Approved: 2020-06-10

Measurement System Identification:
Not Measurement Sensitive

NASA TECHNICAL STANDARD

National Aeronautics and Space Administration

NASA-STD-8739.8A

Approved: 2020-06-10
Superseding NASA-STD-8739.8
With Change 1

SOFTWARE ASSURANCE AND SOFTWARE SAFETY STANDARD

APPROVED FOR PUBLIC RELEASE – DISTRIBUTION IS UNLIMITED

DOCUMENT HISTORY LOG

Status	Document Revision	Approval Date	Description
Baseline	Initial	2004-07-28	Initial Release
	1	2005-05-05	Administrative changes to the Preface; Paragraphs 1.1, 1.4, 1.5, 2.1.1, 2.2.2, 3, 5.1.2.3, 5.4.1.1; 5.6.2, 5.8.1.2, 6.7.1.a, 7.3.2, 7.3.3, 7.5, 7.5.1; Table 1; Appendix A; Appendix C to reflect NASA Transformation changes, reflect the release of NASA Procedural Requirements (NPR) 7150.2, NASA Software Engineering Requirements and to make minor editorial changes. Note: Some paragraphs have changed pages as a result of these changes. Only pages where content has changed are identified by change indications.
	A	2020-06-10	The revised document addresses the following significant issues: combined the NASA Software Assurance Standard (NASA-STD-8739.8) with the NASA Software Safety Standard (NASA-STD-8719.13), reduction of requirements, bring into alignment with updates to NPR 7150.2, added a section on IV&V requirements to perform IV&V, and moved guidance text to an Electronic Handbook. This change combines the updates to NASA-STD-8739.8 and the content of NASA-STD-8719.13. The update includes the NASA software safety requirements and cancels NASA-STD-8719.13 standard.

FOREWORD

This NASA Technical Standard is published by the National Aeronautics and Space Administration (NASA) to provide uniform engineering and technical requirements for processes, procedures, practices, and methods that have been endorsed as standard for NASA facilities, programs, and projects, including requirements for selection, application, and design criteria of an item.

This standard was developed by the NASA Office of Safety and Mission Assurance (OSMA). Requests for information, corrections, or additions to this standard should be submitted to the OSMA by email to Agency-SMA-Policy-Feedback@mail.nasa.gov or via the "Email Feedback" link at https://standards.nasa.gov.

F. Groen for T. Wilcutt
Terrence W. Wilcutt
NASA Chief, Safety and Mission Assurance

June 10, 2020
Approval Date

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF TABLES

SOFTWARE ASSURANCE AND SOFTWARE SAFETY STANDARD

1. SCOPE

1.1 Document Purpose

1.1.1 The purpose of the Software Assurance and Software Safety Standard is to define the requirements to implement a systematic approach to Software Assurance (SA), software safety, and Independent Verification and Validation (IV&V) for software created, acquired, provided, or maintained by or for NASA. Various personnel in the program, project, or facility, and Safety and Mission Assurance (SMA) organizations can perform the activities required to satisfy these requirements. The Software Assurance and Software Safety Standard provides a basis for personnel to perform software assurance, software safety, and IV&V activities consistently throughout the life of the software, that is, from its conception, through creation to operations and maintenance, and until the software is retired.

1.1.2 The Software Assurance and Software Safety Standard, in accordance with NPR 7150.2, NASA Software Engineering Requirements, supports the implementation of the software assurance, software safety, and IV&V sub-disciplines. The application and approach to meeting the Software Assurance and Software Safety Standard will vary based on the system and software products and processes to which they are applied. The Software Assurance and Software Safety Standard stresses coordination between the software assurance sub-disciplines, as well as with system safety, system reliability, hardware quality, system security, and software engineering, to maintain the system perspective and minimize duplication of effort.

1.1.3 The objectives of the Software Assurance and Software Safety Standard include:

a. Ensuring that the processes, procedures, and products used to produce and sustain the software conform to all requirements and standards specified to govern those processes, procedures, and products.

b. Ensuring that the software systems are safe and that the software safety-critical requirements and processes are followed.

c. Ensuring that the software systems are secure.

1.1.4 The Software Assurance and Software Safety Standard is compatible with all software life-cycle models. The Software Assurance and Software Safety Standard does not impose a particular life-cycle model on each software project; it does support a standard set of life-cycle phases as defined in NPR 7150.2.

1.1.5 In this standard, all mandatory actions (i.e., requirements) are denoted by statements containing the term “shall.” The terms “may” denotes a discretionary privilege or permission, “can” denotes statements of possibility or capability, “should” denotes a good practice and is recommended, but not required, “will” denotes expected outcome, and “are/is” denotes descriptive material.

1.2 Applicability

1.2.1 This standard is approved for use by NASA Headquarters and NASA Centers, including Component Facilities and Technical and Service Support Centers. This NASA Technical Standard applies to the assurance of software created by or for NASA projects, programs, facilities, and activities and defines the requirements for those activities. This standard may also apply to the Jet Propulsion Laboratory or other contractors, grant recipients, or parties to agreements to the extent specified or referenced in their contracts, grants, or agreements.

1.3 Documentation and Deliverables

1.3.1 The Software Assurance and Software Safety Standard is not intended to designate the format of program/project/facility documentation and deliverables. The software assurance and software safety information and plans are quality records. The format of the documentation is a program/project/facility decision. The software assurance and software safety organizations will keep records, reports, and metrics, as well as analyses and trending results and should, keep copies of their project plans for future reference and improvements. The software assurance and software safety plans (e.g., the SA plan) can be standalone documents or incorporated within other documents (e.g., part of a software management/development plan, or part of a system Safety and Mission Assurance (SMA) plan).

1.3.2 The software assurance and software safety organization and the project manager are responsible for developing the software assurance and software safety plan(s). The software assurance organization contributes to the development of the Software Management Plan(s) and Software Development Plan(s). The SMA organization has signature authority on software plans and documentation.

1.4 Request for Relief

1.4.1 Tailoring of this standard for application to a specific program or project will be documented as part of program or project requirements and approved by the responsible Center Technical Authority (TA) in accordance with NPR 8715.3, NASA General Safety Program Requirements. Section 4.5 of the standard contains the principles related to tailoring the standard requirements.

2. APPLICABLE AND REFERENCE DOCUMENTS

2.1 Applicable Documents

The applicable documents are accessible via the NASA Technical Standards System at https://standards.nasa.gov or may be obtained directly from the Standards Developing Organizations or other document distributors.https://standards.nasa.gov or may be obtained directly from the Standards Developing Organizations or other document distributors.

2.1.1 Government Documents

NPR 1400.1	NASA Directives and Charters Procedural Requirements

NPR 7120.5	NASA Space Flight Program and Project Management Requirements
NPR 7120.10	Technical Standards for NASA Programs and Projects
NPR 7150.2	NASA Software Engineering Requirements
NPR 8000.4	Agency Risk Management Procedural Requirements
NPR 8715.3	NASA General Safety Program Requirements
NASA-HDBK-2203	NASA Software Engineering Handbook

2.2 Reference Documents

The reference documents listed in this section are not incorporated by reference within this standard but may provide further clarification and guidance.

2.2.1 Government Documents

NPD 2810.1	NASA Information Security Policy
NPD 8720.1	NASA Reliability and Maintainability Program Policy
NPR 1441.1	NASA Records Management Program Requirements
NPR 2810.1	Security of Information Technology
NPR 7123.1	NASA Systems Engineering Processes and Requirements
NASA-STD-7009	Standard for Models and Simulations
NASA-HDBK-8709.22	Safety and Mission Assurance Acronyms, Abbreviations, and Definitions
NASA-HDBK-8739.23	NASA Complex Electronics Handbook for Assurance Professionals
NASA-GB-8719.13	NASA Software Safety Guidebook
NIST SP 800-40	Creating a Patch and Vulnerability Management Program
NIST SP 800-53	Security Controls and Assessment Procedures for Federal Information Systems and Organizations
NIST SP 800-70	National Checklist Program for Information Technology products: Guidelines for Checklist Users and Developers

NIST SP 800-115	Technical Guide to Information Security Testing and Assessment

2.2.2 Non-Government Documents

CMMI®-DEV, V1.3	Capability Maturity Model Integration (CMMI®) for Development, Version 1.3
CMMI®-DEV, V2.0	CMMI® Development, Version 2.0
IEEE 730-2014	Institute of Electrical and Electronics Engineers (IEEE) Standard for Software Assurance Processes
IEEE 982.1-2005	IEEE Standard Measures of the Software Aspects of Dependability, 8 November 2005
IEEE 1012-2017	IEEE Standard for System, Software, and Hardware Verification and Validation, 28 September 2017
IEEE 1028-2008	IEEE Standard for Software Reviews and Audits, 15 August 2008
IEEE 1633-2016	IEEE Recommended Practice on Software Reliability, 22 September 2016
ISO 24765-2017	System and Software Engineering – Vocabulary, Second Edition,

2.3 Order of Precedence

2.3.1 This standard establishes requirements to implement a systematic approach to SA, Software Safety, and IV&V for software created, acquired, provided, or maintained by or for NASA but does not supersede nor waive established Agency requirements found in other documentation.

2.3.2 Conflicts between the Software Assurance and Software Safety Standard and other requirements documents will be resolved by the responsible SMA and engineering TA(ies), per NPR 1400.1, NASA Directives and Charters Procedural Requirements, and NPR 7120.10, Technical Standards for NASA Programs and Projects.

3. ACRONYMS AND DEFINITIONS

3.1 Acronyms and Abbreviations

CMMI®	Capability Maturity Model Integration
COTS	Commercial Off-The-Shelf
GOTS	Government Off-The-Shelf
IEEE	Institute of Electrical and Electronics Engineers
IPEP	IV&V Program Execution Plan
IV&V	Independent Verification and Validation
MOTS	Modified Off-The-Shelf
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
NPD	NASA Policy Directive
NPR	NASA Procedural Requirements
OSMA	HQ Office, Safety and Mission Assurance
OSS	Open Source Software
SASS	Software Assurance and Software Safety
SWE	Software Engineering
TA	Technical Authority

3.2 Definitions

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

Acquirer: The entity or individual who specifies the requirements and accepts the resulting software products. The acquirer is usually NASA or an organization within the Agency but can also refer to the prime contractor-subcontractor relationship as well.

Analyze: Review results in-depth, look at relationships of activities, examine methodologies in detail, follow methodologies such as Failure Mode and Effects Analysis, Fault Tree Analysis, trending, and analysis of metrics. Examine processes, plans, products, and task lists for completeness, consistency, accuracy, reasonableness, and compliance with requirements. The

analysis may include identifying missing, incomplete, or inaccurate products, relationships, deliverables, activities, required actions, etc.

