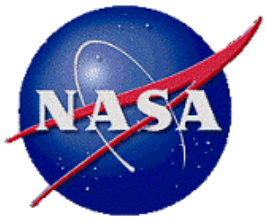

NOT MEASUREMENT SENSITIVE

National Aeronautics and Space Administration

NASA-STD-8719.13B w/Change 1
July 8, 2004

SOFTWARE SAFETY STANDARD

NASA TECHNICAL STANDARD

REPLACES NASA-STD-8719.13A DATED SEPTEMBER 1997

FOREWORD

The NASA Software Safety Standard (hereinafter referred to as "this Standard") is approved for use by NASA Headquarters and all NASA Centers and is intended to provide a common framework for consistent practices across NASA programs.

This Standard was developed by the NASA Office of Safety and Mission Assurance to provide the requirements for software safety across all NASA Centers, programs and facilities. It describes the activities necessary to ensure that safety is designed into the software that is acquired or developed by NASA. All Program/Project Managers, Area Safety Managers, IT managers, and other responsible managers are to assess the inherent safety risk of the software in their individual programs. The magnitude and depth of software safety activities should reflect the risk posed by the software while fulfilling the requirements of this Standard.

This Standard revises NASA-STD-8719.13A. Changes in software technology, software development methodology, and the field of computing necessitate updating this Standard on a regular basis. Requirements for new technology and methodology areas, such as commercial off-the-shelf software, software reuse, and security are included.

Comments and questions concerning the contents of this publication should be referred to the National Aeronautics and Space Administration Headquarters, Director, Safety and Assurance Requirements Management Division, Office of the Chief for Safety and Mission Assurance, Washington, DC 20546.

/s/

Bryan O'Connor

Chief of the Safety and Mission Assurance Office

RECORD OF CHANGES

Change No.	Date	Title or Brief Description	Entered By	Date Entered
1	7/28/2004	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	7/24/2004

CONTENTS

1 SCOPE

1.1 Purpose

This Standard specifies the software safety activities, data, and documentation necessary for the acquisition or development of software in a safety-critical system. Safety-critical systems that include software must be evaluated for software's contribution to the safety of the system during the concept phase, and prior to the start, or in the early phases, of the acquisition or planning for the given software. Unless the evaluation proves that the software is not involved in the system safety, this Standard is to be followed. See section 1.2 for guidance, and section 4.1 for requirements (and definition), on the determination of safety-critical software.

The purpose of this Standard is to provide requirements to implement a systematic approach to software safety as an integral part of the project's overall system safety program, software development, and software assurance processes. It describes the activities necessary to ensure that safety is designed into software that is acquired or developed by NASA and that safety is maintained throughout the software and system life cycle. How these requirements are met will vary with the program, project, facility, Mission, and Center. The NASA-GB-8719.13, Software Safety Guidebook, provides additional information on how to implement software safety and software safety related activities in a manner consistent with the software's role in system safety. The risk posed by safety-critical software will vary with the system safety criticality (e.g., type of hazard) and the level of control or influence the software has on system safety factors. While the requirements of this Standard cannot be tailored, the specific activities and depth of analyses needed to meet the requirements can, and should, be tailored to the software safety risk. That is, while the requirements must be met, the implementation and approach to meeting these requirements may and should vary to reflect the system to which they are applied. Substantial differences may exist when the same software safety requirements are applied to dissimilar projects. Appendix A shows how an example medium-sized project might meet the requirements of this Standard. A compliance matrix listing all of the requirements in this Standard along with the personnel roles and responsibilities required for each requirement, is available in Appendix B. This matrix can be used by the program, project, or facility as a checklist to ensure coverage of all requirements in the Standard

There are two kinds of safety requirements: process oriented 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 must include or implement (e.g., two-fault tolerance). Use of this Standard does not preclude the necessity to follow applicable technical standards.

Software safety activities occur within the context of system safety, system development, and software development and assurance. In an ideal system environment, information flows freely among all elements of the program/project, and concerns are appropriately addressed. Providing the needed information to the concerned parties in a timely manner is key to any successful exchange.

The requirements specified in this Standard will:

- 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 planning, management, and control activities.
- Ensure that software safety is considered throughout the system life cycle, including generation of requirements, design, coding, test, and operation of the software.
- Ensure that software acquisitions, whether off-the-shelf or contracted, have evaluated, assessed, and addressed the software for its safety contributions and limitations.
- Ensure that software verification activities include software safety verifications.
- Ensure that the proper certification requirements are in place and accomplished prior to the actual operational use of the software.
- During operational use of the software, ensure that all changes and reconfigurations of the software are analyzed for their impacts to system safety.

1.2 Applicability

This Standard applies to all safety-critical software acquired or developed by NASA. Section 4.1 (and section 3, Glossary) defines what software is considered safety-critical. Section 4.1 also details the "litmus test" that all projects must apply to their software, to determine if it is safety-critical and therefore subject to this Standard.

The NPR 8715.3 NASA Safety Manual specifies the methodology for determining whether a system is safety-critical. This software safety standard further defines whether the software in a safety-critical system is also safety-critical.

This Standard applies to software that resides in hardware (i.e., firmware). This Standard also applies to government furnished software, purchased software (including commercial-off-the-shelf (COTS) software), and any other reused software when included in a safety-critical NASA system. Safety-critical software can be found in all types of systems, including Flight, Ground Support, and Facilities.

If the system is already in development or is a legacy system, then the software within the system should be assessed for its contribution to the safety of the system. If the software is found to be safety-critical, a plan should be worked out with the safety personnel on how the system will or will not meet the requirements in this Standard. Legacy systems will be addressed on a case-by-case basis and the decisions should be documented. Systems in the maintenance and operation phase should at least have the safety requirements marked during the routine maintenance cycle.

In addition, COTS software cannot be ignored in safety-critical systems. The COTS software should be assessed before use and verified within the system it is contained to ensure the COTS cannot do something inadvertent to cause a hazard (see NASA-GB-8719.13 NASA Software Safety Guidebook).

A key factor to keep in mind when determining the applicability of this Standard is that the presence of non-software hazard controls or mitigations (e.g., operator intervention, hardware overrides) reduces, but does not normally eliminate, the software safety risk. Hence, the need for applying this Standard is not removed. The NASA Software Safety Guidebook, NASA-GB-8719.13, should be used to create a set of activities and analyses tailored to meet the requirements of this Standard.

This Standard does not discourage the use of software in safety-critical systems. When designed and implemented correctly, software is often the first, and sometimes the best, hazard detection and prevention mechanism in the system. Software can be used to prevent problems before they lead to hazardous conditions. This Standard provides requirements that will ensure that the safety-critical software receives the required levels of attention throughout the project life cycle.

1.3 Assumptions

Software covered by this Standard is to be developed following sound software engineering practices and in accordance with appropriate development standards and requirements.

Any software covered by this Standard is also be covered by the NASA-STD-8739.8 NASA Software Assurance Standard. Software safety is a discipline of the software assurance process, and it provides complementary activities to the other software assurance disciplines. This Standard stresses coordination between these disciplines, as well as with system safety and software development, to minimize duplication of effort.

All activities of this Standard are to be accomplished as an integral part of the overall management, engineering, and assurance activities for any safety-critical system containing software.

1.4 Guide to this Standard

1.4.1 *Requirements*

Requirements are designated with a number (e.g., 5.2.1) and contain a *shall* statement. These statements are the only sentences to be considered requirements. In some cases, additional explanatory information is added to clarify or add specific guidelines related to a requirement.

Many sections begin with an introductory paragraph(s) that describes in less formal terms, or at a higher level, the requirements embodied in the section. Additional information, such as good practices, may also be included in these paragraphs. These introductory paragraphs are to be considered as guidance and not as requirements.

1.4.2 *Software Safety Personnel*

The use of the terms software safety personnel, software safety engineer, and software safety manager are not used in this document as prescriptive, required personnel assignments. The authors recognize that one or more individuals with varying titles and additional responsibilities may fill these positions. There is no implication for required use of these titles within any organization. They are merely used herein to provide a consistent label for those with expertise in software and system safety who will be evaluating and performing the functions and procedures discussed in this Standard. This may be fulfilled by systems safety personnel with

strong software backgrounds, software engineers with safety exposure and systems safety guidance, or software assurance engineers with safety expertise.

Software expertise is needed for safety-critical systems with software. The safety of systems with software can be affected by the language, compiler, operating system, software development strategy, software design architecture, development tools, etc. Software safety must go beyond mere identification of systems hazards to possible software functions. For this, a certain expertise is needed as well as an understanding of the system and environment in which the software must operate. It would be great if several individuals contained all this knowledge, but this is not always possible. Thus a collaboration is usually needed between systems, systems safety and software to jointly determine software's safety contribution, the controls, design features, verifications, and requirements needed to assure the system is as safe as possible.

1.4.3 *Plans and Documents*

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, or facility must generate. The software safety information must be recorded within the documentation, but the exact type of documentation is left up to the program, project, or facility.

When specific plans are mentioned (e.g., the Software Safety Plan), they can be standalone documents or incorporated within other documents (e.g., system safety plan, a software management/development plan, or a software or system assurance plan).

.

2 RELATED DOCUMENTATION

2.1 Applicable Documents

Documents cited in this Standard are listed in this section.

