NASA TECHNICAL STANDARD

NASA-STD-8719.13C

National Aeronautics and Space Administration
Washington, DC 20546-0001

Approved: 05-07-2013
Superseding NASA-STD-8719.13B

SOFTWARE SAFETY STANDARD

MEASUREMENT SYSTEM IDENTIFICATION:
NOT MEASUREMENT SENSITIVE

DOCUMENT HISTORY LOG

Status	Document Revision	Approval Date	Description
Baseline		02-12-1996	Initial Release as NSS 1740.13.
Revision	A	09-15-1997	Replaced NSS 1740.13. *(White)*
Change	1	09-15-1997	Headquarters documentation numbering *(White)*
Change	2	09-15-1997	Paragraph 3.1(e); access scope and level of IV&V *(White)*
Revision	B		
Change	1	07-28-2004	Correct erroneous paragraph numbering for paragraphs 5, 5.1.1, 5.1.2, 5.1.3, 5.1.4, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 6, 6.1, 6.2, 6.3, 6.4, and 7. Correction of paragraph numbering caused some page numbers to change, the Table of Contents has been revised accordingly. *(WBHIII)*
Revision	C	05-07-2013	Updated requirements based on Agency-wide best practices for software safety and added requirement numbers. *(MW)*

FOREWORD

This 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 programs and projects, including requirements for selection, application, and design criteria of an item. This Standard is approved for use by NASA Headquarters and NASA Centers, including Component Facilities and Technical and Service Support Centers, and is intended to be applied on NASA contracts.

This Standard was developed by the NASA Office of Safety and Mission Assurance (OSMA) to provide the requirements for ensuring software safety across all NASA Centers, programs, projects, and facilities. It describes the activities necessary to ensure that safety is designed into the software that is acquired for or developed by NASA. All program, project, and facility managers, area safety managers, information technology managers, and other responsible managers are to assess the contribution of software to the inherent safety risk of the systems and operations software in their individual programs, projects, and facilities. The magnitude and depth of software safety activities should be commensurate with reflects the risk posed by the software.

Requests for information, corrections, or additions to this Standard should be submitted to the National Aeronautics and Space Administration, Director, Safety and Assurance Requirements Division, Office of Safety and Mission Assurance, Washington, DC 20546 or via "Feedback" in the NASA Standards and Technical Assistance Resource Tool at http://standards.nasa.gov.

May 7th, 2013

Terrence W. Wilcutt

Approval Date

Chief, Safety and Mission Assurance

TABLE OF CONTENTS

SOFTWARE SAFETY STANDARD

1. SCOPE

1.1 Purpose

The purpose of this Standard is to define the requirements to implement a systematic approach to software safety as an integral part of system safety and the overall safety program of a program, project, or facility. This Standard specifies the software activities, data, and documentation necessary for the acquisition and development of software in a safety critical system. These activities may be performed by a collaboration of various personnel in the program, project, or facility, and Safety and Mission Assurance (SMA) organizations. Safety critical systems that include software are evaluated for software's contribution to the safety of the system during the concept phase, and repeated at each major milestone as the design matures.

This Standard describes the activities required to ensure and promote safety processes that are utilized for software that is created, acquired, or maintained by or for NASA. The NASA-GB-8719.13, NASA Software Safety Guidebook, provides additional information on acceptable approaches for implementing software safety. While the requirements of this Standard must be met, the implementation and approach to meeting these requirements will vary to reflect the system to which they are applied.

Software's effect on system safety can be through the commands executed, the data produced, or the effects on resources (e.g., computer memory; file space; bandwidth). Safety could potentially be compromised if software executes a command unexpectedly, executes the wrong command, generates the wrong data, uses unplanned resources, or uses resources incorrectly. Software safety requirements must encompass all these aspects, covering both action (must-work) and inaction (must not work).

There are two kinds of software safety requirements: process and technical. Both need to be addressed and properly documented within a program, project, or facility. This Standard contains process-oriented requirements (what needs to be done to ensure software safety). Technical requirements are those that specify what the system includes or implements (e.g., two-fault tolerance). Use of this Standard does not preclude the necessity to follow applicable technical standards. Some typical technical software safety requirements are provided as examples in Appendix D of this document. NPR 7150.2, NASA Software Engineering Requirements (section 2.2.12, requirement SWE-0134 in Revision A) contains some minimum technical safety requirements.

Software safety requirements do more than prohibit unsafe system behavior. Software is used to command critical, must-work functions. Software can be used proactively to monitor the system, analyze critical data, look for trends, and signal when events occur that may be precursors to a hazardous state. Software can also be used in the control or mitigation of a hazard, event, or condition. Therefore, program, project, and facility software safety requirements include those requirements that will embody these behaviors, both proactive and reactive, and include the system and software states where they are valid.

The requirements specified in this Standard obligate the program, project, and facility, and safety and mission assurance organizations to:

Identify when software plays a part in system safety and generate appropriate requirements to ensure safe operation of the system.

Ensure that software is considered within the context of system safety, and that appropriate measures are taken to create safe software.

Ensure that software safety is addressed in project acquisition, planning, management, and control activities.

Ensure that software safety is considered throughout the system life-cycle, including mission concept, generation of requirements, design, coding, test, maintenance and operation of the software.

Ensure that the acquisition of software, whether off-the-shelf or contracted, includes evaluation, assessment, and planning for addressing and mitigating risks due to the software's contribution to safety and any limitations of the software.

Ensure that software verification and validation activities include software safety verifications and validations.

Ensure that the proper certification requirements are in place and accomplished prior to the actual operational use of the software.

Ensure that changes and reconfigurations of the software, during development, testing, and operational use of the software, are analyzed for their impacts to system safety.

1.2 Applicability

This Standard applies to all software (see definition of software in section 3 of this Standard) that is determined to be safety critical per the Software Safety Litmus Test in Appendix A. Software includes software tools (e.g. simulators, models, emulators, compiler libraries, built in memory checkers, materials analysis, trajectory analysis, and those used for generation and testing of both hardware and software), are assessed for any contributions to hazards to the extent possible. Commercial Off The Shelf (COTS) software is used within some of NASA's projects as well as constituting the majority of the tools NASA uses. See sections 6.5 and 6.6 as well as Appendix F for the approaches to addressing software safety of COTS software and software tools.

This Standard applies to all software within a program, project, and facility life-cycle activities started after the date of issuance. If development started prior to this version's date of issuance, the previous version applies but the program, project, or facility can choose to use this revision. This Standard can be (but not required to be) retroactively applicable to the software within the program's, project's, or facility's development, maintenance, operations, management, acquisition, and assurance activities started prior to the initial issuance of this Standard. This Standard is approved for use by NASA Headquarters and NASA Centers, including Component Facilities and Technical and Service Support Centers, and may be cited in contract, program, and other Agency documents as a technical requirement. This Standard may also apply to the Jet

Propulsion Laboratory or to other contractors, grant recipients, or parties to agreements only to the extent specified or referenced in their contracts, grants, or agreements.

For the purposes of this Standard, any software to be executed on a processor embedded within a programmable logic device (PLD) will be evaluated from a software safety perspective. The design and resulting hardware will be evaluated from a system safety perspective and is not the purview of this standard. The software tools used to generate safety critical PLDs configuration files will be evaluated from a limited COTS safety perspective as per sections 6.5 and 6.6 of this standard. Safety and Mission Assurance of PLDs/CEs is performed per NASA-HDBK-8739.23, NASA Complex Electronics Handbook for Assurance Professionals. NASA HDBK-4008R, Programmable Logic Devices (PLD) Handbook addresses the engineering processes for developing PLDs.

Software covered by this Standard is also required to implement the NASA software engineering requirements of NPR 7150.2, NASA Software Engineering Requirements, and the NASA software assurance requirements of NASA-STD-8739.8, NASA Software Assurance Standard. This Standard stresses coordination between software engineering and software safety assurance, as well as with system safety and software development, to maintain the system perspective and minimize duplication of effort.

Requirements—i.e., mandatory actions—are denoted by statements containing the term "shall," are identified by "(Requirement)," and are numbered [SSS-###]. The terms: "may" or "can" denote discretionary privilege or permission, "should" denotes a good practice and is recommended, but not required, "will" denotes expected outcome, and "are/is" denotes descriptive material.

This Standard often refers to recording information in an "appropriate document." It is not the intent of this Standard to designate what documents a program, project, organization, or facility must generate. The software safety information is recorded within the program, project, organization, or facility documentation as a quality record, but the exact type of documentation is left to the program, project, or facility. The SMA organization needs to keep their own records, reports, metrics, as well as analyses, lessons learned, and trending results. When specific plans are mentioned (e.g., a software safety plan), they can be standalone documents or incorporated within other documents (e.g., a system safety plan, a software management plan, a software development plan, or a software or system assurance plan). However, both the program, project, or facility, and the SMA have sign-off authority on safety planning and documentation.

1.3 Requirement Relief

Relief from requirements in this Standard for application to a specific program or project shall be formally documented as part of program or project requirements and approved by the Technical Authority in accordance with procedures in NPR 8715.3, paragraph 1.13 and NASA-STD 8709.20, Management of Safety and Mission Assurance Technical Authority.

2. APPLICABLE DOCUMENTS

2.1 General

The documents listed in this section contain provisions that constitute requirements of this Standard as cited in the text. The latest issuances of cited documents apply unless specific versions are designated. Non-use of specific versions as designated shall be approved by the responsible Technical Authority. The applicable documents are accessible via the NASA Standards and Technical Assistance Resource Tool at http://standards.nasa.gov or may be obtained directly from the Standards Developing Organizations or other document distributors. NASA Directives are available at: http://nodis3.gsfc.nasa.gov/.

2.2 Government Documents

National Aeronautics and Space Administration

NPD 7120.4	NASA Engineering and Program/Project Management Policy
NPR 7120.5	NASA Program & Project Management Processes & Requirements
NPR 7123.1	NASA System Engineering Processes and Requirements
NPR 7150.2	NASA Software Engineering Requirements
NPR 8715.3	NASA General Safety Program Requirements
NASA-STD-8709.20	Management of Safety and Mission Assurance Technical Authority (SMA TA) Requirements

2.3 Non-Government Documents

None.

2.4 Order of Precedence

This Standard establishes requirements for software safety but does not supersede nor waive established Agency requirements found in other documentation. Conflicts between this Standard and other requirements documents will be resolved by the responsible Technical Authority.

3. ACRONYMS AND DEFINITIONS

3.1 Acronyms and Abbreviations

ASIC	Application-Specific Integrated Circuit
CASE	Computer-Aided Software Engineering
CDR	Critical Design Review
CE	Complex Electronics
CoFR	Certification of Flight Readiness
COTS	Commercial Off-The-Shelf
CPLD	Complex Programmable Logic Devices
CSCI	Computer Software Configuration Item
DCR	Design Certification Review
DR	Discrepancy Report
FACI	First Article Configuration Inspection
FAR	Flight Acceptance Review
FDIR	Fault Detection Isolation and Recovery

FIFO	First-In-First-Out
FMEA	Failure Modes and Effects Analysis
FPGA	Field Programmable Gate Array
FTA	Fault Tree Analysis
GOTS	Government Off-The-Shelf
IEEE	Institute of Electrical and Electronics Engineers
IT	Information Technology
IV&V	Independent Verification and Validation
LIFO	Last-In-First-Out
MOTS	Modified Off-The-Shelf
MOA	Memorandum of Agreement
MOU	Memorandum of Understanding
NASA	National Aeronautics and Space Administration
OSMA	Office of Safety and Mission Assurance
OTS	Off-The-Shelf
PAR	Pre-flight Acceptance Review
PDR	Preliminary Design Review
PHA	Preliminary Hazard Analysis
PLD	Programmable Logic Devices
PR	Problem Report
RFP	Request For Proposal
SMA	Safety and Mission Assurance
SCM	Software Configuration Management
SoC	System-On-a-Chip
SRS	Software Requirements Specification
SSCA	Software Safety Criticality Assessment
TA	Technical Authority
TRR	Test Readiness Review

3.2 Definitions

The terms used in the main body of this document are consistent with the terms defined in NASA-STD 8709.22, Safety and Mission Assurance Acronyms, Abbreviations, and Definitions. Additional terms used herein and their definitions are:

Acquirer: The entity or individual who specifies the requirements and accepts the resulting software products, including software embedded in a hardware system. The acquirer is usually NASA or an organization within the Agency but can also refer to the Prime contractor – subcontractor relationship as well.

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

Catastrophic: [1] A condition causing fatal injury to personnel, and/or loss of one or more major elements of the flight vehicle or ground facility. A condition that may cause death or permanently disabling injury, major system or facility destruction on the ground, or loss of crew, major systems, or vehicle during the mission.

[2] Loss of human life or permanent disability; loss of major system; loss of vehicle; loss of ground facility; severe environmental damage.

Concur: The term *concur* or *concurrence* means to agree and accept, via signature, the readiness and content of a condition, requirement, report, deviation, document, etc. This also implies that if the stakeholder (e.g. SMA) does not concur, that their sign-off is withheld and the document, waiver, deviation package, test report, hazard report, etc. is not to be considered acceptable until such changes are made to achieve agreement on the deliverable.