Approve: When the responsible originating official, or designated decision authority, of a document, report, condition, etc. has agreed, via their signature, to the content and indicates the document is ready for release, baselining, distribution, etc. Usually, there will be one "approver" and several stakeholders who would need to "concur" for official acceptance of a document, report, etc. (for example, the project manager would approve the Software Development Plan, but SMA would concur on it.)

Assess: Judge results against plans or work product requirements. Assess includes judging for practicality, timeliness, correctness, completeness, compliance, evaluation of rationale, etc., reviewing activities performed, and independently tracking corrective actions to closure.

Assure: When software assurance personnel make certain that others have performed the specified software assurance, management, and engineering activities.

Audit: (1) systematic, independent and documented process for obtaining audit evidence and evaluating it objectively to determine the extent to which audit criteria are fulfilled (2) independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria (3) independent examination of a software product, software process, or set of software processes to assess compliance with specifications, standards, contractual agreements, or other criteria (4) systematic, independent, documented process for obtaining records, statements of fact, or other relevant information and assessing them objectively, to determine the extent to which specified requirements are fulfilled. *Note:* An audit can be an internal audit (first-party) or an external audit (second party or a third party), and it can be a combined or integrated audit (combining two or more disciplines). Audit results are a clear indication of whether the audit criteria have been met. (IEEE Definition)

Concur: A documented agreement that a proposed course of action is acceptable.

Condition: (1) measurable qualitative or quantitative attribute that is stipulated for a requirement and that indicates a circumstance or event under which a requirement applies (2) description of a contingency to be considered in the representation of a problem, or a reference to other procedures to be considered as part of the condition (3) true or false logical predicate (4) logical predicate involving one or more behavior model elements (5) Boolean expression containing no Boolean operators.

Configuration Item: An aggregation of hardware, software, or both, that is established and baselined, with any modifications tracked and managed. Examples include requirements document, data block, Use Case, or unit of code.

Confirm: Checks to see that activities specified in the software engineering requirements are adequately done, and evidence of the activities exists as proof. Confirm includes making sure activities are done completely and correctly and have expected content in accordance with approved tailoring.

Deliverable: Report or item that has to be completed and delivered under the terms of an agreement or contract. Products may also be deliverables, e.g., software requirements specifications, detailed design documents.

Develop: To produce or create a product or document and to mature or advanced the product or document content.

Ensure: When software assurance or software safety personnel perform the specified software assurance and software safety activities themselves.

Event: (1) occurrence of a particular set of circumstances (2) external or internal stimulus used for synchronization purposes (3) change detectable by the subject software (4) fact that an action has taken place (5) singular moment in time at which some perceptible phenomenological change (energy, matter, or information) occurs at the port of a unit.

Hazard: A state or a set of conditions, internal or external to a system that has the potential to cause harm.

Hazard Analysis: Identification and evaluation of existing and potential hazards and the recommended mitigation for the hazard sources found.

Hazard Control: Means of reducing the risk of exposure to a hazard.

Hazardous Operation/Work Activity: Hazardous Operation/Work Activity. Any operation or other work activity that, without the implementation of proper mitigations, has a high potential to result in loss of life, serious injury to personnel or public, or damage to property due to the material or equipment involved or the nature of the operation/activity itself.

Independent Verification and Validation (IV&V): Verification and validation performed by an organization that is technically, managerially, and financially independent of the development organization. (Source: ISO/IEC/IEEE 24765)

Inhibit: Design feature that prevents the operation of a function.

Maintain: To continue to have; to keep in existence, to stay up-to-date and correct.

Mission Critical: Item or function that must retain its operational capability to assure no mission failure (i.e., for mission success). (Source: NPR 8715.3)

Monitor: (1) software tool or hardware device that operates concurrently with a system or component and supervises, records, analyzes, or verifies the operation of the system or component; (2) collect project performance data with respect to a plan, produce performance measures, and report and disseminate performance information.

Participate: To be a part of the activity, audit, review, meeting, or assessment.

Perform: Software assurance does the action specified. Perform may include making comparisons of independent results with similar activities performed by engineering, performing audits, and reporting results to engineering.

Product: A result of a physical, analytical, or another process. The item delivered to the customer (e.g., hardware, software, test reports, data), as well as the processes (e.g., system engineering, design, test, logistics) that make the product possible. (Source: NASA-HDBK-8709.22)

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 management structure that initiates and directs one or more projects. A program implements a strategic direction that the Agency has identified as needed to accomplish Agency goals and objectives. (Source: NPR 7120.5)

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. (Source: NPR 7150.2)

Project Manager: The entity or individual who accepts the resulting software products. Project managers are responsible and accountable for the safe conduct and successful outcome of their program or project in conformance with governing Programmatic requirements. This is usually NASA, but can also refer to the Prime contractor-subcontractor relationship as well.

Provider: A person or entity that provides something.

Risk Posture: A characterization of risk based on conditions (e.g., criticality, complexity, environments, performance, cost, schedule) and a set of identified risks, taken as a whole which allows an understanding of the overall risk, or provides a target risk range or level, which can then be used to support decisions being made.

Safe State: A system state in which hazards are inhibited, and all hazardous actuators are in a non-hazardous state. The system can have more than one Safe State.

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

Safety-Critical Software: Software is classified as safety-critical if it meets at least one of the following criteria:

a. Causes or contributes to a system hazardous condition/event,

b. Provides control or mitigation for a system hazardous condition/event,

c. Controls safety-critical functions,

d. Mitigates damage if a hazardous condition/event occurs,

e. Detects, reports, and takes corrective action if the system reaches a potentially hazardous state.

Software: (1) computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system (2) all or a part of the programs, procedures, rules, and associated documentation of an information processing system (3) program or set of programs used to run a computer (4) all or part of the programs which process or support the processing of digital information (5) part of a product that is the computer program or the set of computer programs. The software definition applies to software developed by NASA, software developed for NASA, software maintained by or for NASA, COTS, GOTS, MOTS, OSS, reused software components, auto-generated code, embedded software, the software executed on processors embedded in programmable logic devices, legacy, heritage, applications, freeware, shareware, trial or demonstration software, and open-source software components. (Source: NPR 7150.2)

Software Assurance: The level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life-cycle, and that the software functions in an intended manner.

Software Developer: A person or thing that develops software, based on program/project requirements.

Software Life-Cycle: The period that begins when a software product is conceived and ends when the software is no longer available for use. The software life-cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase.

Software Peer Review: An examination of a software product to detect and identify software anomalies, including errors and deviations from standards and specifications. (Source: IEEE 1028)

Software Safety: The aspects of software engineering, system safety, software assurance and software safety that provide a systematic approach to identifying, analyzing, tracking, mitigating, and controlling hazards and hazardous functions of a system where software may contribute either to the hazard(s) or to its detection, mitigation or control, to ensure safe operation of the system.

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)

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

Supplier: a person or organization that provides something needed, such as a software product or service.

System Safety: Application of engineering and management principles, criteria, and techniques to optimize safety and reduce risks within the constraints of operational effectiveness, time, and cost.

Tailoring: The process used to refine or modify a requirement for a particular project with a justified purpose.

Track: To follow and note the course or progress of the product.

4. SOFTWARE ASSURANCE AND SOFTWARE SAFETY REQUIREMENTS

4.1 Software Assurance Description

4.1.1 Software assurance is the level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at any time during its life-cycle, and that the software functions in an intended manner and that the software does not function in an unintended manner. The objectives of the Software Assurance and Software Safety Standard include:

a. Ensuring that the processes, procedures, and products used to produce and sustain the software conform to all requirements and standards specified to govern those processes, procedures, and products.

b. Ensuring that the software systems are safe and that the software safety-critical requirements and processes are followed.

c. Ensuring that the software systems are secure.

4.1.2 Project and SMA Management support of the software assurance function is essential for software assurance processes to be effective. The software assurance support minimally includes:

a. Management is familiar with and understands the software assurance function's purposes, concepts, practices, and needs.

b. Management provides the software assurance activities with a level of skilled resources (people, equipment, knowledge, methods, facilities, and tools) to accomplish their project responsibilities.

c. Management receives and acts upon information provided by the software assurance function throughout a project.

4.1.3 The Software Assurance and Software Safety Standard's requirements apply to organizations in their roles as both acquirers and providers. A single organization can use the standard in a self-imposed mode or in a multi-party situation. The same organization or different organizations can implement the Software Assurance and Software Safety Standard's requirements, and the job can originate from a Memorandum of Agreement, Memorandum of Understanding, or a contract.

4.2 Safety-Critical Software Determination

4.2.1 Software is classified as safety-critical, if it meets at least one of the following criteria:

a. Causes or contributes to a system hazardous condition/event,

b. Provides control or mitigation for a system hazardous condition/event,

c. Controls safety-critical functions,

d. Mitigates damage if a hazardous condition/event occurs,

e. Detects, reports, and takes corrective action, if the system reaches a potentially hazardous state.

Note: Software is classified as safety-critical if the software is determined by and traceable to hazard analysis. See appendix A for guidelines associated with addressing software in hazard definitions. See SWE-205. Consideration for other independent means of protection (software, hardware, barriers, or administrative) should be a part of the system hazard definition process.

4.3 Software Assurance and Software Safety Requirements

4.3.1 The responsible project manager shall perform the software assurance and software safety activities defined in Table 1 per the requirements marked applicable in the software engineering requirements mapping matrix for the software component. (SASS-01) In this document, the phrase “Software Assurance and Software Safety Tasks” means that the roles and responsibilities for completing these requirements may be delegated within the project consistent with the scope and scale of the project. The Center SMA Director designates SMA TA(ies) for programs, facilities, and projects, providing direction, functional oversight, and assessment for all Agency software assurance and software safety activities.