2.1.1 *Government documents*

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NPD 8700.1	NASA Policy for Safety and Mission Assurance
NPD 2820.1	NASA Software Policies
NPR 8715.3	NASA Safety Manual
NPD 2810.1	Security of Information Technology
NPR 2810.1	Security of Information Technology
NASA-STD-8739.8	NASA Software Assurance Standard
NASA-GB-8719.13	NASA Software Safety Guidebook

2.1.2 *Non-government documents*

IEEE 1228	Standard for Software Safety Plans
ISO 8402	Quality Management and Quality Assurance – Vocabulary
IEEE 610.12	Standard Glossary of Software Engineering Terminology
RTCA DO-178B	Software Considerations in Airborne Systems and Equipment Certification

2.2 Reference Documents

Documents listed in this section are for reference only.

2.2.1 *Government documents*

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

NPR 7120.5	Program and Project Management Processes and Requirements
NPR 8000.4	Risk Management Procedures and Guidelines

2.2.2 *Non-government documents*

IEEE 12207.0	Standard for Information Technology: Software life cycle processes

DOD Joint Software System Safety Committee	Software System Safety Handbook

3 DEFINITIONS AND ACRONYMS

3.1 Definitions used in this Standard

Term	Definition
Accident	An unplanned event or series of events that results in death, injury, occupational illness, or damage to or loss of equipment, property, or damage to the environment; a mishap. [IEEE 1228]
Baseline	A specification or product that has been reviewed formally and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.
Black Box Testing	Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. [IEEE 610.12]
Certification	The process of formally verifying that a system, software subsystem, or computer program is capable of satisfying its specified requirements in an operational environment for a defined period of time. This includes any requirements for safing the system upon the occurrence of failures with potential safety impacts.
Component	A constituent element of a system or subsystem.
Customer	The NASA program, project, facility, or other entity that acquires software developed by another organization.
Decomposition	The process of breaking a system or component up into constituent parts. For requirements, the top-level requirements will be general, and lower-level (decomposed) requirements will be specific.
Deviation	A documented variance that authorizes departure from a particular safety requirement where the intent of the requirement is being met through alternate means that provide an equivalent level of safety. Deviations are only employed for variances identified prior to development. Deviations do not require a revision to documents defining the affected item.
Failure	Non-performance or incorrect performance of an intended function of a product. A failure is often the manifestation of one or more faults.
Failure Modes And Effects Analysis (FMEA)	A bottom-up systematic, inductive, methodical analysis performed to identify and document all identifiable failure modes at a prescribed level and to specify the resultant effect of the modes of failure.
Fault	An inherent defect in a product which may or may not ever manifest, such as a bug in software code.
Fault Tree Analysis (FTA)	An analytical technique, whereby an undesired system state is specified and the system is then analyzed in the context of its environment and operation to find all credible ways in which the undesired event can occur.

Term	Definition
Fault Detection, Isolation, And Recovery (FDIR)	**Detection:** The ability to discover faults; the process of determining that a fault has occurred. **Isolation:** The process of determining the location or source of a fault. **Recovery:** A process of overcoming a fault without permanent reconfiguration.
Firmware	The combination of a hardware device and computer instructions and/or computer data that reside as read-only software on the hardware device.
Functional Requirements	Functional requirements define what the system or subsystem must do to fulfill its mission, including timing and performance requirements. All requirements that will be expressed in the system, rather than in the process to create the system, are functional requirements.
Hazard	Existing or potential condition that can result in, or contribute to, a mishap or accident.
Hazard Control	Means of reducing the risk of exposure to a hazard. This includes design or operational features used to reduce the likelihood of occurrence of a hazardous effect or the severity of the hazard.
Hazard Mitigation	Any action that reduces or eliminates the risk from hazards.
Independent Verification And Validation (IV&V)	Verification and validation performed by an organization that is technically, managerially, and financially independent of the development organization. IV&V, as a part of Software Assurance, plays a role in the overall NASA software risk mitigation strategy applied throughout the life cycle, to improve the safety and quality of software systems.
Memorandum Of Agreement (MOA)	A written agreement between two or more parties that defines the roles and responsibilities of each party with respect to the collaborative efforts of a particular program/project. A MOA is sometimes called a Memorandum of Understanding (MOU).
Mission-Critical	Item or function that must retain its operational capability to assure no mission failure (i.e., for mission success).
Off-The-Shelf Software	Ready-made software used “as-is” within a system. • Commercial Off-the-shelf (COTS) software refers to purchased software such as operating systems, libraries, or applications. • MOTS (modified off-the-shelf) software is typically a COTS product whose source code can be modified. • GOTS (government off-the-shelf) software is typically developed by the technical staff of the government agency for which it is created.
Partitioning	Separation, physically and/or logically, of safety-critical functions from other functionality.

Term	Definition
Preliminary Hazard Analysis (PHA)	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.
Project Life Cycle	Steady progression of a project from its beginning to its completion and decommissioning. A set of steps or phases through which a project advances. This includes formulation/conception through sign-off and delivery to the customer and may include operations, maintenance and retirement depending on how the project is defined. The operations and maintenance phases through retirement may be a separate project life cycle and as such still needs to address the requirements in this Standard.
Regression Testing	The selective retesting of a system that has been modified to ensure that any defects have been fixed and that no other previously working functions have failed or ceased to work as expected as a result of the changes.
Reused Software	Software created for another system that is incorporated into the system under development.
Risk	The combination of (1) the probability (qualitative or quantitative) that a program or project will experience an undesired event and (2) the consequences, impact, or severity of the undesired event were it to occur.
Safety-Critical	Any condition, event, operation, process, equipment, or system that possesses the potential of directly or indirectly causing harm to humans, destruction of the system, damage to property external to the system, or damage to the environment.

Term	Definition
Safety-Critical Software	Software is safety-critical if it meets at least one of the following criteria: 1. Resides in a safety-critical system (as determined by a hazard analysis) AND at least one of the following: a. Causes or contributes to a hazard. b. Provides control or mitigation for hazards. c. Controls safety-critical functions. d. Processes safety-critical commands or data. e. Detects and reports, or takes corrective action, if the system reaches a specific hazardous state. f. Mitigates damage if a hazard occurs. g. Resides on the same system (processor) as safety-critical software. 2. 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). 3. Provides full or partial verification or validation of safety-critical systems, including hardware or software subsystems.
Safety And Mission Assurance (SMA)	SMA refers to the organization, i.e., the offices and people at all NASA Field Installations and Headquarters, who support customers with policy, process, and standards development; oversight and insight; and technology development and transfer, in the disciplines of safety, reliability, maintainability, and quality.
Safety Assurance	Ensuring that the requirements, design, implementation, verification and operating procedures for the identified software minimizes or eliminates the potential for hazardous conditions.
Software Acquisition	The process of obtaining software from another organization via a documented agreement; a set of activities that are used to acquire software products from another organization.
Software Assurance	The planned and systematic set of activities that ensure that software life cycle processes and products conform to requirements, standards, and procedures. [IEEE 610.12] For NASA this includes the disciplines of Software Quality (functions of Software Quality Engineering, Software Quality Assurance, Software Quality Control), Software Safety, Software Reliability, Software Verification and Validation, and IV&V.
Software Element	A portion of a software item that is logically discrete. The software element will depend on context, and can be a subset of the requirements, software design, software source code, or any software entity.

Term	Definition
Software Development Life Cycle	All activities required to analyze, define, develop, test, and deliver a software product. The development life cycle ends when the software becomes operational and is accepted formally for use by the customer and/or operations. Once operational, any changes/upgrades are to be treated as reduced scale software development lifecycles and the main activities (analyze, define, develop, test and deliver) should apply during these maintenance activities.
Software Hazard	A hazard caused by incorrect software control of hazardous hardware. The software might be functioning correctly (according to its requirements) or in a failure mode.
Software Life Cycle	The period of time that begins when a software product is conceived and ends when the software is no longer available for use. The software life cycle typically includes a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase. [IEEE 610.12] The software development life cycle is a subset of this larger life cycle.
Software Patch	A modification made directly to an object program without reassembling or recompiling from the source program. [IEEE 610.12]
Software Safety	The aspects of software engineering and software assurance that provide a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions (e.g., data and commands) to ensure safer software operation within a system.
Software Safety Analysis	The application of system safety engineering techniques throughout the software life cycle to ensure that errors that could reduce system safety have been eliminated or controlled to an acceptable level of risk.
Software Safety Change Analysis	An evaluation of whether a proposed change could invoke a hazardous state, affect a hazard control, increase the likelihood of a hazardous state, adversely affect safety-critical software, or change the safety-criticality of an existing software component. This activity determines the impact of changes made in assumptions, specifications, requirements, design, code, equipment, test plans, environment, user documentation, and training materials.
Software Safety Plan	A document that details the activities, general relative schedule of needed activities, communication paths and responsibilities for performing software safety activies as part of the systems safety program. This does not have to be a standalone document, but could be included as part of the systems safety plan or, for small projects, an overall assurance plan. While it may be written by either the project/program/facility or by the safety personnel within the Center SMA organization(s), both must sign off on it.

Term	Definition
Stub	A skeletal or special-purpose implementation of a software module, used to develop or test a module that calls or is otherwise dependent on it. [IEEE 610.12]
Subcontracting	An individual, partnership, corporation, or association that contracts with an organization (i.e., the prime contractor) to design, develop, test, verify, and/or manufacture one or more products.
System Hazard Analysis	Identification and evaluation of existing and potential hazards and the recommended mitigation for the hazard sources found. [NPR 8715.3] This includes the verification and validation of the safety functions and hazard controls.
System Life Cycle	All activities required to analyze, define, develop, test, deliver, operate, maintain, and retire a system. The software life cycle (and software development life cycle) is incorporated within the system life cycle.
Traceability	Ability to trace the history, application or location of an entity by means of recorded identifications. [ISO 8402, 3.16] For example, requirements traceability as applied to software safety involves identifying safety-critical requirements/functions then tracing them through design, test, acceptance, changes and upgrades, and through retirement.
Tracing System	A system that enables the traceability, in both the forward and backward directions, of the lineage of a requirement from its first level inception and subsequent refinement to its implementation in a product and the documentation associated with the product.
Validation	Confirmation by examination and provision of objective evidence that the particular requirements for a specified intended use are fulfilled. [IEEE 12207.0] Did we build the right system for the customer?
Variance	Documented and approved permission to perform some act or operation that is contrary to established requirements.
Verification	Confirmation by examination and provision of objective evidence that specified requirements have been fulfilled. [IEEE 12207.0] Was the system built right?
Waiver	A documented variance that authorizes departure from a particular safety requirement where alternate methods are employed to mitigate risk or where an increased level of risk has been accepted by management. Waivers are only employed for variances identified after beginning development or after an item has been submitted for inspection.

