Measurement System Identification:
Not Measurement Sensitive

NASA TECHNICAL STANDARD

National Aeronautics and Space Administration

NASA-STD-8739.8B

Approved: 2022-09-08
Superseding NASA-STD-8739.8A

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 to make minor editorial changes. Note: Some paragraphs have changed pages as a result of these changes. Change indications identify only pages where content has changed.
	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 the NASA-STD-8719.13 standard.
	B	2022-09-08	Brings into alignment with the update to NPR 7150.2D. Update the Appendix A table containing the additional considerations to consider when identifying software causes in hazard analysis.

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.

TABLE OF CONTENTS

DOCUMENT HISTORY LOG 2
FOREWORD 3
TABLE OF CONTENTS 4
LIST OF APPENDICES 4
LIST OF TABLES 4

1. SCOPE 5
1.1 Document Purpose 5
1.2 Applicability 6
1.3 Documentation and Deliverables 6
1.4 Request for Relief 6

2. APPLICABLE AND REFERENCE DOCUMENTS 6
2.1 Applicable Documents 6
2.2 Reference Documents 7
2.3 Order of Precedence 8

3. ACRONYMS AND DEFINITIONS 9
3.1 Acronyms and Abbreviations 9
3.2 Definitions 9

4. SOFTWARE ASSURANCE AND SOFTWARE SAFETY REQUIREMENTS ... 15
4.1 Software Assurance Description 15
4.2 Safety-Critical Software Determination 15
4.3 Software Assurance and Software Safety Requirements 16
4.4 Independent Verification &Validation (IV&V) 48
4.5 Principles Related to Tailoring the Standard Requirements 56

LIST OF APPENDICES

APPENDIX A. Appendix A Guidelines for the Hazard Development involving Software 58

LIST OF TABLES

Table 1. Software Assurance and Software Safety Requirements Mapping Matrix 17
Table 2 Additional considerations to consider when identifying software causes in hazard analysis 59

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 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 and 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 specified requirements and standards that 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 Safety Standard does not impose a particular life-cycle model on each software project; it supports 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" denote 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 keep records, reports, metrics, analyses, and trending results and 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 Program or Project 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 developing 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 is 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 documents 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-1006	Space System Protection Standard
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 are 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 Project 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.

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 metrics analysis. 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 are 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 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 ensuring activities are done completely and correctly and have expected content according to 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 mature or advance 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. Identifying and evaluating 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 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) and 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 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. The project manager is usually NASA but can also refer to the Prime contractor-subcontractor relationship as well.

Provider. A Provider is a NASA or contractor organization that is tasked by an accountable organization (i.e., the Acquirer) to produce a product or service. (Source NASA-HDBK-8709.22)

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. Confirm that the product fulfills its intended use as provided (or as it will be provided). 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. Any organization, which provides a product or service to a customer. By this definition, suppliers may include vendors, subcontractors, contractors, flight programs/projects, and the NASA organization supplying science data to a principal investigator. (In contrast, the classical definition of a supplier is: a subcontractor, at any tier, performing contract services or producing the contract articles for a contractor.). (Source NASA-HDBK-8709.22)

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 specified requirements and standards that 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 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, including security 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, including security plans, are in place and have expected content for the life-cycle events, with proper tailoring for the classification of the software. 2. Develop and maintain 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. Changes to commitments (e.g., software plans) that have been agreed to by the affected groups	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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		and individuals are taken, recorded, and managed.	
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.
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 are established and maintained upon receipt of deliverables (e.g., approvals, reviews).
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.	1. Confirm that software developer(s) periodically report status and provide insight to the project manager. 2. Monitor product integration. 3. Analyze the verification activities to ensure adequacy.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		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.	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. 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 provide 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 project's software requirements, products, procedures, and processes are compliant with the NPR 7150.2 requirements per the software classification and safety criticality for software.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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 delegated to other parties or accomplished by contract vehicles or Space Act Agreements.	1. Confirm that the project maintains a requirements mapping matrix (matrices) for all requirements in NPR 7150.2. 2. Maintain the requirements 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, transfer rights, and conditions of use (e.g., required copyright, author, and applicable license notices within the software code, or a requirement to redistribute the licensed software only under the same license (e.g., GNU GPL, ver.3, license)) have been	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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		addressed and coordinated with Center Intellectual Property Counsel. d. Future support for the software product is planned and adequate for project needs. e. The software component is verified and validated to the same level required to accept a similar developed software component for its intended use. f. The project has a plan to perform periodic assessments of vendor reported defects to ensure the defects do not impact the selected software components.	
3.2		Software Cost Estimation	