Table 1. Software Assurance and Software Safety Requirements Mapping Matrix

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3		Software Management Requirements	
3.1		Software Life-Cycle Planning	
3.1.2	033	The project manager shall assess options for software acquisition versus development.	1. Confirm that the options for software acquisition versus development have been evaluated. 2. Confirm the flow down of applicable software engineering, software assurance, and software safety requirements on all acquisition activities. (NPR 7150.2 and NASA-STD-8739.8). 3. Assess any risks with acquisition versus development decision(s).
3.1.3	013	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.	1. Confirm that all plans are in place, and have expected content for the life-cycle events, with proper tailoring for the classification of the software. 2. Develop a Software Assurance Plan following the content defined in NASA-HDBK-2203 for a software assurance plan, including software safety.
3.1.4	024	The project manager shall track the actual results and performance of software activities against the software plans. a. Corrective actions are taken, recorded, and managed to closure. b. Including changes to commitments (e.g., software plans) that have been agreed to by the affected groups and individuals.	1. Assess plans for compliance with NPR 7150.2 requirements, NASA-STD-8739.8, including changes to commitments. 2. Confirm that closure of corrective actions associated with the performance of software activities against the software plans, including closure rationale.
3.1.5	034	The project manager shall define and document the acceptance criteria for the software.	1. Confirm software acceptance criteria are defined and assess the criteria based on guidance in the NASA Software Engineering Handbook, NASA-HDBK-2203.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.1.6	036	The project manager shall establish and maintain the software processes, software documentation plans, list of developed electronic products, deliverables, and list of tasks for the software development that are required for the project's software developers, as well as the action required (e.g., approval, review) of the Government upon receipt of each of the deliverables.	1. Confirm the following are approved, implemented, and updated per requirements: a. Software processes, including software assurance, software safety, and IV&V processes. b. Software documentation plans, c. List of developed electronic products, deliverables, and d. List of tasks required or needed for the project's software development. 2. Confirm that any required government actions upon receipt of deliverables (e.g., approvals, reviews) are established and maintained.
3.1.7	037	The project manager shall define and document the milestones at which the software developer(s) progress will be reviewed and audited.	1. Confirm that milestones for reviewing and auditing software developer progress are defined and documented. 2. Participate in project milestones reviews.
3.1.8	039	The project manager shall require the software developer(s) to periodically report status and provide insight into software development and test activities; at a minimum, the software developer(s) will be required to allow the project manager and software assurance personnel to: a. Monitor product integration. b. Review the verification activities to ensure adequacy. c. Review trade studies and source data. d. Audit the software development processes and practices. e. Participate in software reviews and technical interchange meetings.	1. Confirm that software developer(s) are periodically reporting status and providing insight to the project manager. 2. Monitor product integration. 3. Analyze the verification activities to ensure adequacy. 4. Assess trade studies, source data, software reviews, and technical interchange meetings. 5. Perform audits on software development processes and practices at least once every two years. 6. Develop and provide status reports. 7. Develop and maintain a list of all software assurance review discrepancies, risks, issues, findings, and concerns.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			8. Confirm that the project manager provides responses to software assurance and software safety submitted issues, findings, and risks and that the project manager tracks software assurance and software safety issues, findings, and risks to closure.
3.1.9	040	The project manager shall require the software developer(s) to provide NASA with software products, traceability, software change tracking information and nonconformances, in electronic format, including software development and management metrics.	1. Confirm that software artifacts are available in electronic format to NASA.
3.1.10	042	The project manager shall require the software developer(s) to provide NASA with electronic access to the source code developed for the project in a modifiable format.	1. Confirm that software developers are providing NASA with electronic access to the source code generated for the project in a modifiable form.
3.1.11	139	The project manager shall comply with the requirements in this NPR that are marked with an "X" in Appendix C consistent with their software classification.	1. Assess that the software requirements, products, procedures, and processes of the project are compliant with the NPR 7150.2 requirements per the software classification and safety criticality for software.
3.1.12	121	Where approved, the project manager shall document and reflect the tailored requirement in the plans or procedures controlling the development, acquisition, and deployment of the affected software.	1. Confirm that any requirement tailoring in the Requirements Mapping Matrix has the required approvals. 2. Develop a tailoring matrix of software assurance and software safety requirements.
3.1.13	125	Each project manager with software components shall maintain a requirements mapping matrix or multiple requirements mapping matrices against requirements in this NPR, including those	1. Confirm that the project is maintaining a requirement mapping matrix (matrices) for all requirements in NPR 7150.2.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		delegated to other parties or accomplished by contract vehicles or Space Act Agreements.	2. Maintain the requirement mapping matrix (matrices) for requirements in NASA-STD-8739.8.
3.1.14	027	The project manager shall satisfy the following conditions when a Commercial-Off-The-Shelf (COTS), Government Off-The-Shelf (GOTS), Modified Off-The-Shelf (MOTS), Open Source Software (OSS), or reused software component is acquired or used: a. The requirements to be met by the software component are identified. b. The software component includes documentation to fulfill its intended purpose (e.g., usage instructions). c. Proprietary rights, usage 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.	1. Confirm that the conditions listed in "a" through "f" are complete for any COTS, GOTS, MOTS, OSS, or reused software that is acquired or used.
3.2		Software Cost Estimation	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.2.1	015	To better estimate the cost of development, the project manager shall establish, document, and maintain: a. Two cost estimate models and associated cost parameters for all Class A and B software projects that have an estimated project cost of $2 million or more. b. One software cost estimate model and associated cost parameter(s) for all Class A and Class B software projects that have an estimated project cost of less than $2 million. c. One software cost estimate model and associated cost parameter(s) for all C and D software projects. d. One software cost estimate model and associated cost parameter(s) for all Class F software projects.	1. Confirm that the required number of software cost estimates are complete and include software assurance cost estimate(s) for the project, including a cost estimate associated with handling safety-critical software and safety-critical data.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.2.2	151	The project manager's software cost estimate(s) shall satisfy the following conditions: a. Covers the entire software life-cycle. b. Is based on selected project attributes (e.g., assessment of the size, functionality, complexity, criticality, reuse code, modified code, and risk of the software processes and products). c. Is based on the cost implications of the technology to be used and the required maturation of that technology. d. Incorporates risk and uncertainty, including cybersecurity. e. Includes the cost of the required software assurance support. f. Includes other direct costs.	1. Assess the project's software cost estimate(s) to determine if the stated criteria listed in "a" through "f" are satisfied.
3.2.3	174	The project manager shall submit software planning parameters, including size and effort estimates, milestones, and characteristics, to the Center measurement repository at the conclusion of major milestones.	1. Confirm that all the software planning parameters, including size and effort estimates, milestones, and characteristics, are submitted to a Center repository. 2. Confirm that all software assurance and software safety software estimates and planning parameters are submitted to an organizational repository.
3.3		Software Schedules	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.3.1	016	The project manager shall document and maintain a software schedule that satisfies the following conditions: a. Coordinates with the overall project schedule. b. Documents the interactions of milestones and deliverables between software, hardware, operations, and the rest of the system. c. Reflects the critical dependencies for software development activities. d. Identifies and accounts for dependencies with other projects and cross-program dependencies.	1. Assess that the software schedule satisfies the conditions in the requirement. 2. Develop a software assurance schedule, including software assurance products, audits, reporting, and reviews.
3.3.2	018	The project manager shall regularly hold reviews of software schedule activities, metrics, status, and results with the project stakeholders and track issues to resolution.	1. Confirm the generation and distribution of periodic reports on software schedule activities, metrics, and status, including reports of software assurance and software safety schedule activities, metrics, and status. 2. Confirm closure of any project software schedule issues.
3.3.3	046	The project manager shall require the software developer(s) to provide a software schedule for the project's review, and schedule updates as requested.	1. Confirm the project's schedules are updated.
3.4		Software Training	
3.4.1	017	The project manager shall plan, track, and ensure project specific software training for project personnel.	1. Confirm that any project-specific software training has been planned, tracked, and completed for project personnel, including software assurance and software safety personnel.
3.5		Software Classification Assessments	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.5.1	020	The project manager shall classify each system and subsystem containing software in accordance with the highest applicable software classification definitions for Classes A, B, C, D, E, and F software in Appendix D.	1. Perform a software classification or concur with the engineering software classification of software per the descriptions in NPR 7150.2.
3.5.2	176	The project manager shall maintain records of each software classification determination, each software Requirements Mapping Matrix, and the results of each software independent classification assessment for the life of the project.	1. Confirm that records of the software Requirements Mapping Matrix and each software classification are maintained and updated for the life of the project.
3.6		Software Assurance and Software Independent Verification & Validation	
3.6.1	022	The project manager shall plan and implement software assurance per NASA-STD-8739.8.	1. Perform according to the software assurance plan and the software assurance and software safety standard requirements in NASA-STD-8739.8.
3.6.2	141	For projects reaching Key Decision Point A, the program manager shall ensure that software IV&V is performed on the following categories of projects: a. Category 1 projects as defined in NPR 7120.5. b. Category 2 projects as defined in NPR 7120.5 that have Class A or Class B payload risk classification per NPR 8705.4. c. Projects selected explicitly by the NASA Chief of the Office of Safety and Mission Assurance to have software IV&V.	1. Confirm that IV&V requirements (section 4.4) are complete on projects that are required to have IV&V.
3.6.3	131	If software IV&V is performed on a project, the project manager shall ensure an IPEP is	1. Confirm that the IV&V Program Execution Plan (IPEP) exists.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		developed, negotiated, approved, maintained, and executed.	
3.6.4	178	If software IV&V is performed on a project, the project manager shall ensure that IV&V is provided access to development artifacts, products, source code, and data required to perform the IV&V analysis efficiently and effectively.	1. Confirm that IV&V has access to the software development artifacts, products, source code, and data required to perform the IV&V analysis efficiently and effectively.
3.6.5	179	If software IV&V is performed on a project, the project manager shall provide responses to IV&V submitted issues and risks, and track these issues and risks to closure.	1. Confirm that the project manager provides responses to IV&V submitted issues, findings, and risks and that the project manager tracks IV&V issues, findings, and risks to closure.
3.7		Safety-critical Software	
3.7.1	205	The project manager, in conjunction with the SMA organization, shall determine if each software component is considered to be safety-critical per the criteria defined in NASA-STD-8739.8.	1. Confirm that the hazard reports or safety data packages contain all known software contributions or events where software; either by its action, inaction or incorrect action, lead to a hazard. 2. Assess that the hazard reports identify the software components associated with the system hazards per the criteria defined in NASA-STD- 8739.8, Appendix A. 3. Assess that hazard analyses (including hazard reports) identify the software components associated with the system hazards per the criteria defined in NASA-STD- 8739.8, Appendix A. 4. Confirm that the traceability between software requirements and hazards with software contributions exist. 5. Develop and maintain a software safety analysis throughout the software development life-cycle.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.7.2	023	If a project has safety-critical software, the project manager shall implement the safety-critical software requirements contained in NASA-STD-8739.8.	1. Confirm that the identified safety-critical software components and data have implemented the safety-critical software assurance requirements listed in this standard.
3.7.3	134	If a project has safety-critical software or mission-critical software, the project manager shall implement the following items in the software: a. The software is initialized, at first start and restarts, to a known safe state. b. The software safely transitions between all predefined known states. c. Termination performed by software of functions is performed to a known safe state. d. Operator overrides of software functions require at least two independent actions by an operator. e. Software rejects commands received out of sequence when execution of those commands out of sequence can cause a hazard. f. The software detects inadvertent memory modification and recovers to a known safe state. g. The software performs integrity checks on inputs and outputs to/from the software system. h. The software performs prerequisite checks prior to the execution of safety-critical software commands. i. No single software event or action is allowed to initiate an identified hazard. j. The software responds to an off-nominal	1. Analyze the software requirements and the software design and work with the project to implement NPR 7150.2 requirement items "a" through "l." 2. Assess that the source code satisfies the conditions in the NPR 7150.2 requirement "a" through "l" for safety-critical and mission-critical software at each code inspection, test review, safety review, and project review milestone. 3. Confirm 100% code test coverage is addressed for all identified software safety-critical software components or assure that software developers provide a risk assessment explaining why the test coverage is not possible for the safety-critical code component. 4. Confirm that all identified safety-critical software components have a cyclomatic complexity value of 15 or lower. If not, assure that software developers provide a risk assessment explaining why the cyclomatic complexity value needs to be higher than 15 and why the software component cannot be structured to be lower than 15. 5. Confirm that the values of the safety-critical loaded data, uplinked data, rules, and scripts that affect hazardous system behavior have been tested. 6. Analyze the software design to ensure:

