement and the higher-level requirement.	Center Level	Project	X	X	X			X		X (not OTS)	X	

					Software Class							Software Class		
Section	NPR SWE #	Requirement Text	Technical Authority	Responsibility	A	B	C	D	E	Class C or Class D Software and Safety Critical Software (see Note 1)		F	G	H
4.1.2.1	53	The project shall collect and manage changes to the software requirements.	Center Level	Project	X	X	X	X				X	X	
4.1.2.2	54	The project shall identify, initiate corrective actions, and track until closure inconsistencies among requirements, project plans, and software products.	Center Level	Project	X	X	X			X		X	X	
4.1.2.3	55	The project shall perform requirements validation to ensure that the software will perform as intended in the customer environment.	Center Level	Project	X	X	X			X		X	X	
4.2.1	57	The project shall develop and record the software architecture.	Center Level	Project	X	X	X			X		X (not OTS)	X(not OTS)	
4.2.2	143	The project 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 Level	Project	Per selection criteria defined in the SWE-143 requirement									
4.3.1	56	The project shall develop, record, and maintain the software design.	Center Level	Project	X	X	X	X		X		X (not OTS)	X(not OTS)	
4.3.2	58	The project shall develop, record, and maintain a design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested.	Center Level	Project	X	X	X			X		X (not OTS)	X(not OTS)	
4.3.3	59	The project shall perform, record, and maintain bidirectional traceability between the following: a. software requirements and software architecture. b. software architecture and software design. c. software requirements and software design.	Center Level	Project	X	X	X			X		X (not OTS)	X(not OTS)	
4.4.1	60	The project shall implement the software design into software code.	Center Level	Project	X	X	X	X				X (not OTS)	X(not OTS)	
4.4.2	61	The project shall select, adhere to, and verify software coding methods, standards, and/or criteria.	Center Level	Project	X	X	X			X		X (not OTS)	X(not OTS)	
4.4.3	135	The project shall verify the software code by using the results from static analysis tool(s).	Center Level	Project	X	X	X			X				
4.4.4	62	The project shall unit test the software code per the plans for software testing.	Center Level	Project	X	X	X	X				X (not OTS)	X(not OTS)	
4.4.5	63	The project shall provide a software version description for each software release.	Center Level	Project	X	X	X	X				X (not OTS)	X(not OTS)	
4.4.6	64	The project shall perform, record, and maintain bidirectional traceability from software design to the software code.	Center Level	Project	X	X	X			X		X (not OTS)	X(not OTS)	
4.4.7	136	The project shall validate and accredit software tool(s) required to develop or maintain software.	Center Level	Project	X	X	X			X		X	X	
4.5.1	65	The project shall establish and maintain: a. Software test plan(s). b. Software test procedure(s). c. Software test report(s).	Center Level	Project	X	X	X	X				X	X	
4.5.2	66	The project shall perform software testing.	Center Level	Project	X	X	X	X				X	X	
4.5.3	67	The project shall verify the requirement to the implementation of each software requirement.	Center Level	Project	X	X	X			X		X	X	
4.5.4	68	The project shall evaluate test results and record the evaluation.	Center Level	Project	X	X	X	X				X	X	
4.5.5	69	The project shall record defects identified during testing and track to closure.	Center Level	Project	X	X	X	X				X	X	
4.5.6	70	The project shall verify, validate, and accredit software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment.	Center Level	Project	X	X	X	X				X		
4.5.7	71	The project shall update software test plan(s) and software test procedure(s) to be consistent with software requirements.	Center Level	Project	X	X	X			X		X	X	
4.5.8	72	The project shall provide and maintain traceability from the software test procedures to the software requirements.	Center Level	Project	X	X	X	X				X	X	
4.5.9	73	The project shall validate the software system on the targeted platform or high-fidelity simulation.	Center Level	Project	X	X	X			X		X		
4.6.1	75	The project shall plan and implement software operations, maintenance, and retirement activities.	Center Level	Project	X	X	X	X				X	X	
4.6.2	77	The project 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 Level	Project	X	X	X	X	X			X	X	