3.2 Acronyms used in this Standard

AOA	Annual Operating Agreement
CASE	Computer Aided Software Engineering
CCB	Change Control Board
CDR	Critical Design Review
CM	Configuration Management
COTS	Commercial Off-the-Shelf
Code Q	NASA Headquarters Office of Safety and Mission Assurance
CoFR	Certification of Flight Readiness
DCR	Design Certification Review
FACI	First Article Configuration Inspection
FAR	Federal Acquisition Regulations
FDIR	Fault Detection, Isolation, and Recovery
FMEA	Failure Modes and Effects Analysis
FTA	Fault Tree Analysis
GB	Guidebook
GOTS	Government Off-the-Shelf
IA	Independent Assessment
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Organization for Standardization
ISS	International Space Station
IV&V	Independent Verification and Validation
MOA	Memorandum of Agreement
MOTS	Modified Off-the-Shelf
MOU	Memorandum of Understanding
NASA	National Aeronautics and Space Administration
NPD	NASA Policy Directive
NPR	NASA Procedural Requirements
OS	Operating System
OTS	Off-the-Shelf
PAR	Preflight Acceptance Review
PDR	Preliminary Design Review
PHA	Preliminary Hazard Analysis
PM	Project Manager
PRACA	Problem Reporting and Corrective Action
PSRP	Payload Safety Review Panel
SAE	Software Assurance Engineer
SMA	Safety and Mission Assurance
SSAE	SMA Software Assurance Engineer
SSE	System Safety Engineer
STD	Standard
TRR	Test Readiness Review
UML	Unified Modeling Language

4 SAFETY-CRITICAL SOFTWARE DETERMINATION

4.1 Determination Process

As systems increase in complexity, software has become a much more important component in system design and operation. Software controls much of the equipment around us, including equipment and systems that can harm us. If that system can lead to injury, death, loss of major equipment, or damage to the environment, then software safety becomes vitally important.

When a system is determined to be safety-critical (e.g., through a preliminary hazard analysis), the use of software within that system must be analyzed. The key for the analysis at this stage is to look at the entire system, and see what roles the software has within it. One cannot just look at the software components in isolation, but rather look at all of the system components (including the end-user of the system). Software cannot be divorced from the system where it resides. Software safety analyses are performed first to determine if the software is safety-critical, and later to evaluate how well the software safety requirements are defined, designed, and implemented in the system..

Some systems are designed such that the hardware prevents the software from accessing the hazardous sub-system(s). In such a system, the software cannot be a hazard cause. However, the software may still be used in other ways (e.g. as a monitor that will shut down power to prevent a hazard) that are safety-critical. Therefore, the software still needs to be evaluated to determine if it does, or does not, meet any of the criteria for safety-critical software.

The software will be safety-critical if it meets the criteria of section 4.1.1.1. Once software is identified as safety-critical, the software acquisition, development, and operations processes are subject to the requirements in the following sections. The relationship of the safety-critical software to the total system safety effort is further discussed in section 4.2.

For projects that are beyond the concept phase at the time this Standard is invoked, the software needs to be evaluated for its contribution to, or impairment of, the safety of the system. The software safety personnel, the Center SMA, systems engineering and project/program/facility management should jointly develop a plan for assuring the software safety of the system, using this Standard to the maximum extent possible. If the system is currently under development, the plan should detail what retrospective activities will be performed (those that would otherwise have occurred, if the Standard was applied from the start) and what future activities and analyses will be performed. Systems that are complete (legacy) and operational need to be evaluated. However, retrospective activities from this Standard should only be applied based on the determined risk of the software. Changes and upgrades to operational software should follow the requirements of section 7.

4.1.1 When the system is determined to be safety-critical, the software shall be evaluated for its contribution to the safety of the system.

4.1.1.1 Until proven otherwise, based on the following evaluation criteria, all software within a safety critical system shall be assumed to be safety critical.

4.1.1.2 Software shall be classified as safety-critical if it meets at least one of the following criteria:

a. Resides in a safety-critical system (as determined by a hazard analysis) AND at least one of the following apply:

1) Causes or contributes to a hazard.

2) Provides control or mitigation for hazards.

3) Controls safety-critical functions.

4) Processes safety-critical commands or data (see note 4-1 below).

5) Detects and reports, or takes corrective action, if the system reaches a specific hazardous state.

6) Mitigates damage if a hazard occurs.

7) Resides on the same system (processor) as safety-critical software (see note 4-2 below).

b. 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).

c. Provides full or partial verification or validation of safety-critical systems, including hardware or software subsystems.

Note 4-1: 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, processes, and transmits the data. However, data that may provide safety information but is *not* required for safety or hazard control (such as engineering telemetry) is not safety-critical.

Note 4-2: Non-safety-critical software residing with safety-critical software is a concern because it may fail in such a way as to disable or impair the functioning of the safety-critical software. Methods to separate the code, such as partitioning, can be used to limit the software defined as safety-critical. If such methods are used, then the isolation method is safety-critical, but the isolated non-critical code is not.

4.1.1.3 The software evaluation shall occur during the concept or formulation phase, prior to the acquisition or planning for the given software for all new projects.

Note: In some situations, the formal software evaluation will be performed by the supplier of the software, and may occur in a later project phase. This should be noted in an appropriate project plan and agreed to by a program/project/facility safety engineer or Center SMA representative. However, the program/project/facility management should still evaluate the system for the potential for safety-critical software before beginning acquisition activities.

4.1.1.4 The evaluation results shall be recorded in an appropriate document.

4.1.1.5 The Center or responsible Program Safety and Mission Assurance (SMA) organization shall approve the evaluation conclusions.

4.1.2 The requirements of this Standard shall apply to all safety-critical software elements regardless of the presence of non-software hazard controls or mitigations (e.g., operator intervention, hardware overrides).

4.2 Software as Part of System Safety Analysis

The system hazard analyses are the first place to identify specific software safety requirements for the program, project, or facility. System hazard analyses will be reviewed by software safety personnel to initially assess software's potential role and then, as the system development matures, assure 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 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, software assurance, software development, software safety, and the Center SMA organization) during the program or project's life cycle.

The Preliminary Hazard Analyses (PHA) identifies potential system hazards and may identify which proposed subsystems contribute to, or are needed to control, those hazards. The PHA and subsequent analyses will point to areas where needed deterrents and controls lead to requirements and design decisions.

The software safety analyses and activities, and their interaction with the system safety program, need to be planned early in the project life cycle. Entrance and exit criteria for the software safety analyses should be 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.

4.2.1 Software safety personnel shall participate in system safety analyses, including the PHA, which is usually conducted during the concept or formulation phase.

4.2.1.1 Identified hazards associated with a specific requirement, design concept and/or operation shall be evaluated for software's contribution to hazard causes, controls, or mitigations.

4.2.1.2 Software safety analyses shall be conducted in conjunction with the overall system safety analyses. System safety analyses provide input into software safety analyses, and results of the software analyses are provided back to the system safety program for use in updating and refining their analyses. These analyses, and the feedback loop, will continue throughout the system life cycle as more detail becomes available, including the design and verification of software safety features.

4.2.2 System safety analyses, including the PHA, subsequent system hazard analyses, and software safety analyses shall be used to create new, or identify existing, software requirements necessary to mitigate or resolve any hazards where software is a potential cause or contributor, or enable software to be used as a hazard control. Such requirements are designated as software safety requirements.

4.2.2.1 Identified software safety requirements and software hazard causes, contributors, and controls shall be recorded in an appropriate document and referenced in a safety plan. The requirements are usually documented in a section of the software requirements specification. The safety plan can be part of a system safety plan, a software management/development plan, a software or system assurance plan, or when warranted, in a standalone software safety plan.

5. SOFTWARE SAFETY MANAGEMENT

5.1 Organization and Responsibilities

Software safety requires a coordinated effort among all organizations involved in the development of NASA software. This includes program/project/facility managers, hardware and software designers, safety analysts, quality assurance, and operations personnel. Those conducting the software safety activities will also interface with personnel from disciplines such as reliability, security, IV&V, human factors and environmental. The responsibility for development of a safe system, including the software elements, belongs to the program/project manager in conjunction with the local safety and mission assurance organization (refer to NPR 8715.3 section 3.2.3).

5.1.1 Center Safety and Mission Assurance Organization

Center Safety and Mission Assurance (SMA) organizations have the responsibility to develop the necessary infrastructure to support the activities required by this Standard, to provide software safety experts to evaluate individual project/program/facility software safety programs, and to assure that the requirements of this Standard are implemented. They create the atmosphere within which individual programs, projects, or facilities operate.
Center SMA organizations are the focal point for assuring a healthy software safety program. Whether the bulk of the analyses is done in-house by the program/project/facility or by the contracting organization, the ultimate responsibility for seeing that an adequate safety program is in place, is the Center SMA organization. They have the responsibility to tell the project if their system is unsafe. They are responsible for representing any problems or concerns to NASA's Office of Safety and Mission Assurance (OSMA) prior to flight or operations. How each Center organizes to fulfill this responsibility is not intended to be implied in this document, only that the additional responsibilities to ensure software safety is adequately addressed.