Programmable Logic Devices (PLD) or Complex Electronics (CE): A programmable logic device or PLD is an electronic component used to implement user-defined functions into digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. PLD functionality is typically described using a hardware description language (HDL), such as Verilog or VHDL, which is then converted, using PLD vendor-specific tools, into the hardware gate structure of the PLD to implement the function. PLDs can be implemented one-time or can be reconfigurable**.**

While PLD functionality is generally described using an HDL, it can also be written in specialized variations of other programming languages. The functionality can range from a simple set of gates to a very complex set of gates (i.e. an embedded processor). The compilation of gates is hardware (regardless of the complexity) including the development of the embedded processor.

For the purposes of this Standard, any software to be executed on a processor embedded within a PLD will be evaluated from a software safety perspective. The design and resulting hardware will be evaluated from a system safety perspective and is not the purview of this standard. The software tools used to generate safety critical PLDs configuration files will be evaluated from a limited COTS safety perspective as per sections 6.5 and 6.6 of this standard.

Critical: [1] The condition where failure to comply with prescribed contract requirements can potentially result in loss of life, serious personal injury, loss of mission, or loss of a significant mission resource. Common uses of the term include critical work, critical processes, critical attributes, and critical items.

[2] A condition that may cause severe injury or occupational illness, or major property damage to facilities, systems, or flight hardware.

Firmware: The combination of a hardware device and computer instructions and data that reside as read-only software on that device. This term is sometimes used to refer only to the hardware device or only to the computer instructions or data, but these meanings are deprecated. The confusion surrounding this term has led some to suggest that it be avoided altogether. For the purposes of this Standard: Firmware is considered as software. Firmware is NOT the same as Programmable Logic Devices/Complex Electronics.

Hazard Analysis: [1] Identification and evaluation of existing and potential hazards and the recommended mitigation for the hazard sources found.

[2] The process of identifying hazards and their potential causal factors.

Likelihood: *Likelihood* is the chance that something might happen. *Likelihood* can be defined, determined, or measured objectively or subjectively and can be expressed either qualitatively or quantitatively (using mathematics). [From ISO 31000 2009 Plain English Risk Management Dictionary.] For this document looking at the software contribution; likelihood does not solely represent a probability of the initiating software cause, as these are systematic faults; it is a qualitative estimate of the likelihood of the software fault to propagate to the hazard (top level event). Factors such as control autonomy are rolled into that likelihood.

Off-The-Shelf (OTS) software: Includes Commercial Off-The-Shelf (COTS), Government Off-The-Shelf (GOTS), and Modified Off-The-Shelf (MOTS) software. OTS software may also be legacy, heritage, and re-use software. Refer to section 6.6 of this Standard for applicability to COTS.

Preliminary Hazard Analysis: A gross study of the initial system concepts. It is used to identify all of the energy sources that constitute inherent hazards. The energy sources are examined for possible accidents in every mode of system operation. The analysis is also used to identify methods of protection against all of the accident possibilities. Software's high level roles in contributing to or protecting the system should be considered and recorded (e.g. software's inadvertent release of an energy source or the detection and inhibit of an energy source).

Provider: The entities or individuals that design, develop, implement, test, operate, and maintain the software products. A provider may be a contractor, a university, a separate organization within NASA, or within the same organization as the acquirer.

Safety Critical: 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.

Safety Data Package: As used within this document refers to all or a combination of Program/Project/Facility failure modes and effect analysis (FMEA), hazard analysis/report, criticality analysis, reliability analysis, computing system safety analysis, orbital debris assessment report/end of mission plan, and documentation utilized to objectively demonstrate the software measures taken to control/mitigate hazards/conditions/events have been verified and successfully implemented. Other analysis such as the software requirements, software design analysis, mapping between the software safety requirements and the safety data package can be included in the Software Safety Case Study.

Software: Computer programs, procedures, rules, and associated documentation and data pertaining to the development and operation of a computer system. Software also includes COTS, GOTS, MOTS, embedded software, reuse, heritage, legacy, auto generated code, firmware, and open source software components. (Definition from source document: NPD 7120.4, NASA Engineering and Program/Project Management) For the purposes of this standard, the term software will also include scripts, glueware, and wrappers.

Software Assurance: The planned and systematic set of activities that ensure that software life-cycle process and products conform to requirements, standards, and procedures. For NASA this includes the disciplines of software quality (functions of software quality engineering, software quality assurance, and software quality control), software safety, software reliability, software verification and validation, and IV&V.

Software-Related Event/Item: Event (e.g., hazard, fault, failure mode or failure) or item (e.g., hazard control, requirement, function or mitigation) whose outcome or desired outcome is inextricably or causally linked to software functionality (i.e., without software, the event or the desired end-state as expressed by the item would not be possible).

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 throughout all phases of the system life cycle.

Software Safety: The aspects of software engineering and software assurance 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 or to its mitigation or control, to ensure safe operation of the system.

Tailoring: The process used to refine or modify an applicable requirement by the implementer of the requirement. If the revised requirement meets/exceeds the original requirement, and has no increase in risk from that of the original requirement, then it may be accepted/implemented by appropriate local authority; otherwise a waiver/deviation may be required. This definition of "tailoring" is for use within the application of the SMA TA as defined by the Chief, Safety and Mission Assurance.

4. ACQUIRER AND PROVIDER GENERAL ROLES & RESPONSIBILITIES

Software safety requires a coordinated effort among all organizations involved in the development, operations, maintenance, and acquisition of NASA systems and software including acquisition professionals, program, project, and facility managers, hardware and software designers and developers, safety analysts, software assurance engineers, and operations personnel. The acquirer and provider program, project, or facility organizations in conjunction with the acquirer and provider safety and mission assurance organizations (see section 4.2 of this Standard) are responsible for the development of a safe program, project, or facility, including the software elements (refer to NPR 8715.3). NASA holds safety as a core value and expects all personnel to contribute to safe systems and working environment.

The acquiring NASA SMA organization shall perform the initial Software Safety Criticality Assessment (SSCA) Part 1 using the Software Safety Litmus Test in Appendix A of this document to determine whether a project has safety critical software (Requirement). [SSS-001]

The initial assessment does not include the risk assessment portion (Part 2) of the SSCA. The initial SSCA (Part 1) can be based on early lifecycle items such as a preliminary system safety analysis, critical items list, and system operational concepts. The initial assumptions are documented based on the SSCA. This early assessment defines the initial applicability of this

Standard (i.e. this Standard is applicable to safety critical software; it is not applicable to non-safety critical software) and allows resources to be planned for software safety activities. The SSCA is re-performed throughout the life-cycle (Refer to SSS-024) to refine this Standard's applicability to the systems computer-software-configuration-items (CSCIs). The SSCA is discussed in more detail in Section 5.0. See SSS-007 for the Technical Authority involvement. Similar to other risk-based scales,, hazard severity categories range from negligible to catastrophic, and likelihood categories range from improbable to likely.

While the focus of software safety resources and this standard is on the critical and catastrophic hazard severity designations, the other parts of the scale need to be shown for completeness and consideration in case the software or a particular hazard slides in to the more critical or higher likelihood range at some point in the future. Negligible software safety risks are not intended to be addressed by this standard and Marginal and Moderate only where the likelihood or severity is very high (please see Table A-2 in Appendix A). This standard represents the minimum requirements for providing the processes for software safety. Applying additional safety requirements (both technical and process), while recommended, is left up to the project or Center. Pre-tailoring based on risk has been provided along with some guidance in the appendices of this standard.

The System Risk Index specifies the hazard risk for the system *as a whole*. The software element of the system inherits from the system risk, modified by the extent with which the software controls, mitigates, or interacts with the hazardous elements. In addition, the complexity of the software and the development environment play a role. Merging these two aspects is not an exact science, and most of the information presented in the Appendices is meant to guide the scoping and then tailoring of the safety effort. The numerous charts provided are to be used only as a starting point when determining the level of safety effort.

The process of scoping the software safety effort begins with the determination of how much software is involved with the hazardous aspects of the system. The Preliminary Hazard Analyses (PHA), initial criticality assessment, preliminary risk analyses, Preliminary Software Hazard Analysis, critical items list, Preliminary Software Risk Assessments, and other analyses provide information for determining whether systems and subsystems should be initially categorized as safety-critical.

Scoping the software safety effort can be accomplished by following four steps:

1. Identify safety-critical software (part 1 of SSCA, aka SW Safety Litmus Test)

2. Determine safety-critical software criticality level (i.e., how critical is it?) (part 2 of SSCA)

3. Determine the extent of development effort and software safety effort required using Appendix A SSCA and the SSS Applicability Matrix.

4. Refine focus and effort as required based on reassessments (repeating steps 1-3 above) at software requirements and design reviews.

The third scoping step leaves some choices for meeting the needed development assurance levels. Using the information gained from this scoping process, as well as input from the safety and assurance engineers, the program, project, or facility can better tailor the effort needed.

4.1 Acquirer and Provider Program, Project, and Facility Organization

The program, project, and facility organizations have the ultimate responsibility for the safety of the system including the software. The program, project, and facility organizations meet and implement the system safety requirements (process and technical) and the software safety requirements (process and technical). Both the acquirer and the provider program, project, and facility organizations are responsible for making sure the system is evaluated for the presence of safety critical software. Detailed acquirer and provider requirements and activities are defined throughout this Standard for the program, project, and facility life-cycle.

4.1.1 Acquirer Organization

The acquirer is responsible for implementing the acquirer software safety program and providing adequate resources for the software safety program throughout the program, project, or facility life-cycle. The acquirer is responsible for overseeing the correct development, implementation, delivery, operations, maintenance, and retirement of the software system.

When safety critical software is developed or acquired by or for NASA, the acquirer organization shall meet the requirements of this Standard (Requirement). [SSS-002] The acquirer's plan to meet the requirements of this Standard is defined as part of the acquirer software safety plan. Refer to section 7.1.1 of this Standard.

When safety critical software is developed or acquired by or for NASA, the acquirer organization shall impose the requirements of this Standard on the provider and verify that the provider implements the requirements of this Standard, along with any other contract specific software safety requirements (Requirement). [SSS-003] The acquirer needs to flow the necessary requirements of this standard and any additional requirements to the provider via the contract, MOA, MOU, or Task Order and then assure that they are followed. This requirement applies to software as defined in Section 3.1. Pretailoring is available in Appendix A of this document, Step 4 of the Criticality Risk Assessment.

4.1.2 Provider Organization

The provider is responsible for implementing the provider software safety program and providing adequate resources for the software safety program throughout the program, project, and facility life-cycle. When safety critical software is developed or acquired by or for NASA, the provider organization shall adhere to the requirements of this Standard (Requirement). [SSS-004] The applicable provider requirements of this standard apply to all software, as defined in Section 3.1.

4.2 Acquirer and Provider Safety & Mission Assurance Organization

The responsible organization performing SMA provides an independent assessment and evaluation of the safety process and technical implementation by either performing all or part of the safety analysis; verifying and validating the software safety analysis and processes performed by the program, project, and facility. Within Safety & Mission Assurance (SMA), the software assurance organizations have the responsibility to ensure the performance of the activities within this Standard.

The acquirer and the provider SMA organizations are responsible for ensuring that the software safety program meets the requirements of this Standard. Detailed acquirer and provider SMA requirements and activities are defined throughout this Standard for the program, project, and facility life-cycle. The SMA organizations have the responsibility to notify the program, project, and facility if the system (or a portion of the system) is unsafe. The SMA organizations are responsible for raising any problems or concerns to the appropriate levels, including to NASA's Office of Safety and Mission Assurance (OSMA), throughout the development and prior to software acceptance. Refer to NASA-STD-8739.8 for the software assurance details and requirements.

4.2.1 Acquirer SMA

The organization responsible for the acquirer SMA functions will meet the acquirer SMA requirements within this Standard (hereinafter referred to as "acquirer SMA"). The acquirer SMA ensures that the acquirer, provider, and provider SMA meet the requirements within this Software Safety Standard.

4.2.2 Provider SMA

The organization responsible for the provider SMA functions will meet the provider SMA requirements within this Standard (hereinafter referred to as "provider SMA"). The provider SMA ensures that the provider meets the requirements within this Software Safety Standard.

5. SOFTWARE SAFETY CRITICALITY ASSESSMENT (SSCA)

As systems increase in complexity, software's importance in system design and operation increases as well. When a system is determined to be safety critical, the initial assumption is that the software used within the system is safety critical. The use of software within that system is assessed with special attention to its contribution to safety. The key for the assessment is to look at the entire system and see what roles and functions are or will be implemented within software. The software components are evaluated not in isolation, but rather as functions within the system (including the end-user of the system).

The Software Safety Criticality Assessment (SSCA) consists of two parts. The first part requires the use of the Software Safety Litmus Test in Appendix A to determine if the software is performing a safety critical function including verification and validation of a safety critical software, hardware, operations component, subsystem, or system. The second part of the SSCA is the software risk assessment that is used to determine the level of risk, by examining both the likelihood and severity of overall software risks. This includes determining the level of software control, autonomy, time to action, complexity, and usage of the software.