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		condition within the time needed to prevent a hazardous event. k. The software provides error handling. l. The software can place the system into a safe state.	a. Use of partitioning or isolation methods in the design and code, b. That the design logically isolates the safety-critical design elements and data from those that are non-safety-critical. 7. Participate in software reviews affecting safety-critical software products.
3.8		Automatic Generation of Software Source Code	
3.8.1	146	The project manager shall define the approach to the automatic generation of the software source code, including: a. Validation and verification of auto-generation tools. b. Configuration management of the auto-generation tools and associated data. c. Description of the limits and the allowable scope for the use of the auto-generated software. d. Verification and validation of auto-generated source code using the same software standards and processes as hand-generated code. e. Monitoring the actual use of auto-generated source code compared to the planned use. 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.	1. Assess that the approach for the auto-generation software source code is defined, and the approach satisfies at least the conditions “a” through “g.”

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.8.2	206	The project manager shall require the software developers and suppliers to provide NASA with electronic access to the models, simulations, and associated data used as inputs for auto-generation of software.	1. Confirm that NASA, engineering, project, software assurance, and IV&V have electronic access to the models, simulations, and associated data used as inputs for auto-generation of software.
3.9		Software Development Processes and Practices	
3.9.3	032	The project manager shall acquire, develop, and maintain software from an organization with a non-expired CMMI-DEV rating as measured by a CMMI Institute Certified Lead Appraiser as follows: a. For Class A software: CMMI-DEV Maturity, Level 3 Rating, or higher for software. b. For Class B software (except Class B software on NASA Class D payloads, as defined in NPR 8705.4): CMMI-DEV Maturity Level 2 Rating or higher for software.	1. Confirm that Class A and B software that is acquired, developed, and maintained by NASA is performed by an organization with a non-expired CMMI-DEV rating, as per the NPR 7150.2 requirement. 2. Assess potential process-related issues, findings, or risks identified from the CMMI assessment findings. 3. Perform audits on the software development and software assurances processes.
3.10		Software Reuse	
3.10.1	147	The project manager shall specify reusability requirements that apply to its software development activities to enable future reuse of the software, including the models, simulations, and associated data used as inputs for auto-generation of software, for United States Government purposes.	1. Confirm that the project has considered reusability for its software development activities.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.10.2	148	The project manager shall evaluate software for potential reuse by other projects across NASA and contribute reuse candidates to the NASA Internal Sharing and Reuse Software systems, however, if the project manager is a contractor then a civil servant must pre-approve all such software contributions; all software contributions should include, at a minimum, the following information: a. Software Title. b. Software Description. c. The Civil Servant Software Technical Point of Contact for the software product. d. The language or languages used to develop the software. e. Any third party code contained therein, and the record of the requisite license or permission received from the third party permitting the Government's use, if applicable.	1. Confirm that any project software contributed as a reuse candidate has the identified information in items "a" through "e."
3.11		Software Cybersecurity	
3.11.2	156	The project manager shall perform a software cybersecurity assessment on the software components per the Agency security policies and the project requirements, including risks posed by the use of COTS, GOTS, MOTS, OSS, or reused software components.	1. Confirm the project has performed a software cybersecurity assessment on the software components per the Agency security policies and the project requirements, including risks posed by the use of COTS, GOTS, MOTS, OSS, or reused software components.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.11.3	154	The project manager shall identify cybersecurity risks, along with their mitigations, in flight and ground software systems, and plan the mitigations for these systems.	1. Confirm that cybersecurity risks, along with their mitigations, are identified and managed.
3.11.4	157	The project manager shall implement protections for software systems with communications capabilities against unauthorized access.	1. For software products with communications capabilities, confirm that the software requirements and software design documentation addresses unauthorized access.
3.11.5	158	The project manager shall ensure that space flight software systems are assessed for possible cybersecurity vulnerabilities and weaknesses.	1. Confirm that space or flight software systems have been assessed for potential cybersecurity vulnerabilities and weaknesses. 2. Perform static code analysis on the software or analyze the project's static code analysis tool results for cybersecurity vulnerabilities and weaknesses.
3.11.6	155	The project manager shall address identified cybersecurity vulnerabilities and weaknesses.	1. Confirm that the project addresses the engineering and assurance identified cybersecurity vulnerabilities and weaknesses.
3.11.7	159	The project manager shall test the software and record test results for the required software cybersecurity mitigation implementations identified from the security vulnerabilities and security weaknesses analysis.	1. Confirm that testing is complete for the cybersecurity mitigation. 2. Assess the quality of the cybersecurity mitigation implementation testing and the test results.
3.11.8	207	The project manager shall identify, record, and implement secure coding practices.	1. Assess that the software coding guidelines (e.g., coding standards) includes secure coding practices.
3.11.9	185	The project manager shall verify that the software code meets the project's secure coding standard by using the results from static analysis tool(s).	1. Analyze the engineering data or perform independent static code analysis to verify that the code meets the project's secure coding standard requirements.
3.12		Software Bi-Directional Traceability	

<table>
<tr><th>NPR 7150.2 Section</th><th>SWE #</th><th>NPR 7150.2 Requirement</th><th>Software Assurance and Software Safety Tasks</th></tr>
<tr><td>3.12.1</td><td>052</td><td>The project manager shall perform, record, and maintain bi-directional traceability between the following software elements:
<table>
<tr><th>Bi-directional Traceability</th><th>Class A, B, and C</th><th>Class D</th></tr>
<tr><td>Higher-level requirements to the software requirements</td><td>X</td><td></td></tr>
<tr><td>Software requirements to the system hazards</td><td>X</td><td>X</td></tr>
<tr><td>Software requirements to the software design components</td><td>X</td><td></td></tr>
<tr><td>Software design components to the software code</td><td>X</td><td></td></tr>
<tr><td>Software requirements to the software test procedures</td><td>X</td><td>X</td></tr>
<tr><td>Software requirements to the software non-conformances</td><td>X</td><td>X</td></tr>
</table></td><td>1. Confirm that bi-directional traceability has been completed, recorded, and maintained.
2. Confirm that the software traceability includes traceability to any hazard that includes software.</td></tr>
<tr><td>4</td><td></td><td>Software Engineering (Life-Cycle) Requirements</td><td></td></tr>
</table>