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 Class C and Class 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., programmatic assumptions/constraints, assessment of the size, functionality, complexity, criticality, reuse code, modified code, and risk of the software processes and products). c. Is based on the cost implications of the technology to be used and the required maturation of that technology. d. Incorporates risk and uncertainty, including end state risk and threat assessments for cybersecurity. e. Includes the cost of the required software assurance support. f. Includes other direct costs.	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, status, performance metrics, and assessment/analysis 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 including the software assurance's/software safety'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. 2. Confirm that software assurance and software safety personnel have completed the basic software assurance and/or software safety training.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
3.5		Software Classification Assessments	
3.5.1	020	The project manager shall classify each system and subsystem containing software in accordance with the highest applicable software classification definitions for Classes A, B, C, D, E, and F software in Appendix D.	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, software safety and IV&V (if required) per NASA-STD-8739.8, Software Assurance and Software Safety Standard.	1. Perform software assurance, software safety and IV&V (if required) according to the software assurance and software safety standard requirements in NASA-STD-8739.8, Software Assurance and Software Safety Standard and the Project's software assurance plan.
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, Risk Classification for NASA Payloads. c. Projects selected explicitly by the Mission	1. Confirm that IV&V requirements (section 4.4) are complete on projects required to have IV&V.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		Directorate Associate Administrator to have software IV&V.	
3.6.3	131	If software IV&V is required for a project, the project manager, in consultation with NASA IV&V, shall ensure an IPEP is developed, approved, maintained, and executed in accordance with IV&V requirements in NASA-STD-8739.8.	1. Confirm that the IV&V Project Execution Plan (IPEP) exists.
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 responds 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, leads 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

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			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 exists. 5. Develop and maintain a software safety analysis throughout the software development life-cycle.
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	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 that the values of the safety-critical loaded data, uplinked data, rules, and scripts that affect hazardous system behavior have been tested. 4. Analyze the software design to ensure: 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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		inputs and outputs to/from the software system. h. The software performs prerequisite checks prior to the execution of safety-critical software commands. i. No single software event or action is allowed to initiate an identified hazard. j. The software responds to an off-nominal condition within the time needed to prevent a hazardous event. k. The software provides error handling. l. The software can place the system into a safe state.	5. Participate in software reviews affecting safety-critical software products.
3.7.4	219	If a project has safety-critical software, the project manager shall ensure that there is 100% code test coverage using the Modified Condition/Decision Coverage (MC/DC) criterion for all identified safety-critical software components.	1. Confirm that 100% code test coverage is addressed for all identified software safety-critical software components or that software developers provide a risk assessment explaining why the test coverage is not possible for the safety-critical code component.
3.7.5	220	If a project has safety-critical software, the project manager shall ensure all identified safety-critical software components have a cyclomatic complexity value of 15 or lower. Any exceedance shall be reviewed and waived with rationale by the project manager or technical approval authority.	1. 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.
3.8		Automatic Generation of Software Source Code	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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.”
3.8.2	206	The project manager shall require the software developers and custom software suppliers to provide NASA with electronic access to the models, simulations, and associated data used as inputs for auto-generation of software.	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	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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 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 appropriate NASA internal sharing and reuse software systems. However, if the project manager is not a civil servant, then a civil servant will pre-approve all such software contributions; all software contributions should include, at a minimum, the following information: a. Software Title. b. Software Description. c. The Civil Servant Software Technical 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 and any required markings (e.g., required copyright, author, applicable license notices within the software code, and the source of each third-party software component (e.g., software URL & license URL)), if applicable. f. Release notes.	1. Confirm that any project software contributed as a reuse candidate has the identified information in items "a" through "e."
3.11		Software Cybersecurity	

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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.
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 per the requirements contained in the NASA-STD-1006, Space System Protection Standard.	1. For software products with communications capabilities, confirm that the software requirements and software design documentation and software implementation address unauthorized access per the requirements contained in the Space System Protection Standard, NASA-STD-1006.
3.11.5	159	The project manager shall test the software and record test results for the required software cybersecurity mitigation implementations identified from the security vulnerabilities and security weaknesses analysis.	1. Confirm that testing is complete for the cybersecurity mitigation. 2. Assess the quality of the cybersecurity mitigation implementation testing and the test results.
3.11.6	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.
3.11.7	185	The project manager shall verify that the software code meets the project's secure coding standard by using the results from static analysis tool(s).	1. Analyze the engineering data or perform independent static code analysis to verify that the code meets the project's secure coding standard requirements.

<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>3.11.8</td><td>210</td><td>The project manager shall identify software requirements for the collection, reporting, and storage of data relating to the detection of adversarial actions.</td><td>1. Confirm that the software requirements exist for collecting, reporting, and storing data relating to the detection of adversarial actions.</td></tr>
<tr><td>3.12</td><td></td><td>Software Bi-Directional Traceability</td><td></td></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><th>Class F</th></tr>
<tr><td>Higher-level requirements to the software requirements</td><td>X</td><td></td><td>X</td></tr>
<tr><td>Software requirements to the system hazards</td><td>X</td><td>X</td><td></td></tr>
<tr><td>Software requirements to the software design components</td><td>X</td><td></td><td></td></tr>
<tr><td>Software design components to the software code</td><td>X</td><td></td><td></td></tr>
<tr><td>Software requirements to the software verification(s)</td><td>X</td><td>X</td><td>X</td></tr>
<tr><td>Software requirements to the software non-conformances</td><td>X</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>
<tr><td>4.1</td><td></td><td>Software Requirements</td><td></td></tr>
<tr><td>4.1.2</td><td>050</td><td>The project manager shall establish, capture, record, approve, and maintain software requirements, including requirements for COTS,</td><td>1. Confirm that all software requirements are established, captured, and documented as part of the technical specification, including requirements for</td></tr>
</table>