SSCA Part 1: Software Safety Litmus Test	Required for all SSCA. All software must use the Software Safety Litmus Test in Appendix A. The results of the analyses must be documented and kept. This Standard does not apply to software that does not pass the SW Safety Litmus Test, however, the test needs to be reapplied as the project changes and the software

	matures, especially through the design phases.
SSCA Part 2: Software Risk Assessment	Performed with each update as the safety critical software matures. Recommend to be part of input to milestone reviews. Not required for initial SSCA performance or if software under consideration did NOT pass the SW Safety Litmus Test and is thus not safety critical. *Refer to SSS-001.* Performed to determine the level of risk the software in whole or in part (e.g. CSCI, or to CSC) poses to the system and thus better determine level of effort. Based on the highest potential hazard contribution, and or controls and mitigations the software performs to contain a risk, the software is assessed at the lowest possible known software configuration unit for the phase of the project. Thus, if at the requirements phase, the software is identified to have 4 potential contributions to various degrees of severity hazard, the highest severity level is the determining risk level for the software. Each assessment and its results must be documented and maintained.
Applicability Matrix	Based on results of SSCA Part 1 and Part 2. Provides "pre-tailoring" of requirements based on risk. *Refer to Appendix A, Step 4.*
SSCA Form	The results of the SSCA are required to be documented and approved. The use of this form is preferred but optional.

The SSCA form documents the software safety criticality as determined by the Software Safety Litmus Test and the software risk assessment (if the software is safety critical). The SSCA is performed during the formulation phase and repeated throughout the lifecycle as required within this document, to determine if the software is safety critical and software's possible contribution to, or control of a system hazard. As the role of software can shift over the development and life of the program, project, and facility, this assessment needs to be repeated.

The acquirer's initial Software Safety Criticality Assessment (SSCA), using Part 1, determines whether the software is safety critical (refer to requirement SSS-001). The provider SMA shall perform a Software Safety Criticality Assessment (SSCA) per Appendix A, first at the software functionality level and then, as the project development progresses, at a level of detail consistent with and appropriate for the analysis of hazards, conditions, and events (Requirement). [SSS-005] Refer to Appendix A of this Standard. Information which can be used to perform the SSCA includes: an initial hazard evaluation; concept of operations (COP), Request For Proposal (RFP), the preliminary system safety analysis; computing system safety analysis; reliability analysis; and system safety analyses. Initially, the software's functionality will be used to help determine its critical role and possible safety-related contributions to hazards, conditions, and events.

As the software development progresses to design and CSCIs are identified, additional SSCAs using the Software Safety Litmus Test help to refine not only the critical contributions of software, but also aid in the traceability and optimal location for adding mitigations and controls. The assessment is completed at the lowest software unit possible to ensure the requirements of this Standard are applied to the safety critical software and need not be applied to non-safety critical software. As the system and the software mature through the life-cycle, the provider with the provider SMA re-performs the SSCA to ensure the correct application of this Standard. This re-assessment is performed at each milestone review, at a minimum (refer to SSS-024). Thus, while a high level assessment in the formulation and contract proposal stages is usually performed to assess the overall safety criticality of the software, as development progresses, more detailed SSCAs need to be performed on the CSCIs, elements, or modules which take into account severity, complexity, autonomy, and likelihood.

The provider shall obtain approval from the acquirer SMA that the provider's Software Safety Criticality Assessment (SSCA) results are accurate (Requirement). [SSS-006] This includes the software safety criticality and software risk assessment.

The software risk assessment allows the provider to refine the requirements applicability based on risk because the software safety requirements within this document are not one-size-fits-all. The risk assessment results are documented on the SSCA and within the provider's software safety plan. Refer to Appendix A. The results may be presented at an Acquirer phase safety reviews or area safety board or independent from planned reviews for approval as documented in the project software safety plan.

Any disagreements between organizations, regarding the SSCA shall be elevated up the NASA Technical Authority SMA chain of command per NASA-STD-8709.20 (Requirement). [SSS-007] The Office of Safety and Mission Assurance (OSMA), working with the headquarters program office, will have the final resolution authority. It is expected that engineering and SMA technical authorities will consult and make each other aware of disagreements in meeting and interpreting requirements.

The provider shall maintain the approved SSCA as a quality record (Requirement). [SSS-008] The approved SSCA will be made available to the acquirer. This includes the analysis results of the Software Safety Litmus Test, the level and identity of the software component, and the results of the software risk assessment as per Appendix C of this Standard. Quality records are maintained per NPR 1441.1, NASA Records Retention Schedules.

6. GENERAL SOFTWARE SAFETY ACTIVITIES THAT APPLY THROUGHOUT THE LIFE-CYCLE

Software safety activities take place in every phase of the system and software life-cycle, beginning as early as the concept phase and continuing on through operations to retirement. Performance of these activities, beginning with up-front participation and continuing with analyses and subsequent reporting of safety problems as they are found during the software life-cycle, leads to timely, better integrated and less costly solutions.

6.1 Documentation Requirements

The exact documents that a provider will produce, document delivery timeframes, and document approval authority are defined by the acquirer during the planning phase (see section 7.1 of this Standard). The provider's documentation deliverables should be reflective of the program, project, or facility size, complexity, classification and safety criticality. Contractual obligations or required standards may also influence the amount and type of documentation produced.

The provider addresses and documents the safety critical software in program, project, or facility documentation as defined in the provider's software safety plan. For example, the software configuration management (SCM) plan could explain how the program, project, or facility tracks and identifies safety critical software or for a small project where there is no standalone SCM plan, these requirements might be described in the management plan or software development plan. If the software safety process is captured across multiple documents, the provider's software safety plan will identify the location of all the software safety requirements. The program, project, and facility documentation that addresses the software safety process and deliverables can include:

- Software Safety Plan
- Software Development Plan
- Software Safety Criticality Assessment (SSCA)
- Software Project Management Plan
- Software Configuration Management Plan
- Software Assurance Plan
- Software Requirements Specification
- Software Design Documentation
- Verification and Validation Plan
- Version Description Documentation
- Safety Analyses and Reports
- Test Documentation
- User Documentation and Procedures
- Operations and Maintenance Plan
- Risk Plans, Analyses and Reports
- Retirement/Replacement Plan

The provider SMA shall have approval authority over the type and distribution (single or multiple software, system, or facility documents) of documentation which contain provider software safety requirements (process and technical) (Requirement). [SSS-009] This allows smaller projects to combine documentation without losing the benefit of provider SMA approval. SMA is seen as co-approval authority on documents that flow down software safety requirements.

The acquirer SMA shall have approval authority over the type and distribution (single or multiple software, system, or facility documents) of documentation which contain acquirer software safety requirements (process and technical) (Requirement). [SSS-010] This allows smaller projects to combine documentation without losing the benefit of acquirer SMA approval.

SMA is seen as co-approval authority on documents that flow down software safety requirements.

6.2 Safety Requirements Traceability

Traceability is a link or definable relationship between two or more entities. Requirements are linked from their more general form (e.g., the system specification) to their more explicit form (e.g., subsystem specifications). They are also linked forward to the design, source code, and test cases. Because many software safety requirements are derived from the system safety analysis, risk assessments, or organizational, facility, vehicle, system specific generic hazards, these requirements; these will also be linked (traced) from those specific safety data products (examples include safety analysis, reliability analysis, etc.).

Tracing requirements is a vital part of system verification and validation, especially in safety verifications. Full requirements test coverage is virtually impossible without some form of requirements traceability. Tracing also provides a way to understand and communicate the impact on the system of changing requirements or modification of software elements. A tracing system can be as simple as a spreadsheet or as complicated as an automatic tracing tool.

The provider shall trace, and maintain currency of, the relationships between software safety requirements and software-related system hazards, controls, conditions, and events (Requirement). [SSS-011] The trace from the software safety requirements to design, implementation, and test is required per NPR 7150.2 (requirements SWE-059, SWE-064, and SWE-072 in Revision A). Software safety requirements must be uniquely identified as per SSS-038. Not all hazard contributions will have controls or mitigations as they may be of lower severity and likelihood of occurrence. When a hazard contribution is agreed by the acquirer SMA and project to be acceptable without mitigation then, that agreement is documented and there is no need to trace it further. It will remain on record within the safety analysis report, safety case or safety data package. The relationship of software safety requirements to hazards, controls, conditions and events is usual kept in the hazard or safety report and in the requirements document where not only is the requirement(s) identified as safety critical, but enough detail is flowed down with the resulting safety requirement(s) in order to capture needed conditions, triggers, contingencies, etc..

The provider will keep the tracing results under configuration management and maintain the tracing results as a program, project, and facility quality record. Refer to section 6.3 of this Standard.

The provider SMA shall evaluate and concur with the tracing results created in SSS-011 (Requirement). [SSS-012] These tracing analysis results will be made available to the acquirer and the acquirer SMA at major milestone reviews and during audits, at which time they will either accept or direct any changes needed. Refer to SSS-014.

6.3 Software Configuration Management Activities

Safety critical software is managed in accordance with the software configuration management process that is documented and approved within the configuration management plan. Refer to NPR 7150.2 section 4.1 for additional details regarding software configuration management.

The provider SMA shall ensure the provider program, project, and facility is implementing configuration management of software safety artifacts (Requirement). [SSS-013] The software safety artifacts include at a minimum:

- Documentation described under section 6.1
- Software safety requirements, design, and code analysis/results
- Test analysis/results
- Trace/compliance matrices
- Waivers and deviations
- Audits, assessments and reviews
- Other documentation as defined in the program, project, or facility software safety plan

This includes configuration management, change control, status accounting, and change verification of safety critical software requirements and software elements. Additional attention to configuration management by the provider is necessary when the provider is acquiring software (and software artifacts) from other providers.

The provider shall provide electronic access to the software safety artifacts, defined in SSS-013, to the acquirer, the acquirer SMA, and IV&V (when IV&V is performed on the provider's program, project, or facility), for major milestone reviews, assessments, and upon request (Requirement). [SSS-014] The acquirer needs "read permission" access (without the need to be onsite of the provider location) to project documentation in order to give feedback as early as possible. The configuration management system, mentioned above, does not have to be unique for software safety artifacts.

6.4 Waivers, Deviations, and Tailoring

Deviations and waivers are required when the software project cannot meet the requirements within this Standard. A deviation is requested prior to implementation and includes an alternative method with proof of why it will be as safe as what is required. A deviation is a planned alternative to a requirement. A waiver is requested after project implementation has failed to meet the requirement. If a requirement does not apply, then the reasons for non-applicability are recorded and agreed to by the acquirer SMA. For other requirements' deviations or waivers, to proceed, the provider will suggest alternative ways to meet the requirement or provide rational why the acquirer can choose to accept the risk. All this is documented in a waiver or deviation package and must include the potential impact to system safety.

Tailoring is the process used to refine or modify an applicable requirement by the implementer of the requirement. If there is no increased risk in the proposed implementation then the local SMA can approve implementation; otherwise a waiver or deviation is required. Significant tailoring outside that provided by the tables in this standard's appendices, should be recorded or noted within the appropriate documents. Refer to NASA-STD-8709.20 Management of SMA Technical Authority Requirements. For deviations and waivers to software engineering requirements, refer to NPR 7150.2, Section 6.

Adherence to requirements applicability, as defined in Appendix A of this Standard, does NOT require a waiver or deviation package when a software risk assessment is performed and documented as part of the approved SSCA.

If one or more requirements (i.e., a numbered “shall [SSS-xxx]” statement) contained within this Standard cannot be met by program, project, and facility SMA, a waiver or deviation package shall be prepared in accordance with NASA-STD-8709.20 (Requirement). [SSS-015] The provider develops the package for the provider and provider SMA requirements within this Standard; the acquirer develops the package for the acquirer and the acquirer SMA requirements within this Standard.

The responsible SMA organization shall evaluate and submit the waiver or deviation package (created in SSS-015) to the responsible NASA SMA Technical Authority or delegated designee for approval in accordance to NASA-STD-8709.20 (Requirement). [SSS-016] The provider SMA evaluates the provider’s package prior to submittal to the acquirer SMA; the acquirer SMA evaluates its own as well as provider packages prior to submittal to OSMA.

The acquirer and acquirer SMA shall maintain a copy of all waivers and deviations to requirements contained in this Standard as a quality record (Requirement). [SSS-017] These variances are provided to the NASA Headquarters Office of Safety and Mission Assurance upon request. Quality records are maintained per NPR 1441.1, NASA Records Retention Schedules.

6.5 Tool Support and Safety Assessment

The use of tools, in a software or hardware development and maintenance effort can be of great benefit in producing consistent analyses, code, finding errors, etc.; however, their use can inadvertently introduce software hazards into the software or hardware. These tools are defined, accredited, and configuration managed per NPR 7150.2. Examples of such tools include: Computer-Aided Software Engineering (CASE) products, compilers, editors, fault tree generators, simulators, verification and validation tools, code generators, SCM tools, emulators, Computer Aided Design (CAD), and test environments for hardware and software.

Software tools, created or purchased, used on safety critical systems could impact those safety critical systems and thus those tools could be safety critical software. Therefore, software tools need to be assessed using the SSCA. A software tool’s potential contributions to injecting or allowing hazardous errors need to be assessed for level of severity and likelihood (See Appendix F of this Standard for checklists that provide guidance and insight into the possible contributions.). Tools bought for use in testing, modeling, simulating test scenarios, configuration managing critical data and files, designing hardware, assessing flight, launch, landing re-entry algorithms, testing and creating configuration files for PLD, as well as any created applications running on any of these tools, will be looked at as COTS first and foremost. As COTS, they will be subjected to the black box, COTS level of software safety. Unless a specific agreement is reached with the vendor, their development processes and products remain proprietary and unavailable. What can be done is to verify the inputs and outputs, check the configuration and version control, check the limits of the tools’ abilities, and address the potential risks of them not performing as “advertised” or as programmed. Any application files that are created to run on a tool will be looked at as software in most cases. An expert in Programmable Logic Devices, structures, or in MatLab, etc. may be needed to assess and address specific language and criteria. The important point is that these tools and any software created to run them are assessed for 1) possible contributions to hazards and 2) that when used for designing, coding or testing any safety features, they do so correctly.