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.1		Software Requirements	
4.1.2	050	The project manager shall establish, capture, record, approve, and maintain software requirements, including requirements for COTS, GOTS, MOTS, OSS, or reused software components, as part of the technical specification.	1. Confirm that all software requirements are established, captured, and documented as part of the technical specification, including requirements for COTS, GOTS, MOTS, OSS, or reused software components.
4.1.3	051	The project manager shall perform software requirements analysis based on flowed-down and derived requirements from the top-level systems engineering requirements, safety and reliability analyses, and the hardware specifications and design.	1. Perform a software assurance analysis on the detailed software requirements to analyze the software requirement sources and identify any incorrect, missing, or incomplete requirements.
4.1.4	184	The project manager shall include software related safety constraints, controls, mitigations, and assumptions between the hardware, operator, and software in the software requirements documentation.	1. Analyze that the software requirements documentation contains the software related safety constraints, controls, mitigations, and assumptions between the hardware, operator, and the software.
4.1.5	053	The project manager shall track and manage changes to the software requirements.	1. Confirm the software requirements changes are documented, tracked, approved, and maintained throughout the project life-cycle.
4.1.6	054	The project manager shall identify, initiate corrective actions, and track until closure inconsistencies among requirements, project plans, and software products.	1. Monitor identified differences among requirements, project plans, and software products to confirm they are addressed.
4.1.7	055	The project manager shall perform requirements validation to ensure that the software will perform as intended in the customer environment.	1. Confirm that the project software testing has shown that software will function as expected in the customer environment.
4.2		Software Architecture	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.2.3	057	The project manager shall transform the requirements for the software into a recorded software architecture.	1. Assess that the software architecture addresses or contains the software structure, qualities, interfaces, and external/internal components. 2. Analyze the software architecture to assess whether software safety and mission assurance requirements are met.
4.2.4	143	The project manager shall perform a software architecture review on the following categories of projects: a. Category 1 Projects as defined in NPR 7120.5. b. Category 2 Projects as defined in NPR 7120.5 that have Class A or Class B payload risk classification per NPR 8705.4.	1. Assess the results of or participate in software architecture review activities held by the project.
4.3		Software Design	
4.3.2	058	The project manager shall develop, record, and maintain a software design based on the software architectural design that describes the lower-level units so that they can be coded, compiled, and tested.	1. Assess the software design against the hardware and software requirements, and identify any gaps. 2. Assess the software design to verify that the design is consistent with the software architectural design concepts and that the software design describes the lower-level units to be coded, compiled, and tested. 3. Assess that the design does not introduce undesirable behaviors or unnecessary capabilities. 4. Confirm that the software design implements all of the required safety-critical functions and requirements. 5. Perform a software assurance design analysis.
4.4		Software Implementation	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.4.2	060	The project manager shall implement the software design into software code.	1. Confirm that the software code implements the software designs. 2. Confirm that the code does not contain functionality not defined in the design or requirements.
4.4.3	061	The project manager shall select and adhere to software coding methods, standards, and criteria.	1. Analyze that the software code conforms to all of the required software coding methods, rules, and principles.
4.4.4	135	The project manager shall use static analysis tools to analyze the code during the development and testing phases to detect defects, software security, and coding errors.	1. Confirm the static analysis tool(s) are used with checkers to identify security and coding errors, and defects. 2. Assess that the project addresses the results from the static analysis tools used by software assurance, software safety, engineering, or the project. 3. Confirm that the software code has been scanned for security defects and confirm the result.
4.4.5	062	The project manager shall unit test the software code.	1. Confirm that the project successfully executes the required unit tests, particularly those testing safety-critical functions. 2. Confirm that the project addresses or otherwise tracks to closure errors, defects, or problem reports found during unit test.
4.4.6	186	The project manager shall assure that the unit test results are repeatable.	1. Confirm that the project maintains the procedures, scripts, results, and data needed to repeat the unit testing (e.g., as-run scripts, test procedures, results).
4.4.7	063	The project manager shall provide a software version description for each software release.	1. Confirm that the project creates a correct software version description for each software release. 2. For each software release, confirm that the software has been scanned for security defects and coding standard compliance and confirm the results.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.4.8	136	The project manager shall validate and accredit the software tool(s) required to develop or maintain software.	1. Confirm that the software tool(s) needed to create and maintain software is validated and accredited.
4.5		Software Testing	
4.5.2	065a	The project manager shall establish and maintain: a. Software test plan(s). …	1. Confirm that software test plans have been established, contain correct content, and are maintained. 2. Confirm that the software test plan addresses the verification of safety-critical software, specifically the off-nominal scenarios.
4.5.2	065b	The project manager shall establish and maintain: ... b. Software test procedure(s). …	1. Confirm that test procedures have been established and are updated when changes to tests or requirements occur. 2. Analyze the software test procedures for: a. Coverage of the software requirements. b. Acceptance or pass/fail criteria, c. The inclusion of operational and off-nominal conditions, including boundary conditions, d. Requirements coverage and hazards per SWE-66 and SWE-192, respectively.
4.5.2	065c	The project manager shall establish and maintain: ... c. Software test report(s).	1. Confirm that the project creates and maintains the test reports throughout software integration and test. 2. Confirm that the project records the test report data and that the data contains the as-run test data, the test results, and required approvals. 3. Confirm that the project records all issues and discrepancies found during each test. 4. Confirm that the project tracks to closure errors, defects, etc. found during testing.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.5.3	066	The project manager shall test the software against its requirements.	1. Confirm test coverage of the requirements through the execution of the test procedures. 2. Perform test witnessing for safety-criticality software. 3. Ensure that any newly identified software contributions to hazards, events, or conditions found during testing are in the system safety data package.
4.5.4	187	The project manager shall place software items under configuration management prior to testing.	1. Confirm that software items to be tested are under configuration management before the start of testing. 2. Confirm the project maintains the software items under configuration management through the completion of testing.
4.5.5	068	The project manager shall evaluate test results and record the evaluation.	1. Confirm that test results are assessed and recorded. 2. Confirm that the project documents software non-conformances in a tracking system. 3. Confirm that test results are sufficient verification artifacts for the hazard reports.
4.5.6	070	The project manager shall use validated and accredited software models, simulations, and analysis tools required to perform qualification of flight software or flight equipment.	1. Confirm the software models, simulations, and analysis tools used to achieve the qualification of flight software or flight equipment have been validated and accredited.
4.5.7	071	The project manager shall update the software test plan(s) and the software test procedure(s) to be consistent with software requirements.	1. Analyze that software test plans and software test procedures cover the software requirements and provide adequate verification of hazard controls, specifically the off-nominal scenarios.
4.5.8	073	The project manager shall validate the software system on the targeted platform or high-fidelity simulation.	1. Confirm that the project validates the software components on the targeted platform or a high-fidelity simulation.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.5.9	189	The project manager shall ensure that the code coverage measurements for the software are selected, implemented, tracked, recorded, and reported.	1. Confirm that code coverage measurements have been selected, performed, tracked, recorded, and communicated with each release.
4.5.10	190	The project manager shall verify code coverage is measured by analysis of the results of the execution of tests.	1. Confirm that the project performs code coverage analysis using the results of the tests or by use of a code coverage tool. 2. Analyze the code coverage measurements for uncovered software code. 3. Assess any uncovered software code for potential risk, issues, or findings.
4.5.11	191	The project manager shall plan and conduct software regression testing to demonstrate that defects have not been introduced into previously integrated or tested software and have not produced a security vulnerability.	1. Confirm that the project plans regression testing and that the regression testing is adequate and includes retesting of all safety-critical code components. 2. Confirm that the project performs the planned regression testing. 3. Identify any risks and issues associated with the regression test set selection and execution. 4. Confirm that the regression test procedures are updated to incorporate tests that validate the correction of critical anomalies.
4.5.12	192	The project manager shall verify through test the software requirements that trace to a hazardous event, cause, or mitigation technique.	1. Confirm that the project verifies the software requirements, which trace to a hazardous event, cause, or mitigation techniques, through testing.
4.5.13	193	The project manager shall develop acceptance tests for loaded or uplinked data, rules, and code that affects software and software system behavior.	1. Confirm that the project develops acceptance tests for loaded or uplinked data, rules, and code that affect software and software system behavior. 2. Confirm that the loaded or uplinked data, rules, scripts, or code that affects software and software

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			system behavior is baselined in the software configuration system.
4.5.14	211	The project manager shall test embedded COTS, GOTS, MOTS, OSS, or reused software components to the same level required to accept a custom developed software component for its intended use.	1. Confirm that the project is testing COTS, GOTS, MOTS, OSS, or reused software components to the same level as developed software for its intended use.
4.6		Software Operations, Maintenance, and Retirement	
4.6.2	075	The project manager shall plan and implement software operations, maintenance, and retirement activities.	1. Assess the plans for maintenance, operations, and retirement for completeness of the required software engineering and software assurance activities. 2. Confirm that the project implements software operations, software maintenance, and software retirement plans.
4.6.3	077	The project manager shall complete and deliver the software product to the customer with appropriate records, including as-built records, to support the operations and maintenance phase of the software's life-cycle.	1. Confirm that the correct version of the products is provided, including as-built documentation and project records. 2. Perform audits on the configuration management processes to verify that all products are being delivered and are the correct versions.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.6.4	194	The project manager shall complete, prior to delivery, verification that all software requirements identified for this delivery have been met, that all approved changes have been implemented and that all defects designated for resolution prior to delivery have been resolved.	1. Confirm that the project has identified the software requirements to be met, the approved changes to be implemented, and defects to be resolved for each delivery. 2. Confirm that the project has met all software requirements identified for the delivery. 3. Confirm that approved changes have been implemented and tested. 4. Confirm that the approved changes to be implemented and the defects to be resolved have been resolved. 5. Approve or sign-off on the projects delivered products.
4.6.5	195	The project manager shall maintain the software using standards and processes, per the applicable software classification throughout the maintenance phase.	1. Perform audits on the standards and processes used throughout maintenance based on the software classification.
4.6.6	196	The project manager shall identify the records and software tools to be archived, the location of the archive, and procedures for access to the products for software retirement or disposal.	1. Confirm that the project has identified the records and software tools for archival. 2. Confirm the project has an archival location and the procedures for archiving and accessing products for software retirement or disposal. 3. Confirm that the project archives all software and records as planned.
5		Supporting Software Life-Cycle Requirements	
5.1		Software Configuration Management	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
5.1.2	079	The project manager shall develop a software configuration management plan that describes the functions, responsibilities, and authority for the implementation of software configuration management for the project.	1. Assess that a software configuration management plan has been developed and complies with the requirements in NPR 7150.2 and Center/project guidance.
5.1.3	080	The project manager shall track and evaluate changes to software products.	1. Analyze proposed software and hardware changes to software products for impacts, particularly to safety and security. 2. Confirm: a. that the project tracks the changes, b. that the changes are approved and documented before implementation, c. that the implementation of changes is complete, and d. that the project tests the changes. 3. Confirm software changes follow the software change control process.
5.1.4	081	The project manager shall identify the software configuration items (e.g., software records, code, data, tools, models, scripts) and their versions to be controlled for the project.	1. Confirm that the project has identified the configuration items and their versions to be controlled. 2. Assess that the software safety-critical items are configuration managed, including hazard reports and safety analysis.
5.1.5	082	The project manager shall establish and implement procedures to: 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.	1. Confirm that software assurance has participation in software control activities. 2. Perform an audit against the configuration management procedures to confirm that the project is following the established procedures.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		c. Identify the persons or groups to make changes at each level.	
5.1.6	083	The project manager shall prepare and maintain records of the configuration status of software configuration items.	1. Confirm that the project maintains records of the configuration status of the configuration items.
5.1.7	084	The project manager shall perform software configuration audits to determine the correct version of the software configuration items and verify that they conform to the records that define them.	1. Confirm that the project manager performed software configuration audits to determine the correct version of the software configuration items and verify that the results of the audit conform to the records that define them.
5.1.8	085	The project manager shall establish and implement procedures for the storage, handling, delivery, release, and maintenance of deliverable software products.	1. Confirm that the project establishes procedures for storage, processing, distribution, release, and support of deliverable software products. 2. Perform audits on the project to ensure that the project is following defined procedures for deliverable software products.
5.1.9	045	The project manager shall participate in any joint NASA/developer audits.	1. Participate in or assess the results from any joint NASA/developer audits. Track any findings to closure.
5.2		Software Risk Management	
5.2.1	086	The project manager shall record, analyze, plan, track, control, and communicate all of the software risks and mitigation plans.	1. Confirm and assess that a risk management process includes recording, analyzing, planning, tracking, controlling, and communicating all of the software risks and mitigation plans. 2. Perform audits on the risk management process for the software activities.
5.3		Software Peer Reviews/Inspections	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
5.3.2	087	The project manager shall perform and report the results of software peer reviews or software inspections for: a. Software requirements. b. Software plans. 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.	1. Confirm that software peer reviews are performed and reported on for project activities. 2. Confirm that the project addresses the accepted software peer review findings. 3. Perform peer reviews on software assurance and software safety plans. 4. Confirm that the source code satisfies the conditions in the NPR 7150.2 requirement SWE-134, "a" through "l," based upon the software functionality for the applicable safety-critical requirements at each code inspection/review. 5. For code peer reviews, confirm that all identified software safety-critical components have a cyclomatic complexity value of 15 or lower or develop a risk for the software safety-critical components that have a cyclomatic complexity value over 15.
5.3.3	088	The project manager shall, for each planned software peer review or software inspection: a. Use a checklist or formal reading technique (e.g., perspective based reading) to evaluate the work products. b. Use established readiness and completion criteria. c. Track actions identified in the reviews until they are resolved. d. Identify the required participants.	1. Confirm that the project meets the NPR 7150.2 criteria in "a" through "d" for each software peer review. 2. Confirm that the project resolves the actions identified from the software peer reviews. 3. Perform audits on the peer-review process.
5.3.4	089	The project manager shall, for each planned software peer review or software inspection, record necessary measurements.	1. Confirm that the project records the software peer reviews or software inspection results measurements.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
5.4		Software Measurements	
5.4.2	090	The project manager shall establish, record, maintain, report, and utilize software management and technical measurements.	1. Confirm that a measurement program establishes, records, maintains, reports, and uses software assurance, management, and technical measures. 2. Perform trending and analyses on metrics (quality metrics, defect metrics) and report.
5.4.3	093	The project manager shall analyze software measurement data collected using documented project-specified and Center/organizational analysis procedures.	1. Confirm software measurement data analysis conforms to documented analysis procedures. 2. Analyze software assurance measurement data collected.
5.4.4	094	The project manager shall provide access to the software measurement data, measurement analyses, and software development status as requested to the sponsoring Mission Directorate, the NASA Chief Engineer, the Center Technical Authorities and Headquarters SMA.	1. Confirm access to software measurement data, analysis, and status as requested to the following entities: - Sponsoring Mission Directorate - NASA Chief Engineer - Center Technical Authorities - Headquarters SMA
5.4.5	199	The project manager shall monitor measures to ensure the software will meet or exceed performance and functionality requirements, including satisfying constraints.	1. Confirm the project monitors and updates planned measurements to assure the software will meet or exceed performance and functionality requirements, including satisfying constraints. 2. Monitor and track any performance or functionality requirements that are not being met or are at risk of not being met.
5.4.6	200	The project manager shall collect, track, and report software requirements volatility metrics.	1. Confirm that the project collects, tracks, and reports on the software volatility metrics. 2. Analyze software volatility measures to evaluate requirements stability as an early indicator of project problems.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
5.5		Software Non-conformance or Defect Management	
5.5.1	201	The project manager shall track and maintain software non-conformances (including defects in tools and appropriate ground software).	1. Confirm that all software non-conformances are recorded and tracked to resolution. 2. Confirm that accepted non-conformances include the rationale for the non-conformance.
5.5.2	202	The project manager shall define and implement clear software severity levels for all software non-conformances (including tools, COTS, GOTS, MOTS, OSS, reused software components, and applicable ground systems).	1. Confirm that all software non-conformances severity levels are defined. 2. Assess the application and accuracy of the defined severity levels to software non-conformances. 3. Confirm that the project assigns severity levels to non-conformances associated with tools, COTS, GOTS, MOTS, OSS, reused software components, and applicable ground systems. 4. Maintain or have access to the number of software nonconformances at each severity level for each software configuration item.
5.5.3	203	The project manager shall implement mandatory assessments of reported non-conformances for all COTS, GOTS, MOTS, OSS, or reused software components.	1. Confirm the evaluations of reported non-conformances for all COTS, GOTS, MOTS, OSS, or reused software components are occurring throughout the project life-cycle. 2. Assess the impact of non-conformances on the safety, quality, and reliability of the project software.
5.5.4	204	The project manager shall implement process assessments for all high severity software non-conformances (closed loop process).	1. Perform or confirm that a root cause analysis has been completed on all identified high severity software nonconformances, the results are recorded, and that the results have been assessed for adequacy. 2. Confirm that the project analyzed the processes identified in the root cause analysis associated with the high severity software non-conformances.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			3. Assess opportunities for process improvement on the processes identified in the root cause analysis associated with the high severity software non-conformances. 4. Perform or confirm tracking of corrective actions to closure on high severity software non-conformances.