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		GOTS, MOTS, OSS, or reused software components, as part of the technical specification.	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 functions as expected in the customer environment.
4.2		Software Architecture	
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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			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	
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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
4.4.3	061	The project manager shall select, define, and adhere to software coding methods, standards, and criteria.	1. Analyze that the software code conforms to all 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, at a minimum, detect defects, software security, code coverage, and software complexity.	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. Confirm the code coverage data, per SWE-219. 5. Confirm the software complexity data, per SWE-220.
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 tests.
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 the 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-066 and SWE-192, respectively. e. Requirements coverage for cybersecurity per SWE-157 and SWE-210.
4.5.2	065c	The project manager shall establish and maintain: ... c. Software test(s), including any code specifically written to perform test procedures. …	1. Confirm that the project creates and maintains any code specifically written to perform test procedures in a software configuration management system. 2. Confirm that the project records all issues and discrepancies in code specifically written to perform test procedures.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			3. Confirm that the project tracks closure errors and defects found in code specifically written to perform test procedures.
4.5.2	065d	The project manager shall establish and maintain: ... d. 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 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 and defects found during testing.
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. Confirm 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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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 that 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 and verification plan(s) and 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.
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 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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			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 through testing, which trace to a hazardous event, cause, or mitigation techniques.
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 system behavior are baselined in the software configuration system. 3. Confirm that loaded or uplinked data, rules, scripts are verified as correct during operations, particularly for safety-critical operations.
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 maintenance, operations, and retirement plans for completeness of the required software engineering and software assurance activities.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			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 for all deliveries on the configuration management processes to verify that all products are being delivered and are the correct versions.
4.6.4	194	The project manager shall complete, prior to delivery, verification that all software requirements identified for this delivery have been met or dispositioned, that all approved changes have been implemented and that all defects designated for resolution prior to delivery 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 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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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 software retirement or disposal products. 3. Confirm that the project archives all software and records as planned.
5		Supporting Software Life-Cycle Requirements	
5.1		Software Configuration Management	
5.1.2	079	The project manager shall develop a software configuration management plan that describes the functions, responsibilities, and authority for the implementation of software configuration management for the project.	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 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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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. c. Identify the persons or groups to make changes at each level.	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 follows the established procedures.
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 follows defined procedures for deliverable software products.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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 software risks and mitigation plans. 2. Perform audits on the risk management process for the software activities.
5.3		Software Peer Reviews/Inspections	
5.3.2	087	The project manager shall perform and report the results of software peer reviews or software inspections for: a. Software requirements. b. Software plans, including cybersecurity. c. Any design items that the project identified for software peer review or software inspections according to the software development plans. d. Software code as defined in the software and or project plans. e. Software test procedures.	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 with a cyclomatic complexity value over 15.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
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.
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. 3. Collect any identified Center/organizational metrics and submit them to the Center/organizational repository.
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,	1. Confirm access to software measurement data, analysis, and status as requested to the following entities: - Sponsoring Mission Directorate

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
		the NASA Chief Engineer, the Center Technical Authorities, Headquarters SMA, and other organizations as appropriate.	- 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 that the project monitors and updates planned measurements to ensure the software meets or exceeds 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.
5.5		Software Non-conformance or Defect Management	
5.5.1	201	The project manager shall track and maintain software non-conformances (including 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.

NPR 7150.2 Section	SWE #	NPR 7150.2 Requirement	Software Assurance and Software Safety Tasks
			4. Maintain or access 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, and/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 project software's safety, quality, and reliability.
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 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. 3. Assess opportunities for 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

4.4.1 IV&V Overview

4.4.1.1 IV&V is a technical discipline of software assurance that employs rigorous analysis and testing methodologies to identify objective evidence and conclusions to provide an independent assessment of critical products and processes throughout the life-cycle. The evaluation of products and processes throughout the life-cycle demonstrates 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.

4.4.1.2 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. Through technical independence, the IV&V team's different perspective allows it to detect subtle errors overlooked by 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 about 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 independent from the development and program management organization, the IV&V team provides 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.

4.4.1.3 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.) to make informed decisions with respect to the development and acceptance of the system and its software.

4.4.1.4 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 demonstrates that the products of a given development phase satisfy the conditions imposed at the start of or during that phase. Validation develops 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.), fully describes the required capability/functionality needed, and solves the right problem.

4.4.1.5 The center of the IV&V effort is on identifying and generating objective evidence that supports the correct operation of the system or refutes the correct operation of the system. The IV&V Provider typically works with the development team to understand this objective evidence, 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 forms the basis for validating lower-level technical artifacts.

4.4.1.6 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 to establish a basis for the analysis results 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 provides IV&V so that the software will operate correctly, safely, reliably, and securely throughout its operational lifetime. The IV&V Project Execution Plan documents this tailored approach and summarizes the cost/benefit trade-offs made in the scoping process.

4.4.1.7 The IV&V requirements 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 IV&V requirements 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 activities and the level of rigor associated with performing those activities. The scoping exercise results are captured in the IV&V Project Execution Plan, as documented below.

4.4.2 IV&V Requirements

The IV&V requirements are applicable to all IV&V efforts performed on a software development project, as tailored by the IV&V Project Execution Plan. The IV&V requirements also serves as the definition of what NASA considers IV&V. IV&V as 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.

4.4.2.1 The IV&V Provider shall conduct planning and risk assessments to determine the specific system/software behaviors (including the software components responsible for implementing the behaviors) to be analyzed.

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 activities 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. Throughout each IV&V project, continuous and iterative feedback, through the execution of analysis, identification of issues and risks, and the collection of deeper mission understanding, allows IV&V projects to “Follow The Risk” and adjust plans. The planning and scoping effort also aid in establishing the initial relationships between the IV&V Provider, the Acquirer, and the Provider.