The provider shall use the SSCA to determine if the tools planned for use on the program, project, and facility (including any changes to these tools) have a potential impact on the safety of the system (Requirement). [SSS-018] If the tools are safety critical, then the tools are subject to the requirements of this Standard as per the guidance and pre-tailoring in Appendix A: SSS Applicability Matrix. The SSCA results can be documented on the SSCA form provided in Appendix A. Proper expertise for a particular tool needs to be included in or conducting the assessment process. Software that runs test and evaluation equipment such as thermal vacuum chambers, shake and vibration tables, acoustic chambers, etc., also need to be evaluated for software's impact to safety.

Performing a SSCA and using the additional guidance below and in Appendix F, determine if the software tools (e.g., simulators, models, emulators, compiler libraries, built in memory checkers, materials analysis, trajectory analysis) have an impact on the safety of the system. If so, then further determine if they are correctly applied and used, operated within range of their limitations, the results are documented, etc. Typically, once it is determined if a tool has any safety implications, an assessment is performed of the severity of impact if the tool provides one or more wrong outputs, and the likelihood of that occurring determines the level of effort or hazard mitigations to employ.

For standalone tools that are a project unto themselves- the tool project documents the credentials and the calibrations as well as the ranges for operation that are to be verified. If created and developed as part of a project, facility, or program, then they are considered and treated as software.

6.6 Off-the-shelf and Reused Software

Use of Off-The-Shelf (OTS) and reused software that is not developed specifically for the safety critical system can be risky. Software in this category includes off-the-shelf software (e.g., operating system, application library, or hardware driver) and previously created software (e.g., from a past project). It is important to evaluate the differences between how the OTS or reused software will be used within the new system, and how it was used in previous systems. The differences in operational constraints or configuration of the software may affect the operation of the OTS or reused software, sometimes in unexpected ways. Refer to Appendix F of this Standard for COTS Software Safety Considerations. The responsibility for this lies on the provider and the acquirer; not on the OTS or reused software supplier unless defined in the OTS or reused software supplier contract.

A SSCA is performed on OTS and reused software per SSS-001and SSS-005. Most of the requirements of this Standard apply directly or indirectly to OTS and reused software. For COTS specific requirements applicability, see Appendix A's Applicability Matrix - COTS column. High level safety requirements and design can still be applied to COTS and any code written to integrate to, or protect it from, the system (wrappers and glueware) still need to be assessed and treated as safety critical software when appropriate. COTS may also have applications that run on them which need to be considered as well.

The provider with the provider SMA shall perform a safety analysis on OTS and reused software to evaluate its ability to meet required functions without impacting safety (Requirement). [SSS-019] The evaluation includes interactions, extra software functionality (even if the extra functionality is not planned for immediate use), the impact on safety, and interfaces to developed code. The evaluation is based on the functions that the OTS or reused software performs for the system. These functions, if referenced as a mitigation or control to a safety critical failure or hazard, are considered safety critical. However, often the problem with the introduction of OTS or reused software is that inadvertent activities are not addressed. Refer to Appendix F of this Standard for additional information on the assurance of OTS software products.

The provider will verify by the method of test that the OTS or reused software meet functional safety requirements and that the system is safe from inadvertent operations and failures. Refer to section 7.5.2.1 of this Standard for requirements on testing.

6.7 Security

The goal of the security measures is to protect against sabotage, collusion, compromise or alteration of safety critical software elements. Security breaches can impact the safety critical systems, therefore, NPD 2810.1 NASA Information Security Policy, and the associated document, NPR 2810.1 Security of Information Technology, are particularly important for safety critical software need to be considered.

The provider performs an analysis to ensure against security weakness and violations for any safety critical software. Refer to NPR 7150.2 (Section 2.1.4, requirements SWE-102, SWE-109, and SWE-115 in Revision A).

6.8 Milestone and Safety Reviews

The program, project, and facility can have multiple milestone and safety reviews. Refer to NPR 7120.5 NASA Program and Project Management Processes and Requirements, and NPR 8715.3 NASA General Safety Program Requirements. Each program and center is responsible for creating a series (phases 0 to 3) of independent safety reviews to monitor the progress of finding, addressing, and verifying the controls and mitigations to hazards, conditions, or events. Milestone reviews include any-NASA specific review such as Systems Requirements Review (SRR) Preliminary and Critical Design Reviews (PDR, CDR), Design Certification Review (DCR), First Article Configuration Inspection (FACI), Test Readiness Review (TRR), Certification of Flight Readiness (CoFR), Preflight Acceptance Review (PAR), Flight Acceptance Review (FAR), and can be formal or informal depending on the size, complexity and criticality of the system under review. Safety reviews (e.g., Payload Safety Review Panel, Computing Safety Panel, Safety Review Panel, Expendable Launch Vehicle Payload Safety Working Group) are usually associated with program, project, or facility milestone reviews but are not the same. Facility System Safety Review, Facility Software Safety Review, while independent from Facility Configuration Management Reviews, is also correlated. Refer to Appendix F for additional information regarding Facility Safety Considerations. The result of each of these program and safety reviews needs to feed into and inform the other. The hazard controls and mitigations agreed upon in a Safety Panel need to feed into the requirements and design of the program reviews.

The provider, working with the provider SMA, shall perform software safety analysis as part of the system safety analysis (Requirement). [SSS-020] This includes evaluating the system safety analysis for changes to hazards, conditions, and events that might impact the software. If PLDs are utilized, the provider SMA will ensure the PLDs are evaluated for their safety impacts on the system. Refer to Appendix C of this Standard for additional information on software safety analysis as part of the system safety analysis.

The acquirer shall assign the acquirer SMA as an approving member of the decision body at all program, project, and facility major milestone and safety reviews evaluating safety critical systems that have software (Requirement). [SSS-021] Refer to introductory paragraph of this section for example reviews. This is not an inclusive list.

The provider shall develop and maintain evidence of compliance to the requirements of this Standard for each major milestone (Requirement). [SSS-022] The provider SMA concurs with the evidence of compliance. The compliance evidence, usually a trace matrix, is made available to the acquirer and the acquirer SMA for evaluation of milestone completeness. For a large or complex program, project, or facility, the acquirer SMA may request the compliance evidence more often or at different reviews. This request would be documented in the provider's software safety plan.

At each safety review and milestone review, the provider shall identify any new or updated software safety requirements or design features (Requirement). [SSS-023] As the program, project, and facility matures through the life-cycle, the provider will use the updated system and software safety analysis to identify new or changes to existing software safety requirements and design features that reflect software control of critical functions, or where software is a potential control, cause or contributor to a hazard, condition, or event. Part of the analysis will determine the severity and likelihood of occurrence and help determine what, if any, mitigation is needed. If no mitigation is needed, due to low likelihood and severity or management acceptance of risk, this too is documented. Refer to NPR 7150.2 (requirements SWE-053,SWE-059, and SWE-117 in Revision A) for related Software Engineering activities.

In preparation for each major milestone review, the provider SMA shall re-evaluate the software using the SSCA and obtain the acquirer SMA approval of any changes and additions (Requirement). [SSS-024] The re-evaluation ensures the SSCA remains accurate and up-to-date as the program, project, or facility matures through its lifecycle. Further, this gives the program, project, and facility software a chance to distinguish between CSCIs that are and are not safety critical and determine the criticality and effort for each.

6.9 Software Change Requests and Discrepancy and Problem Reporting

Software safety must be considered when software is being changed through change requests or discrepancy and problem reporting within a safety critical system.

6.9.1 Software Change Requests

The software changes are tracked by the configuration management system utilized by the program, project, or facility. Contents of the software change request are defined in NPR 7150.2 (requirement SWE-113 in Revision A).

The provider shall assign the provider SMA as a voting member within the provider software change control process for safety critical software (Requirement). [SSS-025] The provider SMA has approval authority over the provider software requirements.

The acquirer shall assign the acquirer SMA as a voting member within the acquirer software change control process for safety critical software (Requirement). [SSS-026] The acquirer SMA has approval authority over changes to acquirer software requirements.

Prior to the change approval, the provider SMA shall analyze software changes, including those that result from problem or discrepancy resolution, for potential safety impacts, including the creation of new hazard contributions and impacts, modification of existing hazard controls or mitigations, or detrimental effect on safety critical software or hardware (Requirement). [SSS-027] In the event the proposed change is implemented prior to change approval, the software must go back through the change control process with SMA's impact analysis prior to proceeding. The program, project, and facility is proceeding at risk prior to change request approval.

The provider SMA shall concur on all changes to safety critical software, including discrepancy report dispositions (Requirement). [SSS-028]

6.9.2 Discrepancy and Problem Reporting and Tracking

Discrepancy Reports (DR) and Problems Reports (PR) contain a description of each problem encountered, recommended solutions, and the final disposition of the problem. These reports are normally generated after the software has reached a level of maturity (e.g., is baselined or in beta version). These PRs and DRs need to be tracked, and management needs to have visibility into past and current software problems.

The provider SMA ensures the use of a system for closed-loop tracking of discrepancies, problems, and failures in the baselined safety critical software products and processes. The closed-loop system can be a program, project, or facility system or unique to software. Often one informal tracking system is used for software up thru unit testing after which a formal, systems level problem reporting and corrective action system is used. However, it should be noted, safety issues need to be elevated and tracked at the system level. The discrepancy, change, and problem reporting system(s) is documented in the provider’s software safety plan.

The provider shall evaluate all discrepancy reports of safety critical systems for software safety impacts (Requirement). [SSS-029]

The provider shall trace the identified safety critical software discrepancies, problems, and failures to the associated system-level hazard, FMEA or fault tree event (Requirement). [SSS-030] This could drive an update if a new contribution or control has been identified or a hazard report or reliability analysis needs to be modified as a result of the DR/PR.

The provider SMA concurs on all discrepancy report dispositions of safety critical software. Refer to section 6.9, SSS-028.

7. SOFTWARE LIFE-CYCLE ACTIVITIES AND ANALYSIS

Software safety tasks are performed for each software life-cycle phase. Many of the safety analyses are evolutionary with the results from one phase feeding the analyses of the next phase. As the program, project, or facility evolves, so does the maturity of the safety analysis.

7.1 Software Safety Planning and Resources

Software safety activities will require resources. These resources may be shared within a program, project, or facility and SMA organization. Resources include financial, schedule, acquirer and provider personnel, computing resources, equipment support, and tools (e.g., testing facilities, test drivers, static analyzers, simulations, models, etc.). Software safety planning is performed early in the life-cycle, ideally at the project conception, to ensure sufficient resources are available for implementation. Refer to Chapter 3 of the NASA-GB-8719.13, NASA Software Safety Guidebook. For existing projects, the requirements of this Standard apply to the project phases yet to be started as of the effective date of this Standard to the extent determined on a case-by-case basis. Refer to section 1.2 for details.

7.1.1 Acquirer and Acquirer SMA Contract, MOU, and MOA and Planning Activities

The acquirer, with the acquirer SMA, shall develop and maintain under configuration management the acquirer software safety plan that includes the items listed below at a minimum (Requirement). [SSS-031] This can be a plan unique to a program, project, or facility, a generic plan for a specific facility or type of project, a center plan, or, in general, part of another acquirer plan. The acquirer software safety plan will include:

a. Identification of relevant stakeholders (including SMA, system safety, and development organizations) with responsibilities and general information including activities, resulting products and schedules.
b. Timeframe for initial SSCA and re-evaluations throughout the life-cycle with SSCA results. As part of the unique portion of an acquirer software safety plan, the Software Classification and safety criticality needs to be documented. Depending on the project timeline and agreement (contract, MOA, etc.) deliverables, need to state the relative times when the analyses will be conducted and the results made available (e.g. 30 days prior to PDR, CDR, and system integration test provide updated SSCA and provide updated SSCA report(s)).
c. Schedule of periodic evaluation and reporting of acquirer adherence to the acquirer software safety plan (based on phase, milestones, or, for larger long term projects, every two years at minimum).
d. General software safety and any unique qualifications or training required.
e. Traceability matrix for the requirements of this Standard showing the relationship between the acquirer's requirements of this Standard and the activities specified in the acquirer's software safety plan.
f. Acquirer SMA evaluation process for provider adherence to the provider software safety plan (e.g., traceability review, process audits, etc.).
g. Resources required to meet the acquirer software safety plan, resources to actually be applied, and resulting reasoning, including indication of any waivers or deviations.

h. Performance of periodic assessment of the provider problem or change reporting system to ensure software safety issues and discrepancies are correctly identified, elevated, and tracked to closure.
i. Process and participation responsibilities in system and facility safety reviews and any specific software safety reviews. This may be in a system or facility safety plan.
j. Acquirer safety analysis approach (including methods for identifying hazards, conditions, and events, assessing risks, implementing controls, and verifying and validating risk reduction) and how the analysis fit in with the overall acquirer system safety analysis, airworthiness flight safety, or flight readiness approach; list of acquirer software safety deliverables.