It is assumed that the Center SMA organization will perform the following actions. The requirements for Center SMA organizations are imposed through NPD and NPR documents. They are included here as a reminder of the expected activities and their interaction with programs, projects, and facilities.

- Establish and maintain a Center software safety program as part of either their systems safety or software assurance program.

- Provide adequate resources, including: personnel trained in software safety, tools, and budget, for the software safety program.

- Assure that software safety is an integral part of the overall system safety and software development/acquisition efforts at their Center.

- Establish and maintain software safety processes, procedures, guidelines and tools which incorporate the requirements of this Standard.

- Ensure that all programs, projects, and facilities at their Center are periodically evaluated for the presence of safety-critical software. Maintain a record of these evaluations and the results at each Center which are made available to the NASA OSMA upon request.

- Gather and maintain a list of all safety-critical software within the Center. The list of safety-critical systems with software are sent to NASA OSMA upon request to help focus Code Q review of programs, projects, and facilities.

- Provide a means to resolve or elevate conflicts or concerns related to software safety requirements or processes.

- Establish a process for the certification of safety-critical software [reference section 5.14].

- Assure that project/program/facility software is evaluated for its role in safety and assure proper inclusion of safety-critical processes and products needed to acquire, develop, verify, certify and maintain safety-critical software. Starting with systems concepts and acquisition and continuing through retirement, the use of software experts will assure the proper balance of software safety planning and execution.

- Ensure software safety coverage is provided and active through the entire program, project, or facility life.

- Assure software safety personnel have evaluated, analyzed and provided input to program, project, and/or facility management on the selection of off-the-shelf or previously created (reused) software for incorporation into safety-critical systems.

- Assure that if IV&V is required on a program, project, or facility, project risk and software criticality determinations are shared between the safety personnel and IV&V.

- Provide monitoring and oversight of contractor software safety activities through the entire program, project, or facility life.

5.1.2 Program/Project/Facility Management Responsibilities

Program, project, or facility managers are responsible for making sure that their system is evaluated for the presence of safety-critical software. They are responsible for implementing a software safety program, providing adequate resources for the program and bear the risks if software safety activities are inadequate. Software safety should be considered as a part of the continuous risk management process adopted by the programs, projects or facilities.

5.1.2.1 Program/project/facility management shall be responsible for software safety planning within the project.

5.1.2.1.1 Program/project/facility management shall consult with software safety personnel regarding the acquisition of safety-critical software and its applicability to this Standard.

5.1.2.1.2 Program/project/facility management shall ensure that the acquired or developed system is periodically evaluated for the use of software in safety-critical functions.

5.1.2.1.3 Program/project/facility management shall provide adequate resources, including trained software safety personnel (trained per NASA policy), schedule time, tools, and budget, to the software safety program.

5.1.2.1.4 Program/project/facility management shall designate personnel to be responsible for the software safety program (e.g., software safety manager) of the project, program, or facility.

5.1.2.1.5 Program/project/facility management shall work with SMA management to provide a means to resolve conflicts related to software safety requirements or processes.

5.1.2.2 Program/project/facility management shall ensure that the software safety program is planned and executed throughout the entire software life cycle.

5.1.2.3 Program/project/facility management shall ensure that software safety is an integral part of the overall system safety and software development efforts.

5.1.2.4 Program/project/facility management shall implement a process or mechanism to document, trace, communicate, and close software safety concerns that result from safety analyses or design reviews, with concurrence of the safety personnel.

5.1.3 Software Safety Personnel

Software safety personnel, whether from the Center SMA organization or working with the project, program, or facility, have the responsibility to analyze the system and software and carry out the required software safety activities. The software safety personnel work in conjunction with the Center SMA system safety organization/personnel to develop and identify software safety requirements, distribute those safety requirements to the project management and software developers, contracts or acquisitions, and monitor their implementation. They also analyze software products and artifacts and provide the results to the system safety organization to be integrated with the other (non-software) subsystem analyses. Regular communication between system safety and software safety ensures that safety analyses are coordinated.

5.1.3.1 A software safety manager shall be assigned to each project, program or facility, with the responsibility to develop and implement the software safety processes and plans.

5.1.3.1.1 The software safety manager shall communicate software safety concerns directly to the project manager for resolution within the project.

5.1.3.1.2 The software safety manager shall follow the approved method to elevate software safety concerns that cannot be resolved within the project.

5.1.3.1.3 The software safety manager shall assure that risks affecting software safety are captured, addressed, and managed as part of program, project, and facility risk management processes, and those risks which could impose a system hazard are captured in the system hazard analyses.

5.1.3.1.4 The software safety manager (or designee) shall be a part of any change control board that approves software modifications affecting safety-critical systems.

5.1.3.1.5 The software safety manager shall provide input to management on the selection of off-the-shelf or previously created (reused) software for incorporation into safety-critical systems.

5.1.3.1.6 The software safety manager shall provide inputs to management regarding requirements to be imposed on a contractor(s) for development of safety-critical software. These requirements include, at a minimum, documentation, process definition, quality assurance and verification and validation requirements as they relate to assuring safety of the system.

5.1.3.2 One or more personnel shall be assigned the responsibility for performing software safety analyses (or assuring it is properly conducted and documented). This person or persons shall be referred to in this document as the software safety personnel.

5.1.3.2.1 Software safety personnel shall have the organizational freedom and authority to analyze and report software safety non-conformances.

5.1.3.2.2 Software safety personnel shall review system hazard analyses for changes that impact the software subsystem.

5.1.3.2.3 Software safety personnel shall provide information on changes in safety-critical software to system safety personnel for evaluation and incorporation into system safety documents.

5.1.3.2.4 Software safety personnel shall support the system safety review process.

5.1.3.2.5 Software safety personnel shall participate in project reviews. These include any NASA-specific reviews, e.g., Preliminary and Critical Design Reviews (PDR, CDR), Design Certification Review (DCR), FACI (First Article Configuration Inspection), Test Readiness Review (TRR), Certification of Flight Readiness (CoFR), Preflight Acceptance Review (PAR), Flight Acceptance Review (FAR), facility reviews, etc.

5.1.4 Other Personnel Responsibilities

Other personnel, such as Software Assurance, System Safety, Software Development, and Software Configuration Management, also play a part in the software safety process.

5.1.4.1 At least one software assurance engineer shall be assigned responsibility for assuring that software safety is planned, approved, and implemented.

5.1.4.1.1 The software assurance engineer shall assure that software safety processes, product standards and procedures are followed.

5.1.4.1.2 The software assurance engineer shall be assigned responsibility for performing software safety assurance audits.

5.1.4.1.3 The software assurance engineer shall report software safety process non-conformances to software and system safety personnel, to project/program/facility management.

5.1.4.2 The project/program/facility person responsible for Software Configuration Management shall assure that software safety elements are properly controlled. This includes performing the software configuration management functions of configuration control, change control, status accounting, and change verification of safety-critical software requirements and software elements.

5.2 Software Safety Planning

Software safety planning is essential in the acquisition or development of safety-critical software. Effective planning assures that adequate safety features are included within the system and software. This requires resources, personnel, time, and money - all of which are shared among the various project areas. Developing a plan early in the project ensures that software safety will be an integral part of the software development or acquisition process.

The Software Safety Plan outlines the project/program/facility software safety process, including organizational structure, interfaces, and the required criteria for analysis, reporting, evaluation, and data retention to provide a safe product. This safety plan describes forms of analysis and provides a schedule for performing a series of these system and subsystem level analyses throughout the development cycle. It also addresses how the results of software safety analyses and the sign-off and approval process should be handled. This plan provides the foundation for all future software safety activities.

The Software Safety Plan may be a separate document or included within other project management documents, such as the Software Management Plan or System Safety Plan. For the purposes of this Standard, regardless of where this information is documented, it will be referred to collectively as the Software Safety Plan in this Standard. IEEE 1228 may be used as a

template for developing the software safety plan. If desired, separate plans may be generated for different parts of the software life cycle (e.g., operations and retirement).

5.2.1 Software safety assessment and planning shall be performed for each software acquisition, development, and maintenance activity, and for changes to legacy systems.

5.2.1.1 Safety program reviews shall be planned and conducted to ensure proper implementation of the software safety program.

5.2.2 Software safety planning shall be implemented at a point in time sufficient to provide direction to personnel performing the software development and assurance activities. Ideally, software safety planning will begin at project conception or formulation. Legacy systems and projects already in development should determine, with input from Center or program SMA, how this Standard should be applied.

5.2.3 The software safety manager shall document software safety planning information in a Software Safety Plan.

5.2.3.1 If the Software Safety Plan is documented in multiple locations, each plan shall include a cross-reference to the safety activities in the associated/related plans.

5.2.3.2 The Software Safety Plan shall be under configuration control.

5.2.4. The Software Safety Plan shall describe how the requirements specified by this Standard will be implemented. For example, this can be done by means of a matrix showing the relationship between requirements of this Standard and the activities specified in the plan.

5.2.5 The Software Safety Plan shall specify the activities to be carried out, the schedule on which they will be implemented, the personnel who will carry out the activities, the methodologies used, and the products that will result.

5.2.6 The Software Safety Plan shall address the interrelationships among system safety, software assurance, software development efforts, and the Center or Program SMA organization.

5.2.6.1 If this project is a candidate for IV&V, the Software Safety Plan shall address, either specifically or by reference to the IV&V MOA, the role of IV&V for the safety-critical software and detail how IV&V will work with the software safety program and personnel.