4.4.2.2 The IV&V Provider shall develop and negotiate with the project an IV&V IPEP.

Note: The IPEP documents the activities, methods, level of rigor, environments, tailoring (if any) of the IV&V 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. A Provider should use a documented analysis approach to track and manage the IV&V effort aligned with ongoing development project efforts. The IPEP documents which software products are subject to which analyses and analysis requirements are wholly, partially, or not applied following the risk assessment and resource constraints. The IPEP 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. The IPEP may require updating throughout the life cycle.

4.4.2.3 The NASA SMA Technical Authority (TA) shall review and concur with the IPEP.

4.4.2.4 The IV&V Provider shall provide analysis results, risks, and assurance statements and data to the responsible organizations’ project management, engineering, and software assurance personnel.

Note: While independent, the IV&V Provider is still a part of a project's overall safety and risk mitigation software assurance strategy. The results of IV&V analysis need to be incorporated into the overall software assurance assessment of the project and provided to the project management. . IV&V Provider should support project milestone reviews and provide the project with an evaluation of the lifecycle review artifacts to assist development management decisions on whether the review criteria have been met and how to proceed going forward.

4.4.2.5 The IV&V Provider shall participate in project reviews of software activities. Participation includes providing status and results of software IV&V activities including, but not limited to, upcoming analysis activities, 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.

4.4.2.6 The IV&V Provider shall provide the responsible organizations' project management, engineering, and software assurance personnel shall insight into the software IV&V and IV&V test activities. At a minimum, the IV&V Provider will be required to allow the responsible organizations' project management, engineering, and software assurance personnel to:

a. Monitor the IV&V activities and plans.

b. Review the verification activities to ensure adequacy.

c. Review IV&V studies and source data.

d. Audit the software IV&V processes and practices.

e. Participate in IV&V software reviews and technical interchange meetings

4.4.2.7 The IV&V Provider shall participate in planned software peer reviews or software inspections guided by the planning and scoping risk analysis documented in the IV&V Project Execution Plan and NASA-HDBK-2203.

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

4.4.2.8 The IV&V Provider shall establish, record, maintain, report, and utilize IV&V management and technical measurements.

Note: The IV&V Provider gathers and analyzes metrics on a periodic basis to perform continuous improvement of IV&V processes and identify indicators of IV&V and project risks.

4.4.2.9 The IV&V Provider shall assess and track software activities' actual results and performance against the software plans and identify and report any risks or findings to the responsible organizations' project management, engineering, and software assurance personnel.

4.4.2.10 The IV&V Provider shall track and evaluate changes to software products to evaluate for possible changes in the IV&V Provider’s risk analysis and potential adverse impacts to the software system and the development effort.

4.4.2.11 The IV&V Provider shall 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 to the responsible organizations’ project management, engineering, and software assurance personnel.

4.4.2.12 The IV&V Provider shall identify, analyze, track, communicate, and record risks to the software and development project in accordance with NPR 8000.4, Agency Risk Management Procedural Requirements to the responsible organizations’ project management, engineering, and software assurance personnel.

4.4.2.13 The IV&V Provider shall verify the project implements the requirements for software listed in NPR 7150.2 and communicate any risks to the responsible organizations’ project management, engineering, and software assurance personnel.

4.4.2.14 The IV&V Provider shall track, record, and communicate defects/issues and other results found during the execution of IV&V analysis and independent IV&V testing to the responsible organizations’ project management, engineering, and software assurance personnel.

4.4.2.15 The IV&V Provider shall ensure that the identified defects and issues are addressed by the project.

4.4.2.16 The IV&V Provider shall ensure that software planned for reuse meets the fit, form, and function as a component within the new application.

4.4.2.17 The IV&V Provider shall ensure that the system architecture contains the computing-related items (subsystems, components, etc.) to carry out the system's mission and satisfy user needs and operational scenarios or use cases.

4.4.2.18 The IV&V Provider shall ensure that a basis for the engineering and planning of computing-related functions is the operations, mission objectives (including mission retirement), and the system.

4.4.2.19 The IV&V Provider shall ensure that feasibility and trade studies provide the results to confidently support the critical decisions that drove the need for the study.

4.4.2.20 The IV&V Provider shall ensure that known software-based hazard causes, contributors, and controls are identified, documented, and trace to the project requirements.

4.4.2.21 The IV&V Provider shall ensure that known security threats and risks are identified, appropriately documented, and updated throughout the mission lifecycle and communicated to the responsible organizations’ project management, engineering, and software assurance personnel.

4.4.2.22 The IV&V Provider shall verify and validate 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 and the interface between system hardware components and other systems. These interfaces are critical to the successful operation and use of the system.

4.4.2.23 The IV&V Provider shall verify and validate that the 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.

4.4.2.24 The IV&V Provider shall ensure that software requirements meet the dependability and fault tolerance required by the system.

4.4.2.25 The IV&V Provider shall ensure that software requirements provide the capability of controlling identified hazards and do not create hazardous conditions.

4.4.2.26 The IV&V Provider shall verify and validate 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. The interactions between these architectural elements result in the realization of the desired behaviors as well as possible undesired behaviors.

4.4.2.27 The IV&V Provider shall verify and validate 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 and 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.