					Software Class							Software Class		
Section	NPR SWE #	Requirement Text	Technical Authority	Responsibility	A	B	C	D	E	Class C or Class D Software and Safety Critical Software (see Note 1)		F	G	H
5.1.1	79	The project 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 Level	Project	X	X	X			X		X	X	
5.1.2	80	The project shall track and evaluate changes to software products.	Center Level	Project	X	X	X	X				X	X	
5.1.3	81	The project 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 Level	Project	X	X	X	X				X	X	X
5.1.4	82	The project shall establish and implement procedures to: a. Designate the levels of control each identified configuration item must pass through. b. Identify the persons or groups with authority to authorize changes. c. Identify the persons or groups to make changes at each level.	Center Level	Project	X	X	X			X		X	X	X
5.1.5	83	The project shall prepare and maintain records of the configuration status of configuration items.	Center Level	Project	X	X	X					X	X	X
5.1.6	84	The project shall perform software configuration audits to determine the correct version of the configuration items and verify that they conform to the records that define them.	Center Level	Project	X	X	X			X		X	X	
5.1.7	85	The project shall establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products.	Center Level	Project	X	X	X	X				X	X	X
5.2.1	86	The project shall identify, analyze, plan, track, control, communicate, and record software risks in accordance with NPR 8000.4.	Center Level	Project	X	X	X			X		X	X	
5.3.1	87	The project shall perform and report the results of software peer reviews or software inspections for: a. Software requirements. b. Software plans. 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.	Center Level	Project	X	X	X			X		X (not OTS)	X	
5.3.2	88	The project 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 required participants.	Center Level	Project	X	X	X			X		X (not OTS)	X	
5.3.3	89	The project shall, for each planned software peer review or software inspection, record basic measurements.	Center Level	Project	X	X				X		X (not OTS)	X	
5.4.1	90	The project shall establish, record, maintain, report, and utilize software management and technical measurements.	Center Level	Project	X	X	X			X		X (not OTS)	X	
5.4.2	93	The project shall analyze software measurement data collected using documented project-specified and/or Center/organizational analysis procedures.	Center Level	Project	X	X	X			X		X (not OTS)	X	
5.4.3	94	The project shall provide access to the software measurement data, measurement analyses and software development status as requested to the sponsoring Mission Directorate, the NASA Office of Chief Engineer and Center repositories.	Center Level	Project	X	X	X			X		X (not OTS)	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. Definitions for software classes are defined below, and the designation of the software as safety critical or non-safety critical in conjunction with the Requirements Mapping and Compliance Matrix in Appendix C. These definitions are based on (1) usage of the software with or within a NASA system, (2) criticality of the system to NASA's major programs and projects, (3) extent to which humans depend upon the system, (4) developmental and operational complexity, and (5) extent of the Agency's investment. Classes A through E cover engineering-related software in decreasing order of this directive's applicable requirements. Classes F through H cover business and IT software in decreasing order of applicable NPR 7120.7 requirements. Using the Requirements Mapping and Compliance Matrix, the number of applicable requirements and their associated rigor are scaled back for lower software classes and software designated as non-safety critical. 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 Safety and Mission Assurance provide dual Technical Authority chains for resolving classification issues. The NASA Chief Engineer is the ultimate Technical Authority for software classification disputes concerning definitions in this NPR.

D.3 Software classification tool details are defined in NASA-HDBK-2203.

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

â??

Class A: Human Rated Space Software Systems

a. Definition:

1. Human Space Flight Software Systems*: Ground and flight software systems developed and/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, or

(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, and entry, descent, and landing.

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:

Extra vehicular 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.

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-STD-3001, Vol. 2 - NASA Space Flight Human System Standard: Human Factors, Habitability, and Environmental Health.

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 manned 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);

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

3. Systems (e.g., simulators, emulators, stimulators, 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. Airborne Vehicles include large scale1 aeronautic vehicles 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.

1 Large-scale (life-cycle cost exceeding $250M) fully integrated technology development system â?" see NPR 7120.8, section 5.1.1.1.

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 space flight.

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, stimulators, facilities) used in testing Class B systems containing software in a development environment.

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

a. Definition:

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 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 that are not covered by Class A or B software.

2. Airborne Vehicles include systems for non-large scale aeronautic vehicles 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).

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, stimulators, or facilities used to test Class A, B, or C software in a development; integration and test environments (development environment, including environments used from unit testing through validation testing); 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; and ground mission support software used for secondary mission objectives, real-time analysis, and planning (e.g., monitoring, consumables analysis, mission planning).