For Contract, MOU, and MOA (this includes Requests for Proposals), per NPR 8715.3 section 9.3, the following are completed:

a. The acquirer will invoke the requirements of this Standard on any Contract, MOA, or MOU that includes safety critical systems. This ensures this Standard is applied to any safety critical software within the safety critical system during all the lifecycles, formulation to retirement. Reference requirements SSS-002 and SSS-003 which invoke this standard on the acquirer and the provider.
b. When the acquirer identifies software safety requirements in addition to the requirements of this Standard for the development or acquisition of the program, project, or facility, the Contract, MOA, or MOU shall include the other software safety requirements (Requirement). [SSS-032] The other software safety requirements are often Center, or program, project, or facility specific or generic process and technical requirements needed to develop, implement and operate a safe system.
c. The acquirer SMA shall ensure the Contract, MOA, or MOU contains the appropriate requirements for safety critical software (Requirement). [SSS-033] This includes: deliverable provider documentation such as: software and system hazard analyses and reports, software safety plans, software safety test procedures, test reports, software deliverables with delivery milestones, process definition and implementation per NPR 7150.2, software safety per this Standard, and software assurance per NASA-STD-8739.8.

The acquirer SMA shall be responsible for evaluating and approving the provider's software safety plan (refer to SSS-036 of this Standard) (Requirement). [SSS-034] This ensures the accuracy and completeness of the provider's software safety plan as well as its ability to be implemented. (Refer to NPR 7123.1) Approval includes the baseline and any changes. NASA is ultimately responsible for the safety of the program, project, and facility, including software safety. This approval ensures NASA has addressed the software safety responsibilities.

7.1.2 Provider and Provider SMA Contract, MOU, and MOA and Planning Activities

The provider with the provider SMA shall develop, baseline and configuration manage a provider's software safety plan for meeting the software safety requirements levied in the Contract, MOU, or MOA, including the requirements of this Standard (Requirement). [SSS-035]

The provider shall obtain approval of the provider's software safety plan from the acquirer SMA (Requirement). [SSS-036] This includes the baseline version and any changes.

The provider shall include all of the following in the provider's software safety plan: [SSS-037] This can be a plan unique to a program, project, or facility, a generic plan for a specific facility or type of project, a division plan, even a center plan, or, in general, part of another provider plan. However, as project, facility, and tasks may vary, some means to document the specifics for that project, facility or task may be needed in addition to a generic software safety process plan. The provider SMA ensures the following items are included in the provider's software safety plan.

a. Designation of responsible organization implementing the provider software safety requirements (process and technical) and the provider SMA requirements. These can be different organizations or a shared responsibility where one addresses the process safety requirements and SMA requirements, while another organization addresses the technical safety requirements.
b. Relevant stakeholders (including Acquirer and Acquirer SMA, system safety, and development organizations) with responsibilities and general information including activities, resulting products and schedules.
c. Software classification (as per NPR 7150.2) and safety criticality (as per Appendix A). This includes documenting risk level, requirements applicability per Appendix A, and assumptions. As the program, project, and facility matures, the software classification, safety criticality, and risk may change or shift according to design. The provider's software safety plan (if the plan is an independent document or part of another document) will need to be updated to reflect the changes.
d. General software safety training and any unique qualifications or training required for or about the specific system and operational environment.
e. Schedule of periodic provider program, project, or facility evaluations and reviews and software safety evaluations and reviews addressing software in safety critical system functions. Evaluations and reviews are performed at major milestone reviews. For small projects, software Class D & E, such as experiments, this may be performed at concept, design and prior to acceptance or any test runs. For additional pre-tailoring, utilize the SSCA and the Applicability Matrix as defined in Appendix A.
f. Safety analysis approach (including methods for identifying hazards, conditions, and events, assessing risks, implementing controls, and verifying and validating risk reduction) and how the analysis fit in with the overall system safety analysis approach; list of software safety deliverables. See section 6.1 of this Standard.
g. Problem and change reporting, evaluation, and tracking system utilized by provider SMA to ensure that the software safety issues and discrepancies are correctly identified, analyzed, elevated, and tracked to closure.
h. Relative schedule of periodic SMA assessments to ensure the validity and execution of the provider's software safety plan throughout the entire software life-cycle. At a minimum, this will be performed by the provider with results available to the acquirer at each major milestone review, prior to delivery, prior to any upgrades going operational, and before retirement. Facilities will need to assess the projects that use them during proposal and prior to usage.
i. Traceability and compliance matrix showing the relationship between requirements of this Standard and the activities specified in the provider's software safety plan.
j. Documentation of the resource requirements and the allocation of those resources to software safety tasks.

Per NASA-STD-8739.8, NASA Software Assurance Standard, the provider SMA will track to closure any software safety issues and risks per the provider software safety plan and elevate to the provider and the acquirer any issues or risks that cannot be resolved. The software safety issues and risks do NOT require a separate tracking system; they can be part of the program, project, or facility tracking system. This function is typically performed by software safety assurance but it can be performed by system safety assurance.

7.2 Software Safety Requirements Determination and Analysis

Software safety requirements development and analysis are performed in conjunction with, or immediately following, the system safety analysis. Software is an integral piece of the system safety analysis. Software safety is included in the system safety to ensure the software related aspects of the hazards, conditions, events, and failures are correctly documented, analyzed, implemented, and verified. Refer to Appendix C of this Standard for additional information regarding software as part of the system safety analysis.

7.2.1 Software Safety Requirements Determination

Software safety requirements are required to carry a unique identification or tag for traceability purposes. A way to mark and trace these requirements throughout the development and operational phases is needed in order to enable assessment of impacts and changes to the requirements. The unique identification or tag can be a special section in the requirements document, or a flag beside the requirement, or within a database. The method of identification is not important as long as it can be used for traceability and assessment.

Software safety requirements do more than protect against unsafe system behavior. Software is used to command critical, must-work functions. Software can be used proactively to monitor the system, analyze critical data, look for trends, and signal or act when events occur that may be precursors to a hazardous state. Once a hazard, condition, or event occurs, software can be used to control or mitigate the effects of that hazard, condition, or event, possibly restoring the system fully or partially, or putting the system into a safe state. Therefore, program, project, and facility software requirements include the software safety requirements that will embody these behaviors, both proactive and reactive, and include the system and software states where they are valid.

NPR 7150.2 (requirement SWE-109 in Revision A) requires that all software requirements be included in a Software Requirements Specification (SRS). This includes the software safety requirements. These requirements are derived from the system safety requirements, standards, program specification, vehicle or facility requirements, and interface requirements. In a model-based development environment, models can represent design specifications and software requirements. The general principles in this Standard apply, but the methodologies, terminology, etc. will vary. DO-178C Software Considerations in Airborne Systems and Equipment Certifications is helpful for understanding the use of models in a formal environment with safety concerns. For flight systems, range safety requirements may need to be applied for launch and reentry.

Detailed software safety technical requirements are system dependent. NPR 7150.2 (requirement SWE-134 in Revision A) provides some general software safety requirements.

Additional, more detailed, software safety requirements for must-work and must-not work functions are provided in Appendix D of this Standard.

The provider shall uniquely identify (or tag) the software requirements which are safety critical (Requirement). [SSS-038] This identifier or tag will provide clear distinction that the requirement is safety critical. For additional information on identification or tagging of software safety requirements, both generic and specific, see NASA-GB-8719.13 NASA Software Safety Guidebook, Section 6.5.

The provider documents the valid and invalid modes or states of operation within the software safety requirements. Refer to NPR 7150.2 (requirement SWE-134 in Revision A).

The provider shall include all software related safety constraints between the hardware, operator, and software in the software requirements documentation (Requirement). [SSS-039] When the software, hardware or operator performs a safety critical function, document the hardware, software, and operator roles in that function, the precedence, and the failure modes as well as any known constraints, controls, mitigations, conditions, timing constraints, limits, and wear-out factors that impact how software needs to respond. Refer to NPR 7150.2 (requirement SWE-109 in Revision A). Also, it is strongly recommended that software requirements developers seek out inputs from the operators and hardware engineers, asking about constraints and operating conditions the software may need to account for. Assuming, without the facts, how hardware will operate over time or under certain conditions and environments leads to potential hazards. This is especially critical if during design or operations and maintenance, the hardware or operating conditions change.

7.2.2 Software Safety Requirements Analysis

The provider SMA works closely with the provider system safety to analyze the decomposition from the system safety requirements, as well as to verify and validate that the software safety requirements maintain the system in a safe state and provide adequate proactive and reactive responses to potential failures.

The provider SMA shall analyze the software requirements which are safety critical for the following (these are iterative processes performed, at a minimum, at each major milestone) [SSS-040]:

a. Correct decomposition from system requirements, as well as all hazards, conditions, and events. The software safety requirements should bi-directionally flow or decompose from/to the parent system requirements.
b. Correct identification as a software safety requirement. This ensures software safety requirements continue to be correctly and uniquely identified as the program, project, or facility matures through the lifecycle.
c. Technical correctness.
d. Verification and validation that the software meets critical functionality, that the associated controls and mitigations work, or the software takes the system to a safe state.

The provider SMA shall concur with the provider's identification (or tagging) of software safety requirements (Requirement). [SSS-041] Disagreements on the identification or tagging will be elevated to the delegated SMA technical authority and OCE technical authority for the technical safety requirements. Software safety process requirements will be elevated through the SMA technical authority. Joint resolution between engineering and SMA is needed to address differences in technical safety requirements identification and tagging as safety critical or not. For additional information on identification or tagging of software safety requirements see NASA-GB-8719.13 NASA Software Safety Guidebook, Section 6.5.

The documented results of the software requirements analysis, including newly identified software safety features, controls or contributions to hazards, failures, conditions, and events, hazard causes, and improperly decomposed requirements, shall be updated in the system safety data package by the provider (Requirement). [SSS-042] The provider ensures the updates, including updates to the SSCA, are incorporated into the system safety data package. Refer to Appendix E of this Standard for a list of items to consider for software contributions to the system hazards, conditions, and events or fault-tree events. Some program, project, and facility have pre-set fault tolerance or fault handling requirements for certain conditions or operations, these should be checked for completeness, proper application and software's contribution.

The provider software safety requirements analysis will be available to the acquirer and the acquirer SMA for program, project, and facility formal reviews, system-level safety reviews, and upon acquirer request. Refer to section 6.3 of this Standard.

7.3 Software Safety Design and Analysis

The software design planning and implementation of software safety requirements will identify which software design features are safety critical and which design methods are used to create them (NPR 7150.2 (requirement SWE-057 in Revision A)). The software design will be implemented so that the software safety features and requirements can be thoroughly tested (NPR 7150.2 (requirement SWE-058 in Revision A)). The safety requirements are traced to the design elements that implement the requirements, and each design element is traced back to its requirements (NPR 7150.2 (requirement SWE-059 in Revision A)). The provider SMA has the responsibility to conduct evaluations of the software design. To the extent practical, the software design will modularize the safety-related aspects of the design (ref. NASA-GB-8719.13 NASA Software Safety Guidebook).

Fault Detection, Isolation and Recovery (FDIR) design solutions can be used to reduce the impacts of hazards. FDIR solutions are a design complement to many safety solutions. These FDIR design solutions are coordinated with the system-level safety team. During the design phase, the software contribution to the system hazards and fault-tree events are further refined through:

- The degree of control that the software exercises over the systems safety critical functions.
- The complexity of the software. From autonomous control of a hazard to timed intervention to engineering data evaluation, to human in the loop, greater complexity dramatically increases the chances of errors. Refer to NASA-GB-8719.13, NASA Software Safety Guidebook.
- The timing criticality of hazardous control actions.
- The likelihood a hazard will occur.
- The worst possible effect of the hazard (severity).

The provider shall uniquely tag the parts of the software design that implements safety design features and methods (e.g., inhibits, fault management, interlocks, assertions, and partitions) (Requirement). [SSS-043] This allows the acquirer and provider insight into the traceability and design aspects of the safety critical requirements and assures traceability to focused testing of the safety design features as well as alerting developers/maintainers that any changes and updates will impact safety. Rechecking and applying the Software Safety Litmus Test (Part 1 of the SSCA) in Appendix A, should be performed.

The provider SMA shall analyze the software design to: [SSS-044]

a. Ensure that the design is appropriately tagged as safety critical.
b. Ensure that the design's correct implementation of safety controls or processes does not compromise other system safety features.
c. Ensure that additional system hazards, causes, or contributions discovered during the software design analysis are documented in the required system safety documentation.
d. Ensure safety review approved controls, mitigations, inhibits, and safety design features are incorporated into the design.
e. Ensure the design maintains the system in a safe state during all modes of operation or can transition to a safe state when and if necessary.
f. Ensure that any partitioning or isolation methods used in the design to logically isolate the safety critical design elements from those that are non-safety critical are effective. This is particularly important with the incorporation of COTS or integration of legacy, heritage, and reuse software. Any software that can write or provide data to safety critical software will also be considered safety critical unless there is isolation built in, and then the isolation design is considered safety critical. Check Appendix A, Software Safety Litmus Test.