4.4.2.28 The IV&V Provider shall verify and validate that the detailed design products are traceable, consistent, complete, accurate, and testable.

Note: Detailed design is the implementation of the algorithms that control and monitor the different parts of the system and allow for interaction between the system and the user and other systems. The detailed design defines how the architectural components 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.

4.4.2.29 The IV&V Provider shall verify and validate 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.

4.4.2.30 The IV&V Provider shall verify and validate 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 requirements are in the detailed design.

4.4.2.31 The IV&V Provider shall verify and validate that the software code products are consistent with architecture, complete with respect to requirements, and testable.

4.4.2.32 The IV&V Provider shall verify and validate that the software code meets the industry best practices and software coding standards.

4.4.2.33 The IV&V Provider shall verify and validate that the security risks in the software code are identified and mitigated.

Note: This includes 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.

4.4.2.34 The IV&V Provider shall verify and validate appropriate use of off-the-shelf software, including ensuring that the project has identified all open-source software used, and that the security risks are identified and mitigated by the use of the open-source software.

4.4.2.35 The IV&V Provider shall verify and validate that the project assesses the software systems for possible security vulnerabilities and weaknesses.

4.4.2.36 The IV&V Provider shall verify and validate the project implements the required software security risk mitigations to ensure that security objectives for software are satisfied.

4.4.2.37 The IV&V Provider shall verify and validate the source code through the use of analysis tools (including but not limited to static, dynamic, composition, and formal analysis tools).

Note: The use of analysis tools may include the verification and validation of the analysis tools used by the development project in the process of developing the software. The results may be from static code analysis, software composition analysis, dynamic code analysis, cyclomatic complexity, or other code quality analysis tools.

4.4.2.38 The IV&V Provider shall verify and validate the relationship between the software design elements and the associated software units is correct, consistent, and complete.

4.4.2.39 The IV&V Provider shall verify and validate 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 minimize one aspect of the emergence of unexpected behaviors: the 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.

4.4.2.40 The IV&V Provider shall 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.

4.4.2.41 The IV&V Provider shall verify and validate that the relationships between the test plans, test procedures, test cases, test design, and source code and system functions allocated to the software are correct, complete, and consistent.

4.4.2.42 The IV&V Provider shall 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.

4.4.2.43 The IV&V Provider shall verify that the software test results meet the associated acceptance criteria to ensure that the software correctly implements the associated requirements.

Note: The IV&V Provider assesses the testing artifacts within 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 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.

4.4.2.44 The IV&V Provider shall verify that the project tests the required software security risk mitigations to ensure that the security objectives for the software are satisfied.

4.4.2.45 The IV&V Provider shall verify that code coverage is measured by analysis of the results of the execution of tests.

4.4.2.46 The IV&V Provider shall verify through independent testing each of the software requirements that trace to a hazardous event, cause, or mitigation technique.

4.4.2.47 The IV&V Provider shall verify the project's acceptance tests for loaded or uplinked data, rules, and code that affects software and software system behavior.

4.4.2.48 The IV&V Provider shall participate in all NASA quality audits, assessment and reviews associated with the project.

4.4.2.49 The IV&V Provider shall assess the software maintenance and operational risks concerning software elements to support the planning of IV&V activities during the maintenance phase.

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 cause changes to the software. IV&V is important because software changes can often have ripple effects throughout the system and cause emergent behaviors. The IV&V analysis provides insight into these possible effects and provides an overall assessment of the impact of the change.

4.5 Principles Related to Tailoring the Standard Requirements

4.5.1 The SMA TA for the project shall review and concur with any tailored Software Assurance and Software Safety Standard requirements.

4.5.2 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, applying 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 applying the Software Assurance and Software Safety Standard requirements. Tailoring from requirements in the Software Assurance and Software Safety Standard is governed by the following requirements. Tailoring at the Center level is 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 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.

4.5.3 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 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, programs, organizations, or other collections of similar software development efforts.

4.5.4 Per SWE-121, the software assurance organization maintains a requirement 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 develops, maintains, and executes plans and procedures that cover the entire software life-cycle and, as a minimum, address the Software Assurance and Safety Standard requirements with approved tailoring.

4.5.5 The SMA management determines the approval of the Software Assurance and Software Safety Standard tailoring 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.

4.5.6 If a system or subsystem development evolves to meet a higher or lower software classification defined in NPR 7150.2, 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.

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 record the risk and rationale for any requirement 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 identifies 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 assure 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 to design 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 Phase 2 safety reviews should complete the design analysis portion of software safety analysis. The software safety analysis supports a requirements gap analysis to identify gaps (SWE-184) and ensure the risk and control strategy documented in hazard reports are correct, as stated. Between Phase 2 and 3 safety reviews, the system hazards analysis and software safety analysis support test plans' analysis to assure adequate off-nominal scenarios (SWE-62, SWE-65a). Finally, in Phase 3, the system hazards analysis should verify the final implementation and uphold the analysis by ensuring test results permit closure of hazard verifications (SWE-68). The final hazardous commands supported the single command and multi-step command needs and finalized pre-requisite checks are in place.

A.1.4 Considerations when identifying software causes in a general software-centric hazard analysis are found in Table 2 below.