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; and automated flight planning systems and primary/major science data collection storage and distribution systems (e.g., Distributed Active Archive Centers).

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:

Systems unique to a research, development, test, or evaluation activity 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 used in testing other Class D 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., the Software Considerations in Airborne System and Equipment Certification, DO-178B, "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., the Software Considerations in Airborne System and Equipment Certification, DO-178B, "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; 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.

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

Class E: Design Concept and Research and Technology Software

a. Definition:

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 to-be-built Class A, B, or C system, or

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

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.

c. Exclusions:

Class E does not include:

1. Space flight systems (i.e., software that meets the space flight portions of Class A, B, or C 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,

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 (Multi-Center or Multi- Program and Project)

a. Definition:

General purpose computing Business and IT software used in support of the Agency, multiple Centers, or multiple programs and projects, as described for the General Purpose Infrastructure To-Be Component of the NASA Enterprise Architecture, Volume 5 (To-Be Architecture), and for the following portfolios: voice, wide-area network, local-area network, video, data Centers, application services, messaging and collaboration, and public Web. A defect in Class F software is likely to affect the productivity of multiple users across several geographic locations and may possibly affect mission objectives or system safety. Mission objectives can be cost, schedule, or technical objectives for any work that the Agency performs.

b. Examples:

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

Class G: General Purpose Computing, Business and IT Software (Single Center or Project)

a. Definition:

General purpose computing, business and IT software used in support of a single Center or project, as described for locally deployed portions of the General Purpose Infrastructure To-Be Component of the NASA Enterprise Architecture, Volume 5 (To-Be Architecture) and for the following portfolios: voice, local-area network, video, data Centers, application services, messaging and collaboration, and public Web. A defect in Class G software is likely to affect the productivity of multiple users in a single geographic location or workgroup but is unlikely to affect mission objectives or system safety.

b. Examples:

Examples of Class G software include, but are not limited to software for Center custom applications such as

Headquarters' Corrective Action Tracking System; Headquarters' User Request Systems; content management system mobile applications; and Center or project educational outreach software.

Class H: General Purpose Desktop Software

a. Definition:

General purpose desktop software as described for the General Purpose Infrastructure To-Be Component (Desktop Hardware and Software Portfolio) of the NASA Enterprise Architecture, Volume 5 (NASA To-Be Architecture). A defect in Class H software may affect the productivity of a single user or small group of users but generally will not affect mission objectives or system safety, but a defect in desktop IT security-related software, e.g., anti-virus software, may lead to loss of functionality and productivity across multiple users and systems.

b. Examples:

Examples of Class H software include, but are not limited to, desktop applications such as word processing applications, spreadsheet applications, and presentation applications.

Appendix E. References

E.1 NASA-STD-3000	Vol. 2 - NASA Space PACE Flight Human System Standard: Human Factors, Habitability, and Environmental Health
E.2 NASA-STD-7009	Standard for Models and Simulations
E.3 NASA-STD-8739.9	Software Formal Inspection Standard
E.4 NASA-HDBK-2203	
NASA Software Engineering Handbook	
E.5 NASA-HDBK-4008	Programmable Logic Devices (PLD) Handbook
E.6 NASA-HDBK-7009	NASA Handbook for Models and Simulations: An Implementation Guide for NASA-STD-7009
E.7 NASA Software Engineering Website	https://nen.nasa.gov/web/software/
E.8 NASA Software Process Across NASA (SPAN) Website	http://span.nasa.gov/
E.9 NASA IV&V Management System	http://www.nasa.gov/centers/ivv/ims/home/index.html
E.10 NASA/SP-2010-3403	NASA Scheduling Management Handbook
E.11 IEEE 828	IEEE Standard for Configuration Management in Systems and Software Engineering
E.12 IEEE 1012	IEEE Standard for Software Verification and Validation
E.13 IEEE 1028	IEEE Standard for Software Reviews and Audits
E.14 ISO/IEC 15939	Systems and Software Engineering-Measurement Process
E.15 ISO/IEC 24765	Systems and Software Engineering-Vocabulary
E.16 CMU/SEI-2010-TR-033	CMMI for Development, Version 1.3
Software Engineering Institute, Carnegie Mellon University, 2010	