The provider SMA will also evaluate the following as they apply (see Appendix E of this Standard for a more complete list of potential software contributors to faults and failures):

- timing constraints
- hardware failures
- common-mode failures
- fault migration
- communications
- interrupts
- concurrency
- critical data algorithms & integrity
- event sequence
- memory/data storage
- FDIR design/fault tolerance
- adverse environments
- invalid inputs
- off-the-shelf or reused software
- design assumptions
- information flow
- mode and state transitions
- transition to safe mode
- single failure points

Safety analyses, such as Preliminary Hazard Analysis (PHA), sub-system hazard analyses, Failure Modes and Effects Analysis (FMEA), Fault Tree Analysis (FTA), can be used to help determine design features which can prevent, mitigate or control failures and faults. The level of failure or fault combinations needs to be evaluated based on system level requirements and software's role in meeting them to determine which design features are best included (e.g., both a software and a hardware failure, or multiple concurrent hardware failures need to be considered).

Based on the software design analysis, the provider shall update data products such as system safety analyses, requirements document, and design document with the identified software-related hazards, conditions and events, along with the associated controls or inhibits. [SSS-045] The design analysis may identify updates to SSCA. Software and system design analyses need to be performed in time to influence the design if further or different safety features are needed. Not all system hazard contributions or events are deemed necessary to have a control or mitigation if the severity or likelihood is low enough. Then it is a matter of assuring that those software hazard conditions really do fall within acceptable risk for not being addressed within software or elsewhere within the system.

From the software design analysis, the provider SMA shall assure and concur that the identified software-related hazards, conditions and events, are complete and the associated controls or inhibits are in the updated data products (Requirement). [SSS-046] The data products can include system safety analyses, requirements document, and design document. Depending on the assigned roles for a project, program, contract, contractor, or Center, SMA may or may not be creating and maintaining the hazard reports, safety data packages, or other safety data products. The same is true about who performs the design analyses. Whether SMA or engineering is the owner or developer, concurrence (or non-concurrence) must be achieved and documented between engineering and SMA. Proposed design controls, mitigations and inhibits are usually the purview of engineering, but SMA can certainly propose them too.

The provider software safety design analysis will be available to the acquirer and the acquirer SMA at project formal reviews, system-level safety reviews, and upon acquirer request. Refer to section 6.3 of this Standard.

7.4 Software Safety Implementation and Analysis

Software implementation will include all software safety design features and methods. For traceability purposes, the provider identifies or tags the safety critical code and data. Refer to NPR 7150.2 (requirements SWE-060, SWE-061, and SWE-064 in Revision A). The provider SMA is responsible for analyzing the method of implementation and identifying any improperly implemented safety features.

The provider shall uniquely identify or tag safety critical code (Requirement). [SSS-047] This allows the trace to be complete; from hazards, conditions, and events, through requirements, through design, to code and testing and verification.

The provider, with the provider SMA, shall perform a code analysis on the safety critical code (Requirement). [SSS-048] The code analysis includes:

a. Verification that the safety critical software code and data meet the project coding standards (NPR 7150.2, (requirement SWE-061 in Revision A)) and, that the safety critical code and data are identified as safety critical. The use of industry coding standards for critical code is recommended. Coding standard recommendations include: FAA or DOD approved standards for commercial or military aircraft (JSF++ coding standards), Motor Industry Software Reliability Association (MISRA) approved standards, and JPL Institutional Coding Standard for the C Programming Language.

b. Verification that software safety design features and methods are correctly implemented in the software code.
c. Verification that the code implementation does not compromise any safety controls or processes, does not create any additional hazard contributions, and maintains the system in a safe state during all modes of operation.
d. Verification that the code substantiates the software safety requirements.
e. Verification all safety critical code units are traceable to safety critical design elements.
f. Verification all safety critical code is traced and documented.

Based on the code analysis (referenced above), the provider shall update in the appropriate data products such as system safety analysis, requirements document, design document and verification document, any new and any changed software-related hazards, conditions, or events, along with the associated controls, mitigations, or inhibits (Requirement). [SSS-049] This includes implementation of safety functions, critical commands and data, or safety features as well as possible additional, undocumented requirements and or design features that may contribute to an unacceptable decrease in safe operations. Certain implementations may render a safety design feature unusable, or unusable in certain required modes or states. It could even contribute to a fault or failure by impacting timing, not cleaning up memory or contribution to a logic fault. See NASA-GD-8719.13, NASA Software Safety Guidebook and the Appendices D, E, and F of this standard for further examples.

Based on the code analysis (referenced above), the provider SMA shall concur with any new and any changed software-related hazards, conditions, or events, along with the associated controls, mitigations, or inhibits in updated data products (Requirement). [SSS-050] Data products can include system safety analysis, requirements document, design document and verification document. Depending on the assigned roles for a project, program, contractor, contract, or Center, SMA may or may not be creating and maintaining the hazard reports, safety data packages, or other safety data products. Whether SMA or engineering is the owner or developer, concurrence must be achieved and documented between engineering and SMA. Proposed implementation of safety controls, mitigations and inhibits are usually the purview of engineering, but SMA can certainly propose them too. Also, engineering or SMA can provide the analyses, in either case, one must check and concur with the other to assure safety is properly implemented.

The provider will make the software safety code analysis results available to the acquirer at project formal reviews, system-level safety reviews, at acquirer request, or for audits. Refer to section 6.3 of this Standard.

7.5 Software Safety Verification & Validation

Software verification and validation is a subset of system verification and validation.

Precedent for Verification and Validation Processes (see section referenced for more info)		
Software Verification and Validation	7.5	Procedures and results.

V&V by Test	7.5.1	Preferred. Complete safety verifications by method of test. If not possible, then perform 7.5.2.
V&V by Analysis, Demo, or Inspection	7.5.2	Pre-approval required. Only done if V&V by test is not possible.
V&V Results	7.5.3	Results analysis perform regardless of method of V&V.
Regression Testing	7.5.4	Performed as needed to ensure changes are accurately verified.

Software safety verifications includes testing at the unit level and component level, as well as integration and acceptance testing. Safety verification and validation must be performed at the level at which the safety features, controls, mitigations and inhibits can be thoroughly verified for off-nominal as well as nominal operation. Requirements and hazards traceability performed throughout the life-cycle is carried into the testing documentation. The provider SMA will analyze the methods of verification and document and report any improperly implemented safety features. It is important to note that software safety verification and functional verification are not the same. Functional verification is used to assure that requirements have been met. While meeting requirements is important, safety verification takes this a step further. Software safety verification uses information from the system safety analysis to identify the controls, inhibits and mitigations that must be verified to work as needed. The safety verifications must be officially recorded and the test procedures, test software, test hardware, results and artifacts must be configuration managed often at even the unit test level. To assure that the risk has been reduced, software safety verification requires aggressively exploring the potential for a hazard using realistic conditions and scenarios, including abnormal and off-nominal conditions. Any safety controls or mitigations are tested and verified to work as expected as well as the software's ability to take the system to a safe state in the presence of unexpected or irresolvable hazardous conditions.

The provider SMA shall concur on all program, project, and facility software safety related test plans and procedures (this includes regression testing) (Requirement). [SSS-051] The provider will ensure all safety checks are included and complete. This includes test procedures to verify the safe operation of software under nominal and off-nominal conditions and for regression testing. Actual testing of safety features, commands and data, controls, mitigations and inhibits is usually the purview of engineering, but SMA needs to assure they are planned for and are adequately included in test procedures.

The provider shall include any software conditions that can impact system performance in the system verification testing (Requirement). [SSS-052] This includes software nominal and off-nominal conditions, software limitations, operator inputs, and known software faults and failures.

If a model, simulator or emulator is used to test our critical systems (hardware, software or a combination) it is necessary to make sure our tests and associated models and simulators are of sufficient rigor, accuracy and depth to reliably supply results that are trusted and true. Analyses of where a test, model or simulator might provide false reliance on results needs to be carried out and any weakness corrected. Many NASA systems must rely on simulations and models to test for conditions and environments often unattainable in the lab or test bench and where use of actual flight hardware is not acceptable. It is important that an analysis of the risks to critical software tests, models and simulators be performed and the resultant level of safety and software engineering rigor be applied as for any safety critical software.

The provider SMA shall concur with all software safety verifications results (Requirement). [SSS-053] While witnessing and signing off on all software safety verifications is preferred, with prior concurrence from the acquirer SMA, the provider SMA can validate the results, as opposed to witnessing the software safety verifications. An example would be software safety testing that takes hours or days to run a procedure or an automated test script where individual test points are not provided. The results could be verified and validated when the "run" completes.

The provider implements and meets section 7.5.1 unless the requirements cannot be verified and validated by test; then the provider implements and meets section 7.5.2.

7.5.1 Verification and Validation by Test

The provider shall verify software safety requirements by the method of test (Requirement). [SSS-054]

The provider shall perform software safety testing to verify that system hazards or fault-tree events and risks related to software have been eliminated or controlled (Requirement). [SSS-055] This includes verification and validation that the software-implemented controls and mitigations are in-place and working.

The provider shall include software safety testing in unit level tests or component level tests that are testing safety aspects of the software (Requirement). [SSS-056] When it is necessary to test software safety features, controls, inhibits, mitigations, data and command exchanges and execution at the unit or component level in order to verify their functionality, then the provider shall document and control this level of testing including documenting test procedures, test runs and results, and SMA verification and concurrence.

Unit level testing is often the only place where the software paths can be completely checked for both the full range of expected inputs and its response to wrong, out of sequence, or garbled input. The stubs and drivers, test suites, test data, models and simulators used for unit testing are very important to capture and maintain for future regression testing and the proof of thorough safety testing. The reports of unit level testing of safety critical software components need to be thoroughly documented as well. All results must be configuration managed.

The provider software safety testing will include verification and validation that the implemented fault and failure modes detection and recovery works as derived from the safety analyses, such as PHAs, sub-system hazard analyses, failure-modes-effects-analysis, fault-tree-analysis. This can include both software and hardware failures, interface failures, or multiple concurrent hardware failures. Fault detection, isolation and recovery (FDIR) is often used in place of failure detection when using software as software can detect and react to faults before they become failures. Refer to NPR 7150.2 (requirement SWE-134 in Revision A).

The provider shall verify and validate, through software safety testing, the correct and safe operation of the software under load, stress, and off-nominal conditions including the operation of software controls and mitigations (Requirement). [SSS-057] This includes testing of time sensitive responses and events and assuring the software takes the system to a safe state in the event of unknown conditions and inputs.

The provider shall verify, through software safety testing, correct and safe operations in known operational and nominal configurations as well as the ability to transition to a safe state (Requirement). [SSS-058]

The provider will document the test results of all software safety verifications according to the test plan or safety plan. Refer to NPR 7150.2 (requirements SWE-065,SWE-066, and SWE-068 in Revision A).

When changes occur within software units or components containing safety critical requirements, these test articles (simulator, test drivers, and stubs) may be used to conduct regression tests. Thus it is important that the test procedures are clearly documented and configuration managed. It also helps if the test procedures are automated and analyzed to assure complete safety coverage.

7.5.2 Verification and Validation by Analysis, Demonstration, or Inspection

For software safety requirements that cannot be verified and validated by test, the provider shall verify these requirements by analysis, demonstration, or inspection with provider SMA and acquirer SMA concurrence (Requirement). [SSS-059]

The methodology chosen and the rationale for selecting analysis or demonstration will be recorded in the software test or software verification and validation plan. Refer to NPR 7150.2, (requirements SWE-066, SWE-067, and SWE-104 in Revision A). Depending on the size and complexity, the methodology and rationale can also be documented in a separate software safety test procedure. Software formal inspections are a recommended verification and validation process for safety requirements, design, test procedures, plans, and code analyses and should be considered throughout the development lifecycle and analyses of changes in conjunction with formal reviews and testing.

7.5.3 Verification and Validation Results

The provider SMA analyzes the results from the software and system safety test process, or the requirements verification evaluation, inspection, or demonstration process as part of the overall test analysis defined per NPR 7150.2 (requirement SWE-068 in Revision A).

The provider SMA shall ensure that any newly identified software contributions to hazards, event, or conditions found during testing are updated in the system safety data package (Requirement). [SSS-060]

Improper software safety implementation will be entered into the problem reporting system by the provider for resolution and eventual retest.

Incomplete or ineffective test procedures can also be managed through the provider problem reporting and resolution process. The provider SMA checks the updated test procedures for completeness and approves the procedures prior to testing.

The provider's software safety test analysis will be available to the acquirer and the acquirer SMA at project formal reviews and system-level safety reviews and upon request. Refer to Section 6.3 of this Standard.

7.5.4 Regression testing

Regression testing (refer to SSS-051) can vary from a minimal retest of a few units to end-to-end testing. From a safety perspective, it is important that any changes made to the software be regression tested. Changes to the hardware will require hardware and software integration regression testing. From a safety perspective, regression testing needs to assure that any safety controls and mitigations, continue to work as required and that no additional hazard contributions have occurred as a result of the changes. The provider SMA needs to review the proposed regression test coverage and procedures to assure the safety features are adequately covered. Questions to consider:

- Do transitions between modes and states impact safety?
- Are changes to parameters, sequence or timing going to change hazard controls, computing algorithms, and safety commands?

Configuration managed software test suites, created during unit and CSCI testing to test the safety features such as controls, FDIR, and fault and failure recovery, will help make regression testing smoother and better assure safety coverage.

SMA focuses on assurance that end-to-end functionality and known hazard controls still work.