Table 2 Additional considerations to consider when identifying software causes in hazard analysis

Software Cause Areas to Consider	**Potential Software Causes**
Data errors	1. Asynchronous communications 2. Single or double event upset/bit flip or Hardware induced error 3. Communication to/from an unexpected system on the network 4. An out-of-range input value, value above or below range 5. Start-up or hardware initiation data errors 6. Data from an antenna gets corrupted 7. Failure of Software Interface to Memory 8. Failure of flight software to suppress outputs from a failed component 9. Failure of software to monitor Bus Controller Rates to ensure communication with all remote terminals on the bus schedule's avionics buses 10. Ground or Onboard database error, error in tag name database I/O configuration 11. Interface error 12. Latent data (Data delayed or not provided in required time) 13. Communication bus overload 14. Missing or failed integrity checks on inputs, failure to check the validity of input/output data 15. Noise, babbling Node - keeps the node so busy it inhibits communication from other nodes 16. Sensors or actuators stuck at some value (all zeros, all ones, some other value) 17. Wrong software state for the input
Commanding errors	1. Command buffer error or overflow 2. Corrupted software load 3. Error in real-time command build or sequence build 4. Failure to command during hazardous operations 5. Failure to perform prerequisite checks before the execution of safety-critical software commands. 6. Ground or Onboard database error for command structure, 7. Error in Command Data introduced by Command Server error 8. The operator gives wrong commands, 9. Wrong command or a miscalculated command sent, 10. Sequencing error (Failure to issue commands in the correct sequence) 11. Command send in wrong software state or software in an incorrect or unanticipated state 12. An incorrect timestamp on the command 13. Missing software error handling on incorrect commands 14. Status messages on command execution not provided 15. Memory corruption, critical data variables overwritten in memory 16. Inconsistent syntax 17. Inconsistent command options 18. Similarly named commands 19. Inconsistent error handling rules
Flight computer errors	1. Board support package software error 2. Boot load software error 3. Boot PROM corruption preventing reset 4. Buffer overrun – A buffer overflow (or buffer overrun) occurs when the volume of data exceeds the storage capacity of the memory buffer. As a result, the program attempting to write the data to the buffer overwrites adjacent memory locations. 5. CPU overload

Software Cause Areas to Consider	Potential Software Causes
	6. Cycle jitter 7. Cycle over-run 8. Deadlock (trying to write to the same memory at the same time or trying to update while reading it) 9. Reset during program upload (PROM corruption) 10. Reset with no restart 11. Single or double event upset/bit flip or Hardware induced error 12. Time to reset greater than time to failure 13. Unintended persistent data/configuration on reset 14. Watchdog active during reboot causing infinite boot loop 15. Watchdog failure 16. Failure to detect and transition to redundant or backup computer 17. Incorrect or stale data in redundant or backup computer
Operating systems errors	1. Application software incompatibility with upgrades/patches to an operating system 2. Cyclomatic Complexity levels, the complexity of software components (preventing thorough testing and increasing likelihood of coding errors) 3. Defects in Real-Time Operating System (RTOS) Board Support software 4. Defects in the Real-Time Operating System (RTOS) Commercial-Off-The-Shelf (COTS) Software 5. Missing or incorrect software error handling 6. Partitioning errors 7. Shared resource errors 8. Single or double event upset/bit flip 9. Doesn't do what the user expects 10. Excessive functionality 11. Missing function 12. Wrong function 13. Inadequate protection against operating system bugs
Programmable logic device errors	1. Cyclomatic Complexity levels, the complexity of software components (preventing thorough testing and increasing likelihood of coding errors) 2. Errors in programming and simulation tools used for PLC development 3. Errors in the Programmable Logic Device interfaces 4. Errors in the Software/Firmware Logic Design 5. Missing software error handling in the Software/Firmware Logic Design 6. Programmable Logic Controller (PLC) logic/sequence error 7. Single or double event upset/bit flip or hardware induced error 8. Timing errors 9. Doesn't do what the user expects 10. Excessive functionality 11. Missing function 12. Wrong function
Flight system time management errors	1. Incorrect data latency/sampling rates 2. Failure to terminate/complete process in a given time 3. Incorrect time sync 4. Latent data (Data delayed or not provided in required time) MET timing issues and distribution 5. Incorrect function execution, performing a function at the wrong time, out of sequence, or when the program is in the wrong state 6. Race Conditions 7. The software cannot respond to an off-nominal condition within the time needed to prevent a hazardous event. 8. Time function runs fast/slow 9. Time skips (e.g., *Global Positioning System* time correction)