5.2.6.2 The Software Safety Plan shall specifically address the mechanism by which safety-critical requirements are generated, implemented, tracked, and verified.

5.2.6.3 The Software Safety Plan shall specify procedures for ensuring prompt follow-up and satisfactory resolution of software safety concerns and recommendations.

5.2.6.4 The Software Safety Plan shall specify how the software safety activity schedule will be synchronized with related program/project activities.

5.2.6.5 The Software Safety Plan shall specify the number and relative schedule of software safety assurance audits.

5.2.6.6 The Software Safety Plan shall document an agreement between the project and NASA Center level SMA detailing when software safety engineers are required to review a system (e.g. when certain types of problems or anomalies are reported) and the proposed solutions or upgrades.

5.2.6.7 The Software Safety Plan will also document responsibility for monitoring the system during operation, and procedures to be followed when those monitoring the system feel safety of the system, environment, or personnel may be threatened.

5.2.7 The Software Safety Plan shall be periodically reviewed to ensure it addresses expected system operational conditions. These reviews consist of routine scheduled reviews, and event driven reviews. As a minimum, these reviews will be performed at the following times:

- Prior to delivery.
- Every 2 years.
- Prior to retirement, extended deactivation, and reactivation after retirement or extended periods.
- When a major change is made to the system or operating procedures.

Note: The Software Safety Plan should be revised when differences exist between the plan and actual/expected conditions. Software safety personnel may generate a completely new plan in place of revising the old plan if desired.

5.3 Personnel Qualifications and Training

The competence of the technical and managerial team members has a direct effect on the success of the project and the safety of the resulting system. Competence is comprised of knowledge, capabilities, and skills, plus the applicability of those skills to the task at hand. A well-qualified team member will have experience, training in the field of knowledge, knowledge of hazards and failures the system is capable of and knowledge of practices used in the organization.

5.3.1 The project/program/facility software safety plan shall have a section describing the training requirements for all project software safety roles. This includes training on or about the specific system and environment the project/program/facility will operate in.

5.4 Resources

Software safety activities will require resources (people, time, tools, etc.). These resources may be shared within a project or organization, or may be limited in other ways. Planning and

monitoring is required to effectively utilize the resources while providing adequate software safety support. Resource use should be monitored during the software safety program implementation. Resources include financial, schedule, safety personnel, other personnel, computer and other equipment support and tools.

5.4.1 Resource requirements and the allocation of those resources to software safety tasks for this project/program/facility shall be specified in an appropriate project plan and in the process planning documents.

5.5 Software Life Cycles

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 and maintenance. Software acquisitions (such as contractors providing software engineering, assurance and/or safety services, or commercial off-the-shelf software) or reuse of existing software require evaluation for any safety impacts. Up-front participation, analyses, and subsequent reporting of safety problems found during the software life cycle facilitate timely and less costly solutions.

5.5.1 The integration of software safety with the chosen software life cycle shall be documented in the project Software Safety Plan.

5.5.2 Software safety activities shall be performed throughout all phases of the software development life cycle. Activities which may be completed within or dependent upon a particular phase, or may need to be updated within successive phases, are documented as such.

5.5.3 Software safety activities shall continue to be performed at a needed level once the system is operational. Section 7 provides requirements for the operational phase of the system.

5.6 Documentation Requirements

The exact documents that a project will produce, and when they will be produced, are determined during the planning phase (e.g., Concept or Formulation phase). Documentation should reflect the project size, complexity, and criticality. Contractual obligations or required standards may also influence the amount of documentation produced. Safety-critical software will need to be addressed in multiple documents, to an appropriate level of detail.

5.6.1 The documents to be prepared as part of the software safety program, and their contents, shall be specified in the Software Safety Plan.

5.6.2 The change and approval process for software safety related portions of all project documents, including the plan itself, shall be specified in an appropriate project plan.

5.6.3 The following documentation shall address safety-critical software:

- Software Safety Plan

- Software Project Management Plan
- Software Configuration Management Plan
- Software Quality Assurance Plan
- Software Requirements Specification
- Software Design Documentation
- Verification and Validation Plan
- Safety Analyses and Reports
- Test Documentation
- User documentation and procedures
- Operations and Maintenance Plan

5.7 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 hazard analysis, these requirements will also be linked to specific hazard reports. For small projects this may done with a spreadsheet or textual document. These simple tools will require some labor intensive work to maintain at the appropriate level but it can be done.

Tracing requirements is a vital part of system verification and validation, and especially in safety verifications. Full requirements test coverage is virtually impossible without some form of requirements traceability. Tracing also provides a way to understand 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 tracing system should be chosen based on project complexity and the number of requirements. The tracing system should be capable of forward (flow down) and backward (to higher level) tracing, especially in large and complex system. The tracing system needs to be operational early in the project life cycle, at least by the time the system requirements are baselined. The tracing system should be updated and maintained throughout the project life cycle.

5.7.1 A tracing system shall map the relationships between software safety requirements and system hazards, as well as trace the flow down of software safety requirements to design, implementation, and test.

5.7.1.1 The software tracing system shall include, or link to, the system-level hazard tracking system to allow tracking of software-related hazard controls and mitigations, and to verify closure of system hazards.

5.7.2 The tracing system shall be under configuration control.

5.7.3 The tracing system reports shall be reviewed by software safety personnel. These reports are, at a minimum, available for project formal reviews.

5.8 Discrepancy and Problem Reporting and Tracking

Discrepancy reports 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). Changes that result from identified problems usually go through a software Change Control Board (CCB) for discussion and approval or rejection. Software safety must be considered for all software changes, and software safety personnel should be on the CCB. These problem reports need to be tracked, and management needs to have visibility into past and current software problems. The problem reporting system that is utilized need not be a separate system from the software development problem reporting system.

5.8.1 There shall be a system for closed-loop tracking of discrepancies, problems, and failures in the baselined safety-critical software products and processes.

5.8.1.1 This system shall trace identified safety-critical software problems back to the system-level hazard involved.

5.8.1.2 Software safety personnel shall approve safety-critical discrepancy report closures.

5.8.2 All discrepancy reports shall be reviewed regularly for safety impacts by software safety personnel.

5.8.3 All software changes including those that result from problem or discrepancy resolution shall be evaluated for potential safety impact, 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.

5.9 Software Configuration Management Activities

Safety-critical software is managed in accordance with a software configuration management process that is approved by the software configuration manager. Software configuration management is practiced during all phases of the software life cycle, from initiation of development through software maintenance, and is responsible for ensuring that any changes during the development and maintenance processes are made in a controlled and complete

manner. Software configuration management includes control of project documentation, source code, object code, data, development and test tools, development and test environments (both hardware and software), and testing software. Configuration management contributes to software safety by ensuring that documentation and source code are updated only through a formal process. Software safety personnel are involved in this formal process (e.g., through a CCB) and can assess the impact of the change to safety-critical software.

5.9.1 Software and documentation shall be placed under strict configuration control, including source code, executables, test plans and procedures, and associated data, prior to verification of the safety requirements.

5.9.1.1 All baselined safety-critical software and associated documentation, simulators, models, test suites, data, etc. shall be maintained in a controlled configuration management system.

5.9.1.2 The organization responsible for Software Configuration Management shall formally provide and document the release of safety-critical software.

5.9.2 All changes, modifications, and patches made to safety-critical requirements, design, code, systems, equipment, test plans, procedures, simulators, models, test suites, or criteria shall be evaluated to determine the effect of the proposed change on system safety.

5.9.2.1 Software safety personnel shall approve changes to baselined safety-critical software.

5.9.3 For software in its operational phase, the configuration management system shall track and control incremental changes to the safety-critical software and its release to operations.

5.9.3.1 Any reconfiguration changes made to the software system on a routine basis (e.g., mission-specific database changes) shall be configuration controlled. This allows a record so that safety impacts may be analyzed if needed.

5.10 Software Assurance Activities

Software Assurance ensures the proper performance of key software safety program activities and the integration of safety into the software being produced. Software assurance periodically reviews and/or audits for compliance with the defined software processes for acquisition, development, and safety assurance of safety-critical software. The requirements in NASA-STD-8739.8, NASA Software Assurance Standard are applicable to safety-critical software.

5.10.1 Acceptance or closure of any system-level hazards related to software shall be dependent on the successful conclusion of all assurance activities linked to its associated software safety requirements.

5.10.2 Software safety tasks shall be coordinated with the overall software assurance disciplines to eliminate duplication of effort.

5.11 Tool Support and Approval

Use of project tools, such as computer-aided software engineering (CASE) products, compilers, editors, fault tree generators, simulators, emulators, and test environments for hardware and software, in a software development and maintenance effort can inadvertently introduce software hazards into the software. The following measures are put in place to lessen this possibility.

5.11.1 The approach to preventing the inadvertent introduction of software hazards by project tools shall be documented in an appropriate project plan. Tools may include CASE products, compilers, editors, fault tree generators, simulators, emulators, and test environments for hardware and software.

5.11.1.1 All project tools that could potentially impact safety-critical software, the degree of impact, and mitigation strategies shall be identified in the appropriate project plan.

5.11.1.2 The process and criteria used to select, approve, and control project tools shall be described in the appropriate project plan.

5.11.1.2.1 The process shall address the following areas: installation of upgrades to previously approved tools, withdrawal of a previously approved tool, and identification of limitations that may be imposed on tool use.

5.11.1.2.2 The software safety manager shall ensure sufficient safety testing and analysis is performed to verify that any changes in the use of project tools does not influence known hazards or adversely affect the residual risk of the software.