7.6 Software Safety Acceptance

Safety critical software is considered accepted when it meets the requirements in this Standard and any additional contract, MOA, or MOU software safety requirements. Waivers and deviations to the safety requirements in this Standard are addressed in section 6.4 of this Standard. Software safety acceptance can be rolled into the system acceptance as long as the software safety features and ability to withstand faults and failures have all been proven at the appropriate level. "Certification" of safety critical software is dependent on the functionality and center implementation. Refer to individual center implementation of "Certification" software processes.

The provider shall collect the objective evidence (i.e., artifacts) that verifies the requirements within this Standard, the provider unique software safety requirements, safety features incorporated into the design, and any additional software safety requirements imposed by the contract, MOU, or MOA have been met (Requirement). [SSS-061] The provider SMA verifies the provider's objective evidence prior to submittal to the acquirer and the acquirer SMA.

Evidence of the following are evaluated as part of the acceptance process:

a. All software contributions to hazards, conditions, or events have been identified.

b. All controls for hazards, conditions, or events that require software implementation have been identified and properly implemented.
c. All software safety requirements and elements have been identified and tracked.
d. All software safety requirements and elements have been successfully implemented and validated, or waivers and deviations have been approved.
e. All software safety requirements and elements have been properly verified, or waivers and deviations have been approved.
f. All discrepancies in safety critical software have been dispositioned with the provider SMA concurrence.
g. All operational workarounds associated with discrepancies in safety critical software have the concurrence of the provider SMA.

The provider shall obtain approval from the acquirer and the acquirer SMA on the provider's evidence of acceptance (Requirement). [SSS-062] As per NPR 8715.3 paragraph 2.5.4, this is not just an acceptance deliverable, but "Safety, like other performance attributes, is monitored during the entire life cycle to ensure that an acceptable level of safety is maintained. Project managers ensure that the performance attributes and precursors that are identified as being important indicators of system safety are monitored" (NPR 8715.3).

7.7 Software Safety Maintenance and Operations

Software safety activities continue throughout the operational life of the system until its eventual retirement and archival. Software in safety critical systems often is subject to "routine" updates and reconfigurations. A key point to remember is that just because a change is of a routine nature, does not mean it is excused from the same requirements as all other changes made to the software.

Part of the on-going SMA activity includes safety evaluations on software changes. Changes to safety critical software require the evaluation and approval by the acquirer SMA and the provider SMA. The requirements of Section 6.9.1 – Software Change Requests within this Standard, apply during this lifecycle.

When a safety critical error is found during operations or if something goes awry, it is recommended that a root cause analysis be performed on how and why this error occurred. This includes an examination of the operational environment and its intended usage.

The requirements of this Standard will continue to be applicable, as defined in the program, project, or facility software safety plan, after the safety critical software, including changes, has been released for operations as defined in the program, project, or facility software safety plan. If maintenance and operations is performed by an organization or contract other than the developing organization or contract, then the acquirer will ensure the requirements of this Standard are invoked on the maintenance and operations contract and may develop a new software safety plan. It may be necessary to transition test procedures, artifacts, test suites, stubs and drivers, models or simulations used to test safety to the new contract. Refer to section 7.1.1 of this Standard.

The provider SMA responsible for operations and maintenance will perform software safety change analysis to evaluate whether the proposed change could invoke a hazardous state, affect a control for a hazard, condition, or state, increase the likelihood or severity of a hazardous state, adversely affect safety critical software, or change the safety criticality of an existing software element. It needs to be kept in mind that changes to the hardware or the software can impact the overall system's safety and while the focus is on changes to software, software also needs to be aware of changes to hardware that may impact how software controls, monitors and analyzes inputs from that hardware. Hardware and software changes can alter the role of software from non-safety critical to safety critical or change the severity from moderate to critical. Refer to section 6.9 of this Standard.

The provider SMA responsible for operations and maintenance analysis activities shall include [SSS-063]:

a. a pre-test assessment of proposed regression testing needed to ensure that the implementation of new software requirements or fixes or patches have not added new safety critical contributions or controls to hazard, conditions, and states as a result of the implementation strategies. Refer to section 7.5. 4 for additional information on regression testing.
b. a post-test assessment of regression testing to ensure that the implementation of new software requirements or fixes or patches have not added new contributions or controls to hazard, conditions, and states as a result of the implementation strategies.

The provider SMA responsible for operations and maintenance will concur on any changes to basic, as built, or approved upgrades of the operational software per NPR 7150.2 section 4.1.

Operational documentation, including user manuals and procedures, will describe all safety related commands, data, input sequences, options, error messages, corrective actions, and other items necessary for the safe operation of the system which software implements.

The provider SMA responsible for operations and maintenance shall evaluate user manuals and procedures (including updates) for safety-related commands, data, input sequences, options, error messages, corrective actions, and other items implemented in software which are necessary for the safe operation of the system, and for any safety impacts (Requirement). [SSS-064] This will ensure that any software-related hazard closures that depend on operational workarounds are properly documented.

Another consideration is repurposing a system with software. If a system's primary purpose is altered at the end of its primary mission, a reevaluation of the safety criticality is needed and any updates to the software needed to alter the functionality need to be assessed for safety criticality.

7.8 Software Safety Retirement

The requirements of this Standard expire for a particular program, project, or facility only upon the completion of the retirement of that program, project, or facility.

The provider shall address the following software impacts in the program, project, or facility maintenance and retirement plan: the safe termination of operations, collection and archiving of data, decommissioning of the system, writing lessons learned, and the retirement of that program, project, or facility (Requirement). [SSS-065] Both the acquirer and the acquirer SMA concur with the provider's retirement plan. This plan can be independent or part of another plan. There are times when the software may be completely replaced or completely or partially remissioned. These situations are included in this requirement. Also see NPR 7150.2 (requirements SWE-075, SWE-076, and SWE-105 in Revision A).

The acquirer SMA and the provider SMA shall concur on how the program, project, or facility retirement plan addresses software (Requirement). [SSS-066]

APPENDIX A: SOFTWARE SAFETY CRITICALITY ASSESSMENT

A.1 Purpose

The purpose of this appendix is to explain the use of and how to perform Software Safety Criticality Assessments. The use of this appendix is required by requirements SSS-001, 005, 018 and 024 of this standard.

A.2 Software Safety Criticality Assessment (SSCA)

The Software Safety Criticality Assessment includes two parts. Performing the Software Safety Litmus Test is part 1of the Software Safety Criticality Assessment. The Software Safety Litmus Test is used by the acquirer SMA and the provider SMA in determining if the program, project, or facility software is safety critical. The Software Risk Assessment, part 2, assesses the amount of critical software within a system, the level of autonomy of that software, the time to criticality, as well as the severity of failure.

SSCA Part 1: Software Safety Litmus Test	Required for all SSCA. All software must use the Software Safety Litmus Test in Appendix A. The results of the analyses must be documented and kept. *This Standard does not apply to software that does not pass the SW Safety Litmus Test, however, the test needs to be reapplied as the project changes and the software matures, especially through the design phases.*
SSCA Part 2: Software Risk Assessment	Performed with each update as the safety critical software matures. Recommend to be part of input to milestone reviews. Not required for initial SSCA performance or if software under consideration did NOT pass the SW Safety Litmus Test and is thus not safety critical. *Refer to SSS-001.* *Performed to determine the level of risk the software in whole or in part (e.g. CSCI, or to CSC) poses to the system and thus better determine level of effort. Based on the highest potential hazard that software contributes to, and or controls and mitigates, as well as the software control level and likelihood of occurrence, the analyses provides an overall level of risk to address software at the lowest possible known software configuration unit for the phase of the project. [e.g. Thus, if at the requirements phase, the total software is identified to have 4 potential contributions to various degrees of severity hazard, the highest severity level is the determining risk level for the software. Later, at SW PDR, it may be found that 2 out of 7 CSCIs contain the safety critical risks, and then a risk level would be assessed for each of the 2 CSCI's.] Each assessment and its results must be documented and maintained.*
Applicability Matrix	Based on results of SSCA Part 1 and Part 2. Provides "pre-tailoring" of requirements based on risk. *Refer to Appendix A.*

SSCA Form SSCA Form.docx	The results of the SSCA are required to be documented and approved. The use of this form is preferred but optional. Note: This form is provided in Word format so project/program/facility acquirers/providers can use it as is or adapt it to their individual needs.

A.3 Part 1 Software Safety Litmus Test

The Software Safety Litmus Test is initially performed during the formulation phase by the acquirer SMA through a series of investigative questions to the program/project/facility and the use of a limited set of well-supported and documented assumptions. Past, similar projects and basic analysis of needed software functionality provide additional inputs to assessing the safety criticality of the software early in the conception phase. This initial assessment is helpful in obtaining needed software safety resources and in writing a successful preliminary Statement of Work, and getting this standard in the RFP and on any contract/MOU/MOA. The Software Safety Litmus Test is just the first, high-level indicator of software's involvement in system safety. Software safety risk assessment as per part 2 of the SSCA, is then needed to determine the actual level of software safety effort. The SSCA is re-performed using the insight gained from safety analysis such as preliminary hazard analysis, system safety data packages, and reliability assessments. As the program/project/facility matures, re-performance of the whole SSCA takes place periodically, usually at milestones and when program/project/ facility/system changes that have the potential to impact the safety criticality status are made.

Software Safety Litmus Test: Software is classified as safety critical (see below for note A-1.0 and A-1.4) if it meets at least one of the following criteria (as per requirement SSS 001):

a. Causes or contributes to a system hazard/condition/event. (see below for note A-1.1 for applicability based on risk)
b. Provides control or mitigation for a system hazards/condition/event (see below for note A-1.1 for applicability based on risk)
   (1) Controls safety critical functions.
   (2) Mitigates damage if a hazard/condition/event occurs.
   (3) Detects, reports, and/or takes corrective action, if the system reaches a potentially hazardous state.
c. Processes safety critical commands (including autonomous commanding)
d. Resides on the same processor as safety critical software and is not logically separated from the safety critical software. (see below for note A-1.2)
e. Processes data or analyzes trends that lead directly to safety decisions (e.g., determining when to turn power off to a wind tunnel to prevent system destruction). (see below for note A-1.3)
f. Provides full or partial verification or validation of safety critical systems, including hardware or software subsystems. (e.g., this can include models and simulations) (see below for note A-1.4)

Note A-1.0: It is expected that an initial, high level systems evaluation of criticality is made early in the project lifecycle based on functions and environment. Later, detailed safety analyses starting with a preliminary hazard analysis and going through sub-system safety/hazard analysis should trigger subsequent evaluations of the related software components. This includes evaluating software only systems such as a database. For example, a database that holds and maintains flight parameters.

Note A-1.1: Tailoring/applicability is allowed based on the software risk assessment, marginal to high. Refer to Appendix A-Part 2 of this Standard.

Note A-1.2: Methods to separate the safety critical software from software that is not safety critical, such as partitioning, may be used. If such software methods are used to separate the code and are verified, then the software used in the isolation method is safety critical, and the rest of the software is not safety critical.

Note A-1.3: If data is used to make safety decisions (either by a human or the system), then the data is safety critical, as is all the software that acquires, alters the data, stores, and transmits the data. However, data that may provide safety information but is not required for safety or hazard control (such as engineering/science telemetry) is not safety critical.

Note A-1.4: The NPR 8715.3C NASA General Safety Program Requirements specifies the methodology for determining whether the overall system, on which the software is residing, is safety critical. Refer to NASA-STD-7009 NASA Standard for Models and Simulations for the detailed requirements for models and simulations. NASA relies heavily on our models, simulators and emulators to test our critical systems under conditions and environments often unattainable in the lab or test bench. NASA must make sure its tests and associated models and simulators are of sufficient rigor, accuracy and depth to reliably supply results that are trusted and true. Analyses of where a test, model or simulator might provide false reliance on results needs to be carried out and any weakness corrected.

A.4 Part 2 Software Risk Assessment and Applicability Matrix.

The software risk assessment uses information from the system safety analysis, if available, such as likelihood and severity of failure. Level of autonomy, system complexity, software size, hardware/software trade-offs are also used to evaluate risk.

The Applicability Matrix defines the minimum set of software safety requirements a program/project/facility must meet based on software risk assessment.

The software risk assessment is performed using the following steps:

a. The software has already been determined to be safety critical. (Refer to Appendix A Part 1 of this Standard)
b. Determine the software control category based on the software's degree of control of the system, severity and likelihood of failure, complexity, and level of autonomy using SSS Table A-1. (For additional guidance refer to Table 3-1 and Table 3-2 of NASA-GB-8719.13 NASA Software Safety Guidebook.)

   The degree of control, complexity, and autonomy are important factors in determining risk, these are not the only factors affecting risk in systems utilizing software. Other measures and characteristics that should be considered when evaluating risk include the following:

(1) Stability and maturity of the software
(2) Common cause failure
(3) Time to criticality, the time needed to respond to a fault or error before it becomes a failure
(4) Use of unproven technologies
(5) Level of failure tolerance
(6) Processes for analyzing and assuring reused and COTS software
(7) The robustness of the software engineering and software assurance processes
(8) The assessment process for replacing hardware functionality with software, if applicable
(9) Degree of human-system interactions and training associated with those interactions

Software Control Category IV mostly seems like it would not have any safety involved; however, it could include: e.g. test scripts for testing safety critical systems, programs that store and retrieve safety critical data or commands for later evaluation or manipulation, software that creates algorithms for safety critical data used by other software or simulators, models and emulators.