4.4 Independent Verification &Validation (IV&V)

4.4.1 The requirements in the IV&V section are applicable to all IV&V efforts performed on a software development project, as tailored by the IV&V Project Execution Plan. It also serves as the definition of what NASA considers to be IV&V. IV&V is a risk mitigation activity, and as such, the application of IV&V analysis and the rigor of that analysis is driven by the IV&V Provider's assessment of software risk.

4.4.2 IV&V Overview

4.4.2.1 IV&V is a technical discipline of software assurance, which employs rigorous analysis and testing methodologies identifying objective evidence and conclusions to provide a second independent assessment of critical products and processes throughout the life-cycle. The secondary evaluation of products and processes throughout the life-cycle contributes to the demonstration of whether the software is fit for nominal operations (required functionality, safety, dependability, etc.), and off-nominal conditions (response to faults, responses to hazardous conditions, etc.). The goal of the IV&V effort is to contribute to the assurance conclusions to the project and stakeholders based on evidence found in software development artifacts and risks associated with the intended behaviors of the software.

4.4.2.2 Three parameters define the independence of IV&V: technical independence, managerial independence, and financial independence.

a. Technical independence requires that the personnel performing the IV&V analysis are not involved in the development of the system or its elements. The IV&V team establishes an understanding of the problem and how the system addresses the problem. It is through technical independence that the IV&V team's different perspective allows it to detect subtle errors overlooked by those personnel focused on developing the system.

b. Managerial independence requires that the personnel performing the IV&V analysis are not in the same organization as the development and program management team. Managerial independence also means that the IV&V team makes its own decisions as to which segments of the system and its software to analyze and test, chooses the IV&V analysis methods to apply, and defines the IV&V schedule of activities. While being independent from the development and program management organization, the IV&V team does provide its findings, in a timely manner, to both of those organizations. The submission of findings to the program management organization should not include any restrictions (e.g., requiring the approval of the development organization) or any other adverse pressures from the development group.

c. Financial independence requires that the control of the IV&V budget be vested in a group independent of the software development organization. Financial independence does not necessarily mean that the IV&V team controls the budget, but that the finances should be structured so that funding is available for the IV&V team to complete its analysis or test work. No adverse financial pressure or influence is applied.

4.4.2.3 The IV&V process starts early in the software development life-cycle, providing feedback to the Provider organization, allowing them to modify products at optimal timeframes and in a timely fashion, thereby reducing overall project risk. The feedback also answers project stakeholders' questions about system properties (correctness, robustness, safety, security, etc.) so they can make informed decisions with respect to the development and acceptance of the system and its software.

4.4.2.4 The IV&V Provider performs two primary activities, often concurrently: verification and validation. Each of the activities provides a different perspective on the system/software. Verification is the process of evaluating a system and its software to provide objective evidence as to whether or not a product conforms to the build-to requirements and design specifications. Verification holds from the requirements through the design and code and into testing. Verification seeks to demonstrate that the products of a given development phase satisfy the conditions imposed at the start of or during that phase. Validation tasks, on the other hand, seek to develop objective evidence that shows that the content of the engineering artifact is the right content for the developed system/software. The content is accurate and correct if the objective evidence demonstrates that it satisfies the system requirements (e.g., user needs, stakeholder needs, etc.), that it fully describes the required capability/functionality needed and that it solves the right problem.

4.4.2.5 The center of the IV&V effort is on the discovery of objective evidence that supports the correct operation of the system or refutes the correct operation of the system. To understand this objective evidence, the IV&V provider typically works with the development team, which provides artifacts such as concept studies, operations concepts, and requirements that define the overall project. The IV&V provider uses these materials to develop an independent understanding of the project's commitment to NASA, which then forms the basis for validation of lower-level technical artifacts.

4.4.2.6 Two principles help guide the development and use of objective evidence.

a. Performing IV&V throughout the entire development lifetime is the first principle; potential problems should be detected as early as possible in the development life-cycle. Performing IV&V throughout the entire development lifetime provides the IV&V team with sufficient information with which to establish a basis for the results of the analysis and provides early objective evidence to the development and program management groups to help keep the development effort on track early in the life-cycle.

b. The second principle is "appropriate assurance." Given that it is not possible to provide IV&V on all aspects of a project's software, the IV&V provider and project should balance risks against available resources to define an IV&V program for each project that will provide IV&V so that the software will operate correctly, safely, reliably, and securely throughout its operational lifetime. The IV&V Project Execution Plan will document this tailored approach and summarize the cost/benefit trade-offs made in the scoping process.

4.4.2.7 The requirements in the IV&V section are analyzed and partitioned according to the type of artifact. The requirements do not imply or require the use of any specific life-cycle model. It is also important to understand that IV&V applies to any life-cycle development process. The

requirements in the IV&V section document the potential scope of analysis performed by the IV&V Provider and the key responsibility of the software project to provide the information needed to perform that analysis. Additionally, scoping the IV&V analysis is according to the application of the risk assessment to determine the prioritization of tasks and the level of rigor associated with performing those tasks. The scoping exercise results are captured in the IV&V Project Execution Plan, as documented below.

4.4.3 IV&V Planning/Management/Program Execution.

4.4.3.1 The IV&V Provider shall: (SASS-02)

a. Conduct an initial planning and risk assessment effort to determine the specific system/software behaviors (including the software components responsible for implementing the behaviors) to be analyzed, the IV&V tasks to be performed, the rigor to be applied to the tasks, and any tailoring of the requirements in this standard to be applied to the IV&V effort.

Note: IV&V is a focused activity that prioritizes IV&V analysis to address the highest developmental and operational software risks. IV&V priority is based on the combination of the potential for software impacts to safety and mission success and the probability factors for latent defects. IV&V analysis tasks provide coverage with a degree of rigor that reflects the priority level. The initial planning and scoping effort based on the risk assessment define the starting point for the IV&V analysis. The planning and scoping effort also aid in establishing the initial relationships between the IV&V Provider, the Acquirer, and the Provider.

b. Develop and negotiate with the project an IV&V Execution Plan documenting the activities, methods, level of rigor, environments, tailoring (if any) of these requirements, and criteria to be used in performing verification and validation of in-scope system/software behaviors (including responsible software components) determined by the planning and scoping effort.

Note: A provider should use a documented analysis approach to track and manage the IV&V effort in alignment with on-going development project efforts. The IV&V Project Execution Plan documents which software products will be subject to which analyses, and which analysis requirements will be fully, partially, or not applied following the risk assessment and resource constraints. The IV&V plan also serves as a communication tool between the project and IV&V to set expectations for the IV&V products produced throughout the life-cycle.

c. Provide analysis results, risks, and assurance statements/data to all the responsible organizations' project management, engineering, and assurance personnel.