5.11.2 The software safety manager shall approve the approach.

5.12 Off-the-shelf Software (COTS/GOTS/OTS)

Including software in a safety-critical system that was not developed specifically for that 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. Differences in configuration of the software or operational constraints may affect the operation of the OTS/reused software, sometimes in unexpected ways.

5.12.1 All off-the-shelf and reused software shall be evaluated for the potential to impact safety-critical functions within the current system.

5.12.1.1 Safety-critical OTS and reused software shall undergo safety analysis that considers its ability to meet required safety functions, extra functionality, even if not planned for use that may be present, the impact on safety, and interfaces to developed code.

5.12.1.2 Software safety analysis shall consider the interactions of COTS software components with the developed software and any other COTS software that is part of the system.

5.12.1.3 Additional analysis, testing, or a combination thereof shall be performed to verify safety-critical OTS or reused software to the same level required of in-house developed software to the extent possible via black box testing.

5.13 Contract Management

The requirements for software safety apply to software developed or acquired by NASA. When safety-critical software is acquired by a program/project/facility, this Standard must be imposed on those who perform the software development. Safety-critical software may be acquired from contractors, subcontractors, non-NASA government agencies, universities, and other NASA Centers.

This Standard is levied on all parties who develop safety-critical software, including NASA or other government agencies, contractors, and subcontractors. If the software is being acquired without specific software safety clauses or this Standard invoked in the contract or agreement (e.g., MOA/MOU), then the contract or agreement should be either renegotiated for inclusion of these software safety requirements or the NASA program/project/facility must implement and adhere to this Standard.

5.13.1 The contract or MOA/MOU shall include provisions sufficient to assure that the contracted safety-critical software is developed according to this Standard.

5.13.1.1 The contract or MOA/MOU shall include all software safety deliverables, including the software safety plan, preliminary and subsequent hazard analyses, safety-critical software development audit reports, and verification reports.

5.13.1.2 The contract or MOA/MOU shall specify how the customer (i.e., the NASA program/project) will determine if the contractor is performing the software safety activities properly.

5.13.1.3 The contract or MOA/MOU shall define a method for problem reporting and corrective action between the contractor and the customer.

5.13.1.4 The contract or MOA/MOU shall specify that customer agreement is required for changes to baselined safety-critical software elements.

5.14 Certification Process

Safety-critical software is certified by ensuring it is produced in accordance with the requirements in this Standard. This may be a Center, program, project or facility specific certification process. Regular communication between all parties helps ensure a smooth

certification process. Waivers/deviations to safety requirements are addressed in section 5.15 of this Standard.

5.14.1 There shall be an official certification process established, documented, and conducted prior to the release of any safety-critical software for its intended operational use.

5.14.2 Center Safety and Mission Assurance software safety personnel shall participate in the certification process.

5.14.3 The software safety organization shall participate in evaluation of the following areas as part of their certification process:

a. All software hazards have been identified.

b. All hazard controls that require software implementation have been identified.

c. All software safety requirements and elements have been identified and tracked.

d. All software safety requirements and elements have been successfully validated, or waivers/deviations have been approved.

e. All software safety requirements and elements have been properly verified, or waivers/deviations have been approved.

f. All discrepancies in safety-critical software have been dispositioned with the safety organization's concurrence, per the certification process.

g. All operational workarounds associated with discrepancies in safety-critical software have the concurrence of the Center or Program safety organization, per the certification process.

5.14.4 Personnel conducting software safety functions shall be prepared to represent the software to an appropriate safety panel for certification.

5.14.5 The organization providing the safety engineering shall approve the results and reports prior to acceptance of the software and the system. The Center SMA organization reviews the results and provides final certification or approval for operation of safety-critical products and facilities.

5.15 Waivers/Deviations

When the software project cannot meet the safety requirements or the certification criteria in this Standard, a waiver or deviation should be submitted for approval to the certifying authorities. A deviation is requested prior to development 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 within the development process or even, possibly near the end when the

project has tried but failed to meet the requirement. To proceed, the project will suggest alternative ways to meet the requirement or choose to accept the risk. Waiver and deviation packages should include the potential impact to system safety analyses, such as FMEA's, Hazard Reports and/or other program/project/facility risk evaluations.

5.15.1 If one or more requirements (i.e., a numbered "shall" statement) contained within this Standard cannot be met by any safety-critical software project, a waiver/deviation package shall be prepared by a software safety expert and approved according to NPR 8715.3.

5.15.2 The project shall submit a written request for a waiver/deviation, detailing the justification to support the waiver/deviation.

5.15.3 The Center SMA organization shall maintain a copy of all variances to safety requirements contained in this Standard, and provide these variances to the NASA Headquarters Office of Safety and Mission Assurance upon request.

5.16 Security

The goal of the security measures is to protect against sabotage, collusion, compromise or alteration of safety-critical software elements. Security breaches could impact the safety-critical system, therefore, NPD 2810.1, Security of Information Technology, and the associated guidance, NPR 2810.1, should be considered for safety-critical software.

6 SOFTWARE DEVELOPMENT AND SAFETY ANALYSES

The following subsections describe which software safety tasks to perform for each software development life cycle phase, using the waterfall life cycle as the primary life cycle methodology. Many of the safety analyses are iterative, taking place during the software development life cycle (i.e., the results from one phase feed the analyses of the next). As the detail of the program/project/facility evolves, so does the maturity of the safety analysis.

The tracing system mentioned in section 5.7 should be used to trace the flow down of the software safety requirements to design, implementation, and test. The tracing system should also map the relationships between software safety requirements and system hazard reports.

6.1 Software Safety Requirements and Analysis

Software safety requirements development and analysis is performed in conjunction with or immediately following the system hazard analysis. Software safety requirements carry a unique identification in the software requirements specification for traceability purposes. A way to mark and trace these requirements throughout the development and operational phases is needed in order to easily check for impacts and changes to the requirements.

Software safety personnel, working closely with the system safety team, have the responsibility to analyze these software safety requirements for correctness of decomposition from the system safety requirements, as well as verifying that the software safety requirements will maintain the system in a safe state and provide adequate response to potential failures. Software safety requirements should do more than prohibit unsafe behavior. 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. Therefore, software safety requirements should be included that will embody these behaviors, both proactive and reactive, and include the system and software states where they are valid.

6.1.1 Software safety requirements shall be developed and included in the software requirements specification.

6.1.1.1 Software safety requirements shall be derived from the system safety requirements, environmental requirements, standards, program specification, vehicle or facility requirements, interface requirements, system hazard reports, and system hazard analyses [ref. section 4.2].

6.1.1.2 Software safety requirements, both generic and specific, shall be clearly identified as such in the software requirements specification.

6.1.1.3 Software safety requirements shall be expressed and structured so that they are clear, precise, unequivocal, verifiable, testable, maintainable and feasible.

6.1.1.4 Software safety requirements shall include the modes or states of operation under which they are valid, and any modes or states in which they are not applicable.

Note: These requirements are also referred to as “must work” and “must not work” functions. For example, the safety-critical commands and checks which initiate a robotic arm movement must not work during system initiation or perhaps when in maintenance mode.

6.1.1.5 Any safety related constraints between the hardware and software shall be included in the software requirements documentation. That is, when the software and hardware work together to perform a safety-critical function, their roles, precedence, and failure modes, are documented and understood.

6.1.2 Software safety personnel shall analyze the software safety requirements, both technical and procedural.

6.1.2.1 The analysis methodology shall be recorded in an appropriate document and include the following steps, at a minimum:

a. Verify that all software safety requirements meet the requirements of section 6.1.1 and sub-sections.

b. Examine the software safety requirements for ambiguities, inconsistencies, omissions, and undefined conditions.

c. Verify that all software safety requirements are traceable to system safety requirements, environmental requirements, standards, program specification, vehicle or facility requirements, interface requirements, and system hazard reports.

d. Verify that the software safety requirements provide adequate response to potential failures. Areas to consider should include, but are not limited to, limit ranges, relationship logic for interdependent limits, out-of-sequence event protection, timing problems, sensor or actuator failures, voting logic, hazardous command processing requirements, Fault Detection, Isolation, and Recovery (FDIR), switchover logic for failure tolerance, and the ability to reach and maintain a safe state if so required.

e. Verify that the software safety requirements include positive measures to prevent potential problems and implement required “must work” functions.

6.1.2.2 The documented results of the analysis, including any newly identified hazards, hazard causes, and improperly decomposed requirements, shall be provided to the responsible system safety personnel.

6.1.2.3 Improperly decomposed requirements shall be documented for project-level resolution.

6.1.2.4 The software safety requirements analysis results shall be presented at project formal reviews and system-level safety reviews by the responsible safety organization.

6.2 Software Design and Safety Analysis

Software design of safety requirements will identify software safety design features and methods. The software design will ensure that software safety features and requirements can be thoroughly tested. The safety requirements are traced to the design elements that implement the requirement, and each design element is traced back to the requirements which implements it. Design elements will be identified as safety-critical in design documentation. Software safety personnel have the responsibility to conduct an analysis of the software design. The analysis methodology will be documented and any improperly designed safety features will be documented and reported.

6.2.1 All functional software safety requirements shall be incorporated into the software design.

6.2.1.1 The software design shall identify safety design features and methods (e.g., inhibits, failure detection and recovery, interlocks, assertions, and partitions) that will be used to implement the software safety requirements.

6.2.1.2 The software design shall allow software safety features and requirements to be thoroughly tested.

6.2.1.3 Design elements that implement safety-critical requirements or can potentially affect the safety-critical elements through failure or other mechanisms, shall be designated as safety-critical.