SSS Table A-1—Software Control Categories

Software Control Categories	**Descriptions** *(Software meets one or all of the below statements with the categories. Select the best fit and highest category of the software within the defined system/subsystem. This can be analyzed at a subsystem level.)*
IA *	Partial or total autonomous control of safety-critical functions by software.
	Complex system with multiple subsystems, interacting parallel processors, or multiple interfaces.
	Some or all safety-critical software functions are time critical exceeding response time of other systems or human operator.
	Failure of the software, or a failure to prevent an event, leads directly to a hazard's occurrence.
IIA&IIB *	Control of hazard by software but other safety systems can partially mitigate. Software detects hazards, notifies system of need for safety actions.
	Moderately complex with few subsystems and/or a few interfaces, no parallel processing.
	Some hazard control actions may be time critical but do not exceed time needed for adequate human operator or automated system response.
	Software failures will allow, or fail to prevent, the hazard's occurrence.
IIIA&IIIB*	Several non-software mitigating systems prevent hazard if software malfunctions.

	Redundant and independent sources of safety-critical information.
	Somewhat complex system, limited number of interfaces.
	Mitigating systems can respond within any time critical period.
	Software issues commands over potentially hazardous hardware systems, subsystems or components requiring human action to complete the control function.
IV	No control over hazardous hardware. No safety-critical data generated for a human operator.
	Simple system with only 2-3 subsystems, limited number of interfaces.
	Not time-critical.

* A = software control of hazard/condition/event. B = Software generates safety data for the system.

c. Determine the software control risk index based on the software control category and the hazard category using SSS Table A-2. (For additional guidance refer to Table 3-3 and Table 3-4 of NASA-GB-8719.13 NASA Software Safety Guidebook.)

The Software Risk Matrix is established using the hazard/condition/event categories for the columns and the Software Control Categories (SSS Table A-1) for the rows. A Software Control Risk Index is assigned to each element of the matrix.

A low index number from the Software Risk Matrix does not mean that a design is unacceptable. Rather, it indicates that greater resources need to be applied to the analysis and testing of the software and its interaction with the system.

SSS Table A-2—Software Control Risk Index

Software Control Category	**Hazard/Condition/Event Category**			
	Negligible/Marginal	**Moderate**	**Critical**	**Catastrophic**
IA	3 (Moderate)	2 (Medium)	1 (High)	1 (High)
IIA & IIB	5 (Low)	3 (Moderate)	2 (Medium)	1 (High)
IIIA & IIIB	5 (Low)	5 (Low)	3 (Moderate)	2 (Medium)
IV	5 (Low)	5 (Low)	4 (Marginal)	3 (Moderate)

Note: * A = software control of hazard. B = Software generates safety data for human operator

1 = High Risk: Software controls catastrophic or critical hazards
2 = Medium Risk: Software control of catastrophic or critical hazards is reduced, but still significant.
3 = Moderate Risk: Software control of moderate hazards
4 = Marginal Risk: Software control of moderate to marginal hazards
5 = Low Risk: Negligible hazard or little software control

Another main factor that the system safety risk takes into consideration, and so must software, is the likelihood/probability that the hazard condition or event will occur. This is most often an educated guess, especially for software, and the ranges within each category of likelihood are defined per program, project or facility. Table A-3 shows one commonly used system risk index found in many hazard analyses documents.

Table A-3—System Hazard Prioritization – System Risk Index

Systems Severity Levels	**System Hazard Likelihood of Occurrence**				
	Improbable	Unlikely	Possible	Probable	Likely
Catastrophic	4	3	2	1	1
Critical	5	4	3	2	1
Moderate	6	5	4	3	2
Negligible	7	6	5	4	3

1 = Highest Priority (Highest System Risk), 7 = Lowest Priority (Lowest System Risk)

Eventually each systems hazard is assessed with a systems risk index and then the action to be taken is determined hazard by hazard. The initial overall systems hazard risk index as well as the designation assigned to any known systems hazards can be one of software's first clues into assessing the software risk as software will inherit all or part of the system risk level. Applying the software risk index to all the software of a program/project/facility, or once defined into CSCI's, on a CSCI by CSCI level, has to be done with the most critical potential software hazard as the determining risk index. That is why it is important to continue to refine and repeat the SSCA through the design phase. But for planning purposes, we have to consider the risk index predicted based on highest possible software hazard contributions, mitigations and control. Applying the Likelihood of Occurrence to the software risk levels determined above, results in the following table:

Table A-4—Software Safety Prioritization – Software Risk Index

Software Control Risk Index Levels	**Likelihood of Occurrence**				
	Improbable	Unlikely	Possible	Probable	Likely

SW 1 (High)	4	3	2	1	1
SW 2 (Medium)	5	4	3	2	1
SW 3 (Moderate)	6	5	4	3	2
SW 4 (Marginal)	7	6	5	4	3
SW 5 (Low)	7	7	6	5	4

1 = Highest Priority (Highest Software Risk), 7 = Lowest Priority (Lowest Software Risk)

So, what does software have control over, how critical is that control and what is the likelihood that something bad will happen due to software are all taken into consideration. In general, it is expected that for Low Software Safety Prioritization Risks (7-5s and Green in Table A-4 above) the main requirements for both acquirer and provider would be to perform the Software Safety Criticality Assessment, keep a record of it, and revisit the SSCA upon any changes to the software functionality or design to assure the software maintains its low risk status. For all High and Medium Software Safety Prioritization Risks (1 & 2s in the Table A-4 above) all of the requirements of this standard will apply as written with some leeway in how they are accomplished based on the size and use of the program/project/facility. The Moderate and some Marginal that fall in the 3's and 4's on the Table A-4 need the most guidance and project specific details to determine the best risk mitigations or acceptance of risk. Some pre-tailoring is provided in the Applicability Matrix below; however, it should be considered mostly as guidance, especially for the potential Medium and Moderate hazards of higher Likelihood, further criteria will be needed and adjustments considered beyond the minimum requirements put forth here, especially for potentially catastrophic or critical software hazards contributions, controls or mitigations. Just because there is a low likelihood of a software contribution to catastrophic or critical hazard does not mean that some level of discipline and precautions should not be taken.

Another possible way to map all these factors is given in the example combined software risk index below.

SSS Appendix A-2
Example combined so

d. The applicability of requirements in this Standard is determined through the use of the software risk index and the SSS Applicability Matrix included below (in Excel format).

SSS Applicability
Matrix.xlsx

e. Finally, document the SSS requirements that are applicable to the program/project/facility with references on how the SSS requirements will be implemented. Recommend the use of Appendix B-2. Refer to requirements SSS-031 and SSS-037 of this Standard.

NOTE: The provider SMA and the acquirer SMA have approval of the applicability through approval of the program/project/facility software safety plan. Refer to requirements SSS-035 and SSS-036.

NOTE: The software risk assessment is maintained throughout the lifecycle. If the risk level changes, the applicability of requirements will change and should be reflected in the provider's software safety plan.

NOTE: This Applicability Matrix does not define "how" the provider meets the individual requirements of this Standard.

APPENDIX B: REVISION B – C TRACE, AND REV. C COMPLIANCE

B.1 Purpose

This appendix includes a trace between revision B and revision C of this Standard; and a trace matrix for Revision C compliance

B.2 Trace from Revision B to Revision C

The Software Safety Standard trace from Revision B to Revision C Blank trace table is provided here in Excel format.

Trace from Rev B to
Rev C.xlsx

B.3 Trace Matrix for Revision C Compliance

The Software Safety Standard Revision C Blank trace table is provided here in Excel format to provide a standardized table for the acquirer and provider program/projects/facilities to ensure compliance to the requirements of this standard.

SSS Rev C
Matrix.xlsx

APPENDIX C—SOFTWARE AS PART OF THE SYSTEM SAFETY ANALYSIS

C.1 Purpose

The purpose of this appendix is to present additional information regarding software as part of the system safety analysis.

C.2 Software as Part of the System Safety Analysis

The system safety analyses and system reliability analyses identify specific software safety needs and controls for the program, project, or facility (i.e., system); previous or similar mission or facility configurations may also contribute information on software safety. System hazard analyses and system reliability analyses will be reviewed by software safety personnel to initially assess software's potential role and then, as the system development matures, ensure that changes and findings at the system level are incorporated into the software as necessary. The software safety analyses, which provide input back to the system safety activities, are a subset of the overall system hazard and reliability analyses and are not conducted in isolation. System and software safety analyses are performed iteratively over the life of the system as the system is better defined or changes are made. An effective system safety analysis effort requires a continuous exchange of information among all team members (including project management, system safety, system reliability, software assurance, software development, software safety, and the Center SMA organization) during the program, project, or facility life-cycle.

System safety analysis tools such as Preliminary Hazard Analyses, Functional Hazard Analyses, and Fault Tree Analyses identify potential system hazards/events/conditions and may identify which proposed subsystems contribute to, or are needed to control, those hazards/events/conditions. The system hazard analyses will point to areas where needed mitigations and controls lead to requirements and design decisions.

The software safety analyses and activities, and relationship with the system safety program, need to be planned early in the project life-cycle. Entrance and exit criteria for the software safety analyses is part of this planning and coordination effort. See the NASA Software Safety Guidebook, NASA-GB-8719.13, for more details on planning the software safety program and integrating it with other project activities.

APPENDIX D—SOFTWARE SAFETY TECHNICAL REQUIREMENTS

Below are additional detailed technical software safety requirements that can be imposed on the program/project/facility. They include general software safety requirements, and must-work and must-not-work software safety requirements to be met by any program/project/facility. These are in addition to NPR 7150.2 (requirement SWE-134 in Revision A) and were derived from both the Constellation and International Space Station Software Safety Requirements.

SSS Appendix D.docx

The Software Safety Technical Requirements are provided here in Word format to allow the program/projects/facilities to adapt for their individual needs.

APPENDIX E—SOFTWARE CONTRIBUTORS TO FAULTS, FAILURES, AND HAZARDS

When evaluating software contributions to faults, failures, and hazards, the following generic software defects should be considered:

SSS Appendix E.docx

The Software Contributors to Faults, Failures, and Hazards are provided here in Word format to allow the program/projects/facilities to adapt for their individual needs.

APPENDIX F—CONSIDERATIONS FOR COTS, TOOLS, AND FACILITY SOFTWARE SAFETY CONSIDERATIONS

F.1 Purpose

This appendix includes considerations for COTS software safety, a checklist for software tools, and facility safety considerations. One example is provided. It is expected that these checklists will be modified to meet the needs of a particular program, project, or facility.

F.2 COTS Software Safety Considerations*

SSS Appendix F-4
COTS Checklist.xlsx

SSS Appendix
F-1.docx

An example checklist is provided as well as an excerpt from *Wetherholt, Martha, "The Software Assurance of COTS Software Products," American Institute of Aeronautics and Astronautics.

F.3 Software Safety Checklist Items for Software Tools

F-2 SW Safety Cklist
for tools EXAMPLE.do

SSS Appendix F-2a
SW Safety Cklst for S

The Software Safety Checklist for Software Tools is provided here in Word format to allow the program/projects/facilities to utilize for their individual needs. An Example is also provided.

F.4 Facility Safety Considerations

SSS Appendix
F-3.docx

APPENDIX G: REFERENCES

G.1 Purpose

The purpose of this appendix is to provide guidance and is made available in the reference documents below. The following documents are considered to be useful as background information for the reader in understanding the subject matter but do not constitute requirements of the Standard.

G.2 Reference Documents

The following reference documents are cited within this Standard.

G.2.1 Government Documents

NPD 2810.1	NASA Information Security Policy
NPD 7120.4	NASA Engineering and Program/Project Management Policy
NPR 1441.1	NASA Records Retention Schedules
NPR 2810.1	Security of Information Technology
NPR 7120.5	NASA Program & Project Management Processes & Requirements
NPR 7123.1	NASA System Engineering Processes and Requirements
NPR 7150.2	NASA Software Engineering Requirements
NPR 8715.3	NASA General Safety Program Requirements
NASA-STD-7009	NASA Standard for Models and Simulations
NASA-STD-8709.20	Management of Safety and Mission Assurance Technical Authority (SMA TA) Requirements
NASA-STD-8709.22	Safety & Mission Assurance Acronyms, Abbreviations, & Definitions
NASA-STD-8739.8	NASA Software Assurance Standard
NASA-GB-8719.13	NASA Software Safety Guidebook
NASAHDBK8739.23	NASA Complex Electronics Handbook for Assurance Professionals
forthcoming	PLD Handbook

G.2.2 Non-Government Documents

DO-178C	Software Considerations in Airborne Systems and Equipment Certifications

G.3 Further Guidance

The following reference documents are recommended for further guidance.

G.3.1 Government Documents

NPD 8700.1	NASA Policy for Safety and Mission Assurance
NPR 8000.4	Agency Risk Management Procedural Requirements
NPR 8715.7	NASA Expendable Launch Vehicle Payload Safety Program
NASA/SP-2011-3422	NASA Risk Management Handbook
NASA/SP-2010-580	NASA System Safety Handbook Volume 1, System Safety Framework and Concepts for Implementation.

G.3.2 Non-Government Documents

IEEE 12207.2008	Systems and Software Engineering: Software life-cycle processes