Note: While independent, the IV&V Provider is still a part of the overall safety and risk mitigation software assurance strategy for a project. The results of IV&V analysis need to be incorporated into the overall software assurance assessment of the project as well as provided to the project management.

d. Participate in project reviews of software activities by providing status and results of software IV&V activities including, but not limited to, upcoming analysis tasks, artifacts needed from the project, the results of the current or completed analysis, defects, and risks to stakeholders, customers, and development project personnel.

Note: The most significant positive impact of IV&V analysis is when the analysis results are in phase with the development effort. Communicating defects after development artifacts are baselined increases the cost to make the changes. Additionally, the inclusion of the IV&V Provider in ongoing technical meetings keeps the IV&V Provider informed of possible changes that may affect future IV&V tasking. Supporting the ongoing technical meetings allows the IV&V Provider an opportunity to provide real-time feedback on these changes.

e. Participate in planned software peer reviews or software inspections and record peer review measurements guided by the planning and scoping risk analysis documented in the IV&V Project Execution Plan as well as by the content of the items being reviewed or inspected.

Note: The IV&V Provider should be involved in the review/inspection process for all system/software items within the scope of their analysis.

f. Establish, record, maintain, report, and utilize software management and technical measurements.

g. Assess and track the actual results and performance of software activities against the software plans and identify and report any risks or findings.

h. Track and evaluate changes to software products to evaluate for possible changes in the IV&V Provider's risk analysis as well as for potential adverse impacts to the software system and the development effort.

i. Assess the software development lifecycle for suitability for the problem to be solved, and identify and communicate any risks associated with the chosen life cycle.

j. Identify, analyze, track, communicate, and record risks to the software and development project in accordance with NPR 8000.4, Agency Risk Management Procedural Requirements.

k. Track, record, and communicate defects/issues and other results found during the execution of IV&V analysis during the software development effort to include issues and results found during the conducting of independent IV&V testing.

4.4.3.2 IV&V Work during Concept Development. The IV&V Provider shall verify and validate that the concept documentation represents the delineation of a specific implementation solution to solve the Acquirer's problem. (SASS-03)

Note: The objective of Concept IV&V is to understand the selected solution and to validate the role of software as it relates to providing the capability(ies) that support high priority or high-risk system capability(ies).

a. Ensure that software planned for reuse meets the fit, form, and function as a component within the new application.

b. Ensure that the system architecture contains the computing-related items (subsystems, components, etc.) to carry out the mission of the system and satisfy user needs and operational scenarios or use cases.

c. Ensure that a basis for the engineering and planning of computing-related functions is the operations, mission objectives (including mission retirement), and the system.

d. Ensure that feasibility and trade studies provide the results to confidently support the critical decisions that drove the need for the study.

e. Identify and document the known software-based hazard causes, hazard contributors, and hazard controls.

f. Identify and document software-based security threats and risks, and the project implements the relevant regulatory requirements.

4.4.3.3 IV&V Work during Requirements Development. The IV&V Provider shall verify and validate: (SASS-04)

a. That the project implements the requirements for software listed in NPR 7150.2 (SWE-134), and risk-driven assessments determine the types of IV&V analyses.

b. That the in-scope software requirements and system requirements are, at a minimum, correct, consistent, complete, accurate, readable, traceable, and testable.

Note: Software usually provides the interface between the user and the system hardware as well as the interface between system hardware components and other systems. These interfaces are critical to the successful operation and use of the system.

c. That the project-identified mitigations for identified security risks are in the software requirements.

Note: Security is an essential aspect of any system development effort. In most systems, software provides the primary user interface. The user interface is an element of the system that can provide undesired access. A system concept design should address known security risks through various features in the system.

d. That the project-identified mitigations for identified security risks are correct and testable.

4.4.3.4 IV&V Work during Design. The IV&V Provider shall verify and validate: (SASS-05)

a. That the relationship between the in-scope system/software requirements and the associated architectural elements is traceable correct, consistent, and complete.

Note: Architectural elements are responsible for implementing specific behaviors within the software and the overall system. It is the interactions between these architectural elements that result in the realization of the desired behaviors as well as possible undesired behaviors.

b. That the software architecture meets the user's safety and mission-critical needs as defined in the requirements.

Note: The architecture provides the foundation for the development of the software. It also significantly impacts how the software deals with faults and failures, as well as how the software interfaces with the user and system components. Analysis of the architecture provides early insight into how the software is structured and how that structure can implement the requirements.

c. That the detailed design products are traceable, consistent, complete, accurate, and testable.

Note: Detailed design is the implementation of the algorithms that will control and monitor the different parts of the system as well as allow for interaction between the system and the user and other systems. The detailed design defines how the architectural components will behave to support the interactions defined in the architecture. Analysis of the detailed design includes looking at the low-level software components in the software system.

d. That the interfaces between the detailed design components and the hardware, users, operators, other software, and external systems are correct, consistent, complete, accurate, and testable.

Note: While the architecture defines the interactions between the architectural elements, each element is generally composed of lower-level components defined by the detailed design. The interfaces between these components are important in ensuring that the architectural element meets its assigned responsibilities.

e. That the relationship between the software requirements and the associated detailed design components is correct, consistent, and complete.

Note: The detailed design components capture the approach to implementing the software requirements, including the requirements associated with fault management, security, and safety. Analysis of the relationship between the detailed design and the software requirements provides evidence that all of the requirements are in the detailed design.

4.4.3.5 IV&V Work during Implementation. The IV&V Provider shall verify and validate: (SASS-06)

a. That the software code products are consistent with architecture, complete with respect to requirements, and testable.

b. That the software code meets the project software coding standard.

c. That the security risks in the software code are identified and mitigated.

d. Analysis to assess the source code for the presence of open-source software including ensuring that the project has identified all open-source software used, the use of the open-source software is consistent with any licensing requirements and that the security risks are identified and mitigated by the use of the open-source software.

e. That software problem reports generated by the IV&V provider have been addressed entirely by the project.

f. That the project assesses the software systems for possible security vulnerabilities and weaknesses.

g. That the project implements the required software security risk mitigations to ensure that security objectives for software are satisfied.

h. The source code through the use of analysis tools (including but not limited to static, dynamic, and formal analysis tools).

Note: The use of analysis tools may include the verification and validation of the results of the analysis tools used by the development project in the process of developing the software.

i. That the relationship between the software design elements and the associated software units is correct, consistent, and complete.

j. That the relationship between software code components and corresponding requirements is correct, complete, and consistent.

Note: For all of the implementation requirements, it is with code that the development of software reaches its lowest level of abstraction and that the software capabilities are implemented. Evaluating the relationship between the source code and the design components and requirements provides evidence that only the specified requirements and components are in the system. Evaluating the relationship between the source code and the design components and requirements helps to minimize one aspect of the emergence of unexpected behaviors: inclusion of behaviors not specified in the requirements. The overall analysis of the code is essential in assuring that the code does implement the required software behaviors. From a safety perspective, it is important to evaluate the code and assure that known software safety and security issues such as buffer overflows and type mismatches, among many others, are not used in safety-critical aspects of the software.

4.4.3.6 IV&V Work during Testing. The IV&V Provider shall: (SASS-07)

a. Verify and validate that test plans, test procedures, test cases, test environment (including simulations), and test design at all levels of testing (unit, integration, system, acceptance, etc.) are correct, complete, and consistent for verification and validation of the source code and system functions allocated to the software.

b. Verify and validate that the relationships between the test plans, test procedures, test cases, and test design and source code and system functions allocated to the software are correct, complete, and consistent.

c. Verify that the test plans, test cases, test design, and test procedures contain objective acceptance criteria that support the verification of the associated requirements for both nominal and off-nominal conditions.

d. Verify that the software test results meet the associated acceptance criteria to ensure that software correctly implements the associated requirements.

Note: The IV&V Provider assesses the testing artifacts within the context of the system's meaning concerning the desired capabilities and expected operational system environment. The assessment includes an examination of testing at system boundary conditions to include unexpected conditions. The testing analysis assures that the project tests all of the requirements and that the system does what the requirements state it should do. The testing analysis also includes analysis of the traceability information between the tests and the requirements.

e. Verify that the project tests the required software security risk mitigations to ensure that the security objectives for the software are satisfied.

4.4.3.7 IV&V Work during Operations/Maintenance. The IV&V Provider shall assess the software maintenance plan concerning software elements to support the planning of IV&V activities during the maintenance phase. (SASS-08)

Note 1: The approach to software development on some projects results in different parts of the software going into operation at different times in the overall project life-cycle. For example, a lander mission to Mars may complete the software needed for the cruise phase to Mars while continuing to work on the entry, descent, landing, and surface operations software. Some software may also have an extended lifetime such that operational updates are anytime during the operational use of the software.

Note 2: In some cases, software anomalies will cause changes to the software. The use of IV&V is important in that changes to software can often have ripple effects throughout the system as well as cause emergent behaviors. The IV&V analysis provides insight into these possible effects as well as providing an overall assessment of the impact of the change.

4.5 Principles Related to Tailoring the Standard Requirements

4.5.1 Software requirements tailoring is the process used to seek relief from standard 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 assurance, software safety, and IV&V efforts may be tailored where justified and approved. NASA established the TA governance model to approve and mitigate any changes to the application of the requirements in the Software Assurance and Software Safety Standard. Tailoring from requirements in the Software Assurance and Software Safety Standard is governed by the following requirements. Tailoring at the Center level is to be decided by the SMA TA and the NPR 7150.2 requirements mapping matrix applicability. Tailor the software assurance, software safety, and IV&V requirements using the following levels:

a. The first level of tailoring is the Software Classification Decision, see NPR 7150.2.

b. The second level of tailoring is the tailoring in the project's Software Requirements Mapping Matrix, see NPR 7150.2.

c. The third level of tailoring is the tailoring by the Software Assurance TA of the Software Assurance and Software Safety Standard requirements that correspond to the project's Software Requirements Mapping Matrix requirements.

4.5.2 The Software Assurance and Software Safety Standard establishes a baseline set of requirements for software assurance, software safety, and IV&V to reduce risks on NASA projects and programs. Each project has unique circumstances, and tailoring can be employed to modify the requirements set for software assurance, software safety, and IV&V effort. Determining the tailoring of requirements is a joint software engineering effort and SMA 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.