6.2.1.3.1 Software design documentation shall clearly identify all safety-critical design elements.

6.2.1.4 To the extent practical, the software design shall modularize the safety-related aspects of the design [ref. NASA-GB-8719.13, Software Safety Guidebook].

6.2.2 Software safety personnel shall analyze the software design.

6.2.2.1 The analysis methodology shall be recorded in an appropriate document (e.g., software safety plan or software assurance plan).

6.2.2.2 The analysis methodology shall include the following steps, at a minimum:

a. Verify that the software design meets the requirements of section 6.2.1 and sub-sections.

b. Verify that the design does not compromise any safety controls or processes, that any additional hazard, hazard cause, or hazard contribution is documented, and that the design

maintains the system in a safe state during all modes of operation. The analysis should, at a minimum, consider:

- timing constraints
- hardware failures
- common-mode failures
- fault migration
- communications
- interrupts
- concurrency
- event sequence
- fault tolerance
- FDIR design
- adverse environments
- invalid inputs
- off-the-shelf or reused software
- design assumptions
- information flow

c. Verify that safety features incorporated in the design are adequate for their function.

d. Safety analyses, such as PHAs, sub-system hazard analyses, FMEAs (Failure Modes and Effects Analysis), FTAs (Fault Tree Analysis), shall be used to help determine design features to prevent, mitigate or control failures and faults, and the level of failure/fault combinations to include (e.g., both a software and a hardware failure, or multiple concurrent hardware failures).

e. Verify that any partitioning or isolation methods used in the design adequately isolate the safety-critical design elements from those that are non-safety-critical. This is particularly important with the incorporation of COTS.

f. Verify all safety-critical design elements are traceable to software safety requirements, and vice versa.

6.2.2.3 The documented results of the analysis including any newly identified hazards, shall be provided to the responsible system safety personnel.

6.2.2.4 The software safety design analysis results shall be presented at project formal reviews and system-level safety reviews.

6.3 Software Implementation and Safety Analysis

Software implementation will include all software safety design features and methods. For traceability purposes, safety-critical code and data will be commented as such. Software safety personnel are responsible for analyzing the method of implementation, documenting the analysis methodology used, and documenting and reporting any improperly implemented safety features.

6.3.1 All software safety design features and methods shall be implemented in the software code.

6.3.1.1 The software coding standards shall incorporate requirements for clearly identifying safety-critical code and data within source code comments, and strongly discouraging unsafe language features such as pointers or memcopy, requiring these features to also be clearly

identified and documented whenever used [ref. checklist in NASA-GB-8719.13, NASA Software Safety Guidebook].

6.3.1.2 The software coding standard shall be used in the development of software code.

6.3.2 Software safety personnel shall analyze the software implementation (e.g., code).

6.3.2.1 The analysis methodology shall be recorded in an appropriate document (e.g., software safety plan or software assurance plan).

6.3.2.2 The analysis methodology shall include the following steps, at a minimum, and can include source code reviews and inspections:

a. Verify that the safety-critical software code and data meets the requirements of section 6.3.1 and sub-sections.

b. Verify that design safety features and methods are correctly implemented in the software code.

c. Verify that the code implementation does not compromise any safety controls or processes, does not create any additional hazards, and maintains the system in a safe state during all modes of operation. The analysis should, at a minimum, consider the elements detailed in 6.2.2.2.b.

d. Ensure that code and data verification activities adequately substantiate all software safety requirements, to the extent that a requirement can be verified at a component or unit level.

e. Verify all safety-critical code units are traceable to safety-critical design elements.

6.3.2.3 The documented results of the analysis, including any newly identified hazards and improperly implemented safety features, shall be provided to the responsible system safety personnel.

6.3.2.4 The software safety code analysis results shall be presented at project formal reviews and system-level safety reviews.

6.3.3 Verification of each safety-critical code unit and data shall be completed prior to the unit's incorporation in a higher-level code package.

6.4 Software Test and Safety Analysis

Software testing will include safety testing at the unit level and component level, as well as system and acceptance testing. The results of the lower level tests and all artifacts used to conduct the tests will be documented. In addition, all artifacts used to conduct the tests will be placed under configuration management. Traceability will be carried through into the testing

documentation. Software safety personnel analyze the methods of testing, document the analysis methodology used, and document and report any improperly implemented safety features.

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 this 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 needs to be thoroughly documented as well.

6.4.1 All functional software safety requirements and safety-critical software elements shall be verified by testing.

6.4.1.1 Testing shall verify that system hazards related to software have been eliminated or controlled to an acceptable level of risk.

6.4.1.2 Unit level tests and component level tests shall include software safety testing.

6.4.1.2.1 Any simulators, test drivers and stubs, along with any test data, used for testing at the unit level shall be configuration controlled and documented.

6.4.1.2.2 Any simulators, test drivers and stubs, along with any test data, used for testing at the component level shall be configuration controlled and documented.

6.4.1.2.3 The results of unit level and component level tests and the test procedures, simulators, test suites, drivers, stubs and data shall be documented.

Note: 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.

6.4.1.3 System and acceptance tests shall include software safety testing.

6.4.1.3.1 Correct and safe operation of the software in conjunction with system hardware and operator inputs shall be verified prior to system acceptance.

6.4.1.3.2 System testing shall verify the correct and safe operation of the system in the presence of failures and faults including software, hardware, input, timing, memory corruption, communication, and other failures.

6.4.1.3.3 Safety analyses, such as PHAs, sub-system hazard analyses, FMEAs, FTAs, shall be used to determine which failures to test for, and the level of failure combinations to include (e.g., both a software and a hardware failure, or multiple concurrent hardware failures).

6.4.1.3.4 System testing shall verify the correct and safe operation of the system under system load, stress, and off-nominal conditions.

6.4.1.3.5 System testing shall verify correct and safe operations in all anticipated operational and off-nominal configurations.

6.4.1.4 Additional hazardous states or contributors identified during testing shall undergo complete analysis prior to software delivery or use.

6.4.2 Requirements that cannot be verified by test shall be verified by evaluation, inspection, or demonstration.

6.4.2.1 The rationale for selecting evaluation, inspection, or demonstration shall be recorded in an appropriate document (e.g., system safety report, hazard analysis).

6.4.2.2 The evaluation, inspection, or demonstration methodology shall be recorded in an appropriate document.

6.4.2.3 The software safety personnel shall concur with both the rationale for not performing a test and the selected evaluation, inspection, or demonstration methodology used to verify the requirement.

6.4.3 The results from the software and system test process, or the requirements verification evaluation, inspection, or demonstration process, shall be analyzed.

6.4.3.1 The analysis methodology shall be recorded in an appropriate document.

6.4.3.2 The analysis methodology shall include the following steps, at a minimum:

a. Verify that the software and system tests data meet the requirements of section 6.4.1 and sub-sections.

b. Verify that the requirements verification evaluation, inspection, or demonstration data meet the requirements of section 6.4.2 and sub-sections

c. Verify via test coverage analysis that all safety requirements, functions, controls, and processes have been completely covered within the unit, component, system, and acceptance level tests.

d. Verify that all software safety requirements have been tested, or evaluated, inspected, or demonstrated.

e. Verify that all software safety functions are correctly performed and that the software system does not perform unintended functions.

6.4.3.3 The documented results of the analysis, including any newly identified hazards and improperly implemented safety features, shall be provided to the responsible system safety personnel.

6.4.3.4 Improperly implemented safety features shall be input into the problem reporting system for project-level resolution.

6.4.3.5 The software safety test analysis results shall be presented at project formal reviews and system-level safety reviews.

7. OPERATIONAL USE OF THE SOFTWARE

When the software is delivered and starts its operational phase, this does not mean the end of the software safety activities. These 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 typical example of this type of change is mission specific database updates in flight systems. A key point to remember is that just because a change is of a routine nature, this does not excuse it from the same requirements as all other changes made to the software.

During system operations, the project may not require full time safety personnel. For some projects, another organization may be responsible for maintenance and operations. While the previous software safety plan can be updated, it may be better to create a new safety plan which specifically addresses the operational and retirement phase of the system. This plan will address the safety and assurance resources required, and their interaction with operations and maintenance activities. The plan should identify the development and assurance processes used when changes are made to the software. Since personnel changes are expected during this phase, the plan should address how adequate software and system safety expertise for this system will be maintained.

Part of the on-going assurance activity should be to perform a safety evaluation. Not every data or software change will require a safety engineer's approval, but the project and NASA Center or Program SMA need to have an agreement as to what types and levels of changes require safety personnel to examine the situation and proposed solutions or upgrades. Software assurance may be the monitors of the system under operation and alert safety when they feel safety of the system, environment, or personnel is of concern.

Note: 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 this error occurred. This should include an examination of the operational environment and its intended usage.

7.1 The requirements of this Standard shall continue to be applicable after the safety-critical software has been released for operations.

7.2 The software safety requirements to specify, develop, analyze, and test safety-critical software, shall apply to all changes made to the software or routine operational updates (e.g., mission specific database updates).

7.2.1 Software safety change analysis shall evaluate whether the proposed change could invoke a hazardous state, affect a hazard control, increase the likelihood of a hazardous state, adversely affect safety-critical software, or change the safety-criticality of an existing software element.

7.2.1.1 The analysis activity shall include an assessment of the amount of regression testing needed to verify that the implementation of new software requirements has not affected the implementation of existing safety-critical software.

7.2.1.2 Software safety personnel shall concur on any changes to basic, as built, or approved upgrades of the operational software.

7.3 Operational documentation, including user manuals and procedures, shall describe all safety related commands, data, input sequences, options, and other items necessary for the safe operation of the system.