Software Cause Areas to Consider	**Potential Software Causes**
	10. Loss or incorrect time sync across flight system components 11. Loss or incorrect time Synchronization between ground and spacecraft Interfaces 12. Unclear software timing requirements 13. Asynchronous systems or components
Coding, logic, and algorithm failures, algorithm specification errors	1. Auto-Coding errors as a cause 2. Bad configuration data/no checks on external input files and data 3. Division by zero, 4. Wrong sign, 5. Syntax errors, 6. Error coding software algorithm, 7. Error in positioning algorithm, 8. Case/type/conversion error/unit mismatch 9. Buffer overflows 10. High Cyclomatic Complexity levels (above 15), the complexity of software components (preventing thorough testing and increasing likelihood of coding errors) 11. Dead code or used code 12. Endless do loops 13. Erroneous Outputs 14. Failure of Flight Computer Software to Transition to or Operate in a Correct Mode or State 15. Failure to check safety-critical outputs for reasonableness and hazardous values and correct timing. 16. Failure to generate a process error upon detection of arithmetic error (such as divide-by-zero). 17. Failure to open a software problem report when an unexpected event occurs 18. Inadvertent memory modification 19. Incorrect "if-then" and incorrect "else." 20. Missing default case in a switch statement 21. Incorrect implementation of a software change, software defect, or software non-conformance 22. Incorrect passes (too many or too few or not at the correct time) 23. Incorrect software operation if no commands are received or if a loss of commanding capability exists (Inability to issue commands) 24. Insufficient or poor coding reviews, inadequate software peer reviews 25. Insufficient use of coding standards 26. Interface errors 27. Missing or inadequate static analysis checks on code 28. Missing or incorrect parameter range & boundary checking 29. Non-functional loops 30. Overflow or underflow in the calculation 31. Precision mismatch 32. Resource contention (e.g., thrashing: two or more processes accessing a shared resource) 33. Rounding or truncation fault 34. Sequencing error (Failure to issue commands in the correct sequence) 35. Software is initialized to an unknown state; failure to properly initialize all system and local variables are upon startup, including clocks. 36. Too many or too few parameters for the called function 37. Undefined or non-initialized data 38. Unfound defects in the software 39. Untested COTS, MOTS, or reused code 40. Incomplete end-to-end testing

Software Cause Areas to Consider	Potential Software Causes
	41. Incomplete or missing software stress test 42. Errors in the data dictionary or data dictionary processes 43. Confusing feature names 44. More than one name for the same feature 45. Repeated code modules 46. Failure to initialize a loop-control 47. Failure to initialize (or reinitialize) pointers 48. Failure to initialize (or reinitialize) registers 49. Failure to clear a flag
Fault tolerance and fault management errors	1. Missing software error handling 2. Missing or incorrect fault detection logic 3. Missing or incorrect fault recovery logic 4. Problems with the execution of emergency safing operations 5. Failure to halt all hazard functions after an interlock failure 6. The software cannot respond to an off-nominal condition within the time needed to prevent a hazardous event. 7. Common mode software faults o Lack of backup flight software o Lack of dissimilar software o Lack of dissimilar control algorithms 8. A hazard causal factor occurrence isn't detected o Software fails to sense a change in the condition of the hardware o Failure to detect a problem o Failure to detect and react to a system failure (Examples of system failures: Aural Warning System Failure, Communication System Failure, Facility Fire and Gas Detection System Failure, Critical Sensor Failure, Facility Control System Failure, Control System Failure, Common Control System Failure) 9. False positives in Fault Detection Algorithms 10. Failure to perform prerequisite checks before the execution of safety-critical software commands. o Failure to verify operator input commands 11. Failure to terminate/complete process in a given time 12. Memory corruption, critical data variables overwritten in memory 13. Single or double event upset/bit flip or Hardware induced error 14. Incorrect interfaces, errors in interfaces 15. Missing self-test capabilities 16. Failing to consider stress on the hardware 17. Incomplete end-to-end testing 18. Incomplete or missing software stress test 19. Errors in the data dictionary or data dictionary processes 20. Failure to provide or ensure secure access for input data, commanding, and software modifications.
Software processes errors	1. Failure to implement software development processes or implementing inadequate processes 2. Inadequate software assurance support and reviews 3. Missing or inadequate software assurance audits 4. Failure to follow the documented software development processes 5. Missing, tailored, or incomplete implementation of the safety-critical software requirements in NPR 7150.2

Software Cause Areas to Consider	Potential Software Causes
	6. Missing, tailored, or incomplete implementation of the safety-critical software requirements in SSP 50038, Computer-Based Control System Safety Requirements, 7. Incorrect or incomplete HW/SW Lab Testing 8. Inadequate testing of reused or heritage software 9. Failure to open a software problem report when an unexpected event occurs 10. Failure to include hardware personnel in reviews of software changes, software implementation, peer reviews, and software testing 11. Failure to perform a safety review on all software changes and software defects. 12. Defects in Commercial and Modification of Off-The-Shelf (OTS) Software, failure to perform assessments of available bug fixes and updates available in Commercial-Off-The-Shelf (COTS) software (Consider: operating systems, runtime libraries, device drivers packaged with the OS, third-party FPGA libraries) 13. Insufficient use of coding standards 14. Missing or inadequate static analysis checks on code 15. Incorrect version loaded 16. Incorrect configuration values or data 17. No checks on external input files and data 18. Errors in configuration data changes being uploaded to spacecraft 19. Software/avionics simulator/emulator errors and defects 20. Unverified software 21. Unverified COTS software 22. High Cyclomatic Complexity levels (over 15), the complexity of software components (preventing thorough testing and increasing likelihood of coding errors) 23. Incomplete or inadequate software requirements analysis 24. Compound software requirements 25. Incomplete or inadequate software hazard analysis 26. Incomplete or inadequate software safety analysis 27. Incomplete or inadequate software static analyses 28. Incomplete or inadequate software test data analysis 29. Unrecorded software defects found during informal and formal software testing 30. Auto-Coding tool faults and defects 31. Errors in UML design models 32. Software errors in hardware simulators due to lack of understanding of hardware requirements 33. Incomplete or inadequate software test data analysis 34. Inadequate BIT (Built- in-Test) coverage 35. Inadequate regression testing and unit test coverage of flight software application-level source code (especially safety-critical software) 36. Failure to test all nominal and planned contingency scenarios (breakout and re-rendezvous, launch abort) and complete mission duration (launch to docking to splashdown) in the hardware in the loop environment ○ Incomplete testing of unexpected conditions, boundary conditions, and software/interface inputs. 37. Use or persistence of test data, files, or config files in an operational scenario 38. Failure to provide multiple paths or triggers from safe states to hazardous states 39. Interface Control Documents (ICD) and Interface Requirements Documents (IRD) Errors 40. System requirements errors 41. Misunderstanding of hardware configuration and operation