4.5.3 Per SWE-121, the software assurance organization maintains a requirements mapping matrix or multiple requirements mapping matrices against requirements in the Software Assurance and Software Safety Standard, including those delegated to other parties or accomplished by contract vehicles. Per SWE-013 and SWE-039, the software assurance organization conducts risk assessment efforts to determine the software assurance, software safety, and IV&V tasks to be performed, and the rigor of each task. The software assurance organization will develop, maintain, and execute plans and procedures that cover the entire software life-cycle and, as a minimum, address the requirements of the Software Assurance and Software Safety Standard with approved tailoring.

4.5.4 The approval of the Software Assurance and Software Safety Standard tailoring is determined by the SMA management at the Center Level in conjunction with the project. The request for relief from a requirement includes the rationale, a risk evaluation, and reference to all material that justifies supporting acceptance. The organization submitting the tailoring request informs the next higher level of involved management of the tailoring request in a timely manner. The dispositioning organization reviews the request with the other organizations that could be impacted or have a potential risk (i.e., to safety, quality, cybersecurity, health) with the proposed changes; and obtains the concurrence of those organizations.

4.5.5 The Center SMA TA shall review and agree with any tailored Software Assurance and Software Safety Standard requirements. (SASS-09)

4.5.6 If a system or subsystem development evolves to meet a higher or lower software classification as defined in NPR 7150.2 then the software assurance, software safety, and IV&V organizations shall update their plan(s) to fulfill the applicable requirements per the Requirements Mapping Matrix and any approved changes, and initiate adjustments to applicable contracts to meet the modified requirements. (SASS-10)

4.5.7 The responsibilities for approving changes to the software engineering requirements are in the NPR 7150.2. When the requirement and software class are applicable, the projects will record the risk and rationale for any requirement that is not completely implemented by the project. The projects can document their related mitigations and risk acceptance in the approved Requirements Mapping Matrix.

APPENDIX A. GUIDELINES FOR THE HAZARD DEVELOPMENT INVOLVING SOFTWARE

A.1 Software Contributions to Hazards

A.1.1 Hazard Analysis should consider software's ability, by design, to cause or control a given hazard. It is a best practice to include the software within the system hazard analysis. The general hazard analysis should consider software common-mode failures that can occur in instances of redundant flight computers running the same software.

A.1.2 Software Safety Analysis supplements the system hazard analysis by assessing the software performing critical functions serving as a hazard cause or control. The review assures compliance with the levied functional software requirements, including SWE-134. The software should not violate the independence of hazard inhibits, and the software should not violate the independence of hardware redundancy. The Software Safety Analysis should follow the phased hazard analysis process. A typical Software Safety Analysis process begins by identifying the must work and must not work functions in Phase 1 hazard reports. The system hazard analysis and software safety analysis process should assess each function, between Phase 1 and 2 hazard analysis, for compliance with the levied functional software requirements, including SWE-134. For example, Solar Array deployment (must work function) software should place deployment effectors in the powered off state when it boots up and requires arm and fire commands in the correct order within 4 CPU cycles before removing a deployment inhibit. The analysis also assesses the channelization of the communication paths between the inputs/sensors and the effectors to assure there is no violation of fault tolerance by routing a redundant communication path through a single component. The system hazard analysis and software safety analysis also assures the redundancy management performed by the software supports fault tolerance requirements. For example, software can't trigger a critical sequence in a single fault-tolerant manner using a single sensor input. Considering how software can trigger a critical sequence is required for the design of triggering events such as payload separation, tripping FDIR responses that turn off critical subsystems, failover to redundant components, and providing closed-loop control of critical functions such as propellant tank pressurization.

A.1.3 The design analysis portion of software safety analysis should be completed by Phase 2 safety reviews. At this point, the software safety analysis supports a requirements gap analysis to identify any gaps (SWE-184) and ensuring the risk and control strategy documented in hazard reports are correct, as stated. Between Phase 2 and 3 safety reviews, the system hazard analysis and software safety analysis supports the analysis of test plans to assure adequate off-nominal scenarios (SWE-62, SWE-65a). Finally, in Phase 3, the system hazards analysis should verify the final implementation and verification upholds the analysis by ensuring test results permit closure of hazard verifications (SWE-68) and that the final hazardous commands support the single command and multi-step command needs and finalized pre-requisite checks are in place.

A.1.4 The following section includes useful considerations and examples of software causes and controls:

a. Does software control any of the safety-critical hardware?

b. Does software perform critical reconfiguration of the system during the mission?

c. Does the software perform redundancy management for safety-critical hardware?

d. Does the software determine when to perform a critical action?

e. Does the software trigger logic to meet failure tolerance requirements?

f. Does the software monitor hazard inhibits, safety-critical hardware/software, or issue a caution and warning alarm used to perform an operational control?

g. Does the software process or display data used to make safety-critical decisions?

h. Does the flight or ground software manipulate hazardous system effectors during prelaunch checkouts or terminal count?

i. Does the software perform analysis that impacts automatic or manual hazardous operations?

j. Does the software serve as an interlock preventing unsafe actions?

k. Does the software contain stored command sequences that remove multiple inhibits from a hazard?

l. Does the software initiate any stored command sequences, associated with a safety-critical activity?

m. Does software violate any hazard inhibits or hardware redundancy independence (channelized communication/power paths, stored command sequences/scripts, FDIR false positive, etc.)?

n. Can the software controls introduce new hazard causes?

o. Are the software safety-critical controls truly independent?

p. Can common cause faults affect the software controls?

q. Can any of the software controls used in operational scenarios cause a system hazard?

r. Does the software control switch-over to a backup system if a failure occurs in a primary system?

s. Is the software that makes safety-critical decisions fault-tolerant?

t. Does the software provide an approach for recovery if the system monitoring functions fail?

u. Does the software allow the operators to disable safety-critical controls unintentionally?

v. Does the software provide safety-critical cautions and warnings?

w. Is the software capable of diagnosing and fixing safety-critical faults that might occur in operations?

x. Does the software provide health and status of safety-critical functions?

y. Does the software process safety-critical commands (including autonomous commanding)?

z. Can software that provides full or partial verification or validation of safety-critical systems generate a hazard if it has a defect, fault, or error?

aa. Can a defect, fault, or error in the software used to process data or analyze trends lead to safety decisions that cause a system hazard?

bb. Do software capabilities exist against the potential use cases and planned operations throughout all phases of use, and through transitions between those phases/states?

A.1.5 Additional considerations when identifying software causes in a general software-centric hazard analysis are found in Table 2 below.

Table 2. Additional considerations when identifying software causes in hazard analysis

Additional considerations	Causes	Controls
Computer Reset	1. Reset with no restart 2. Reset during program upload (PROM corruption) 3. Boot PROM corruption preventing reset 4. Watchdog active during reboot causing infinite boot loop	1. Disable FDIR and watchdogs during boot 2. Redundant computers
Memory	1. Memory corruption 2. Out of Memory 3. Buffer overrun 4. Deadlock (trying to write to same memory at the same time or trying to update while reading it) 5. Single Event Upset/bit flip 6. Shared Memory (can defeat redundancy independence)	1. No Dynamic allocation of memory after initialization 2. EDAC/Memory scrubbing 3. Memory margin
CPU	1. Cycle overrun 2. Cycle jitter 3. Stack processing/ordering incorrect (e.g., FIFO vs. LIFO vs. unknown) 4. One task failing to release shared resource 5. Single Event Upset	1. CPU utilization and cycle time margin 2. Self-checking CPU cores

Additional considerations	Causes	Controls
	6. Hardware failure	
Commanding	1. Inadvertent/Erroneous commanding 2. Failure to command 3. Command out of sequence 4. Command corrupted or illegitimate command=	1. Command validation (cyclic redundancy check, timestamp, destination/source) 2. Acknowledgment/ negative acknowledgment
Communication and Input/Output	1. Communication bus overload 2. Lack of or intermittent communication 3. Complex configuration input 4. Babbling Node	1. Communication bus utilization margin 2. Use of lossless protocol 3. Downstream input voting 4. External safing function 5. Bus Guardian
Display Data	1. Incorrect Data (unit conversion, incorrect variable type, etc.) 2. Stale Data	1. Visual indication of stale data 2. Watchdog timer
Events and Actions	1. Out-of-sequence event protection 2. Multiple events/actions trigger simultaneously (when not expected) 3. Error/Exception handling missing or incomplete 4. Inadvertent mode transition	1. Fault Management 2. Pre-requisite logic 3. Interlocks
Timekeeping	1. Time runs fast/slow 2. Time skips (e.g., Global Positioning System time correction) 3. Time sync across components 4. Oscillator Drift	1. Diverse/redundant time sources with fault down logic 2. Robust time sync design that can deal with the loss of external time sources 3. Prelaunch checkout of Oscillators
Timing Problems	1. Priority inversion 2. Failure to terminate/complete process in a given time 3. Data latency/sampling rate too slow 4. Race Conditions 5. Non-determinism	1. Static and Dynamic Analysis Tools 2. Coding standards
Coding, Logic, and Algorithm failures	1. Division by zero 2. Bad data in = bad data out (no parameter range & boundary checking) 3. Dead code 4. Unused code 5. Non-functional loops	1. Use of industry-accepted coding standard 2. Use of safe math libraries

Additional considerations	Causes	Controls
	6. Endless do loops 7. Incorrect passes (too many or too few or not at the correct time) 8. Incorrect "if-then" and incorrect "else." 9. Too many or too few parameters for the called function 10. Case/type mismatch 11. Precision mismatch 12. Rounding or truncation fault 13. Resource contention (e.g., thrashing: two or more processes accessing a shared resource) 14. Bad configuration data/no checks on external input files and data 15. Inappropriate equation 16. Undefined or non-initialized data 17. Limit ranges 18. Relationship logic for interdependent limits 19. Overflow or underflow in the calculation	3. Robust software development, quality, and safety processes
Input Failures	1. Noise 2. Sensors or actuators stuck at some value (all zeros, all ones, some other value) 3. A value above or below range 4. Value in range but incorrect 5. Physical units incorrect 6. Inadequate data sampling rate	1. Sensor Health Checks 2. Input Validation, Sanity Checks
User Interface Errors	1. Wrong commands are given by the operator 2. No commands are given by the operator 3. Status and messages not provided for operations, systems, and inhibits 4. Ambiguous or incorrect messages 5. User display locks up/fails	1. Two-step commands 2. GUI style guide 3. Software interlocks to prevent human error
Configuration Management	1. Incorrect version loaded 2. Incorrect configuration values	1. Version CRC check after software/configuration load
Security	1. Denial/Interruption of Service 2. Spoofed/Jammed inputs 3. An unauthorized input/access	1. Message filtering to detect spoofing 2. Ensure software has data source validation checking features