Software Cause Areas to Consider	Potential Software Causes
	42. Hardware requirements and interface errors, Incorrect description of the software/hardware functions and how they are to perform 43. Missing or incorrect software requirements or specifications 44. Missing software error handling 45. Requirements/design errors not fully defined, detected, and corrected) 46. Failure to identify the safety-critical software items 47. Cyclomatic Complexity levels 48. Failure to perform a function, performing the wrong function, performing the function incompletely 49. An inadvertent/unauthorized event, an unexpected, unwanted event, an out-of-sequence event, the failure of a planned event to occur 50. The magnitude or direction of an event is wrong 51. Out-of-sequence event protection 52. Multiple events/actions trigger simultaneously (when not expected) 53. Error/Exception handling missing or incomplete 54. Inadvertent or incorrect mode transition for required vehicle functional operation; undefined or incorrect mode transition criteria; unauthorized mode transition 55. Failure of flight software to correctly initiate proper transition mode 56. Software state transition error 57. Software termination is an unknown state. 58. Errors in the software data dictionary values (typical components of a data dictionary entry are: Channelization data, e.g., bus mapping, vehicle wiring mapping, hardware channelization, description of each I/O variable, formats, unit of measure, and definition, Rate group data, calibrated sensor data describes the format, units of measure, and definition of each sensor, Specify data reduction for transforming raw data into calibrated data, Sensor data qualification criteria and senor data disqualification criteria, Telemetry format/layout and data describe the telemetry mode, format, packetization, and definition, Specify what types of packets are allowed to be sent for each of the telemetry modes, Specify maximum data rate for each mode, Data recorder format/layout and data, Command definition, e.g., onboard, ground, test specific, describes each of the commands processed by the software work package. Effecter command information provides information about the setting of names or flags that cause command executions in the software work products, Operational limits, Scheduling procedures, Partitioning methods, means of preventing partition breaches, partitioning faults, and partitioning metric data.
Human-machine interface errors	1. Incorrect Data (unit conversion, incorrect variable type) 2. Stale Data 3. Poor design of Human Machine Interface (HMI) o Warnings for system failures are not evident on HMI o Timing issues result in warnings not being displayed soon enough to allow for human intervention o Operator data display overload 4. Too much, too little, incorrect data displayed 5. Ambiguous or incorrect messages 6. User display locks up/fails 7. Missing software error handling o Failure to check the validity of input/output data o Failure to check for constraints in algorithms/specifications and valid boundaries 8. Unsolicited command (Command issued inadvertently, cybersecurity issue or without cause) 9. Wrong Command or a Miscalculated Command to be Sent

Software Cause Areas to Consider	Potential Software Causes
	10. Failure of human interface software to check operator inputs 11. Failure to pass along information or messages 12. Display refresh rate leads to an incorrect operator response 13. Lack of ordering scheme for hazardous event queues (such as alerts) in the human-computer interface (i.e., priority versus time of arrival, for example, when an abort must go to the top of the queue) 14. Incorrect labeling of operator controls in the human interface software 15. Failure to check for constraints in algorithms/specifications and valid boundaries 16. Failure of human interface software to check operator inputs 17. Failure to pass along information or messages 18. Stale data 19. Display refresh rate leads to an incorrect operator response 20. Lack of ordering scheme for hazardous event queues (such as alerts) in the human-computer interface (i.e., priority versus time of arrival, for example, when an abort must go to the top of the queue) 21. No onscreen instructions 22. Undocumented features 23. States that appear impossible to exit 24. No cursor 25. Failure to acknowledge an input 26. Failure to advise when a change takes effect 27. Wrong, misleading, or confusing information 28. Poor aesthetics in the screen layout 29. Menu layout errors 30. Dialog box layout errors 31. Obscured instructions 32. Misuse of color
Security and virus errors	1. Denial/Interruption of Service 2. Spoofed/Jammed inputs 3. Missing capabilities to defect insider threat activities 4. Inadvertent or intentional memory modification 5. Inadvertent or unplanned mode transition 6. Missing software error handling or defect handling 7. Unsolicited command (Command issued inadvertently, cybersecurity issue or without cause) 8. Stack-based buffer overflows are common and leverage stack memory that only exists during the execution time. 9. Heap-based attacks are harder to carry out and involve flooding the memory space allocated for a program beyond memory used for current runtime operations 10. Cybersecurity vulnerability or computer virus o Virus infection in console PC 11. Inadvertent access to ground system software 12. Destruct commands incorrectly allowed in a hands-off zone
Unknown/Unknowns errors	1. Undetected software defects 2. Unknown limitations for COTS (operational, environmental, stress) 3. COTS extra capabilities 4. Incomplete or inadequate software safety analysis for COTS components 5. Compiler behavior errors or undefined compiler behavior 6. Software defects and investigations that are unresolved before the flight