7.3.1 All error message descriptions and corrective actions shall be included in operational documentation.

7.3.2 Software safety personnel shall review any updates to user manuals and procedures for safety impacts, and to ensure that any software-related hazard closures that depend on operational workarounds are properly documented.

7.4 The requirements of this Standard expire for a particular facility or system only upon the retirement of that facility or system.

7.4.1 When the facility or system is retired, there shall be a retirement plan which addresses the safe termination of operations, decommissioning, and retirement of that system or facility.

APPENDIX A. EXAMPLE - TAILORING ACTIVITIES TO IMPLEMENT STANDARD

A.1 Introduction

While the requirements of the NASA Software Safety Standard cannot be tailored, the activities performed to meet the requirements in the Standard can and should be tailored. That is, while the requirements must be met, the implementation and approach to meeting these requirements may and should vary to reflect the system to which they are applied. The level of risk posed by the safety-critical software will be a function of the hazard criticality and the degree of control the software has over the safety functions of the system. The NASA Software Safety Guidebook, NASA-GB-8719.13, provides additional information on determining the software risk level and how to tailor activities and analyses to that risk level.

The scenario below provides an example of how an International Space Station (ISS) payload project might implement the requirements of the Standard. The tailoring activities required for this project are illustrated below in sections A.3 through A.6 for sections 4 through 7 of the Standard. Section A.2 provides background information on the ISS payload project.

A.2 ISS Payload Information

The ISS payload has a catastrophic hazard (venting toxic gases), which requires three independent controls. Hardware will provide two of the controls. The software will monitor the system and shut down power (which closes the vent valves) if there is a problem (the third hazard control). This software has direct control of the vent valves. In addition, the software package must perform many additional functions in support of the science objectives. All the software in the system resides on the same processor.

A.3 Safety-Critical Software Determination (section 4 of Standard)

A.3.1 Determination Process (section 4.1 of Standard)

After applying the Software Safety Litmus Test, the project software assurance engineer (SAE) determines the software to be safety-critical since it provides control for a hazard. The hazard control software will be a separate task from payload functional software, but operates on the same system. Therefore, all software on the system is safety-critical. This determination is communicated to the project system safety engineer (SSE) for inclusion in the system safety analyses.

A risk assessment of the software is performed by the SAE per the NASA-GB-8719.13, Software Safety Guidebook. The software risk level result (based on Table 3-3 in NASA-GB-8719.13) is determined to be a two, or medium risk. The required software safety effort is determined to be "moderate" for the hazard control. The other software in the system will require a “moderate” safety effort, and must be evaluated to ensure it cannot affect the hazard control.

A.3.2 Software as Part of System Safety Analysis (section 4.2 of Standard)

The project’s SSE performs a Preliminary Hazard Analysis (PHA), documents the results, and places the results under configuration management (CM). The SAE contributes to the PHA and uses it and the results of the Software Safety Litmus Test to determine if any software is safety-critical. The SAE communicates with the SSE, project software personnel, the project manager (PM), and system safety or

software safety representatives from the Safety and Mission Assurance (SMA) organization to work any issues that arise. For example, the project will need to decide on the allocation of safety requirements between hardware and software, the number or depth of required analyses, the methodology to implement safety features, and the classification of each software component. All of these issues will require discussion and agreement between all parties. Because this is an ISS payload, Payload Safety Data Packages and Hazard Reports are used to document any software hazard causes or controls, and the safety verifications that are agreed to by the project, the ISS Payload Safety Review Panel (PSRP), and the Center SMA software safety representative.

A.4 Software Safety Management (section 5 of Standard)

A.4.1 Organization and Responsibilities (section 5.1 of Standard)

The responsibilities for project personnel include:

- **Project Manager (PM):** Designates personnel roles and responsibilities within the project. Ensures that adequate resources (personnel, time, and budget) are planned for software safety activities.
- **System Safety Engineer (SSE):** Coordinates all safety activities and acts as the software safety manager.
- **Software Assurance Engineer (SAE):** Performs all software safety activities (i.e. performs both roles - software safety and software assurance). The SAE was working 20% on this project. With the additional software safety tasks, this was raised to 40%.
- **Software Lead:** Works with the SAE to develop software requirements, design, code, and tests that implement or verify all safety features.
- **Configuration Manager:** Controls software documentation and code.

The responsibilities of the Center SMA organization include:

- **Software Safety Engineer:** Provides expertise to the project as needed. Reviews the software portion of the system safety plan and perform audits of the project software safety activities prior to Critical Design Review (CDR) and Pre-ship Review. The audit of the Pre-ship Review is used as part of the certification process to verify that all assurance activities required were performed for a given hazard control.
- **Software Assurance Engineer:** Provides expertise to the project as needed. Reviews the planned software assurance activities and may approve the Software Assurance Plan. Audits the software assurance process to verify compliance with the Software Assurance Plan. Ensures coordination of IV&V Facility activities with the project.
- **System Safety Engineer:** Provides expertise to the project as needed.
- **SMA Director:** Signs off on the certification process. Approves and rejects waivers and deviations. Reviews the software safety status throughout the life cycle, if requested.

The project SAE works with the Center SMA software assurance engineer (SSAE) and system safety personnel, sharing information and discussing options for identifying, implementing, and verifying software safety requirements. For significant software safety issues, the PM obtains SMA's approval or concurrence on the issues, or formally documents them as an accepted risk (if appropriate).

The project SAE works closely with the software team to identify any problems or issues early, and to provide guidance when the team is considering various options that might have a safety impact. The SAE provides system safety with information on how the software requirements and design may impact the safety functions. Software is included in the hazard and safety analyses, where appropriate.

A.4.2 Software Safety Planning (section 5.2 of Standard)

NASA-GB-8719.13 was used to select the software safety activities, based on the software safety risk, which are detailed in the system safety plan. Documents are listed for each activity - mostly analysis reports and audit reports. A matrix mapping these activities to the requirements of this Standard is included in the system safety plan. Software safety elements in all software documentation have a special header that identifies them as safety-critical. This formatting is specified in the Software Management and Development Plan. The plans are reviewed, and revised if necessary, before each project formal review, each payload safety review, and when new hazards are discovered.

The PM established a chain of command within the project, from software safety to system safety to the PM. Anyone with a safety concern that is not satisfactorily resolved by the PM is encouraged to contact the Center SMA organization for additional information and guidance. SMA has developed an informal "discussion" process for concerns that can be submitted as formal concerns to SMA, if warranted. SMA will then follow up on the concerns.

The project team has weekly meetings and many in-the-hall discussions to maintain cognizance of project activities and status. Regular meeting topics include safety (including software safety) status, risks (especially safety related risks), CM status, and anticipated system and software changes. The PM encourages all team members to meet individually with the PM at least monthly to review progress and to air any concerns.

A.4.3 Personnel Qualifications and Training (section 5.3 of Standard)

To meet the requirement for trained software safety personnel, both the project SAE and the software lead take the System Safety Fundamentals and the Software System Safety courses offered by NASA Safety Training Center. The project SAE also takes the Advanced System Safety Practice course to ensure they will be able to recognize and control safety operating risks within acceptable limits.

A.4.4 Resources (section 5.4 of Standard)

The project allocated additional personnel resources (increasing the SAE's time) to cover software safety activities. The project software schedule, which is rolled up into the master project schedule, includes software safety activities and milestones. The Software Assurance Plan describes the software safety analyses and activities that will be performed by the SAE and, where appropriate, the SSAE for the project.

A.4.5 Software Life Cycles (section 5.5 of Standard)

The Software Management and Development Plan references the software safety activities that are described in the Software Assurance Plan, and describes how they work with the software development activities. These activities are tied to specific life cycle phases where appropriate. The project is using an incremental development life cycle, so many of the documents and tests will be developed first for a minimal software product, then added to for each increment. Software safety activities will mirror this approach.

A.4.6 Documentation Requirements (section 5.6 of Standard)

The various plans and documents for the project that include software safety information and activities are:

- **Project Plan** - Documents the working relationships and communication paths within the team.

- **System Safety Plan** - Points to the Software Assurance Plan for software safety activities, etc.
- **Software Management and Development Plan** - Includes specific schedule and resource values for safety-critical software development. References the software safety risk analysis performed, which defines the "safety effort level" required (per NASA-GB-8719.13).
- **Software Assurance Plan** - Contains the software safety activities that will be carried out, the schedule, and documentation produced. Identifies when outside software assurance is needed to audit software safety activities.
- **Software Configuration Management Plan**- Includes identification scheme for files containing safety-critical software. Defines how the Change Control Board (CCB) handles software that is safety-critical.
- **Project schedule** - Includes software safety activity schedule.
- **Software Requirements Specification** - Includes a section of software safety requirements. Safety-critical requirements are identified with a ".s" appended to their number.
- **Software Design Specification** - Safety-critical software is described in a section on the design document. This section also discusses how the software is partitioned.
- **Safety analyses and reports** - Includes or references software safety analyses results and reports.
- **Test documentation** - Identifies when a safety-critical requirement is being tested and requires the project SAE to witness testing and sign verification.
- **User Documentation and Procedures** - Identifies hazardous commands, as well as commands that could jeopardize the safety functions. Identifies trends, off-nominal values, and other indications from system health and status that could indicate the system is moving toward an unsafe state.
- **Operations and Maintenance Plan** - Identifies the procedure for upgrading software, including required regression testing, approvals, etc.
- **Safety Data Package** – Includes documentation of the design and implementation process used for safety-critical software, and the test and inspection results. Safety Data Package is submitted to the PSRP.
- **Flight system configuration files -** Uploaded to the experiment and are under configuration control. Provisions are in place for handling changes to the files in an emergency/critical situation.

A.4.7 Traceability (section 5.7 of Standard)

The project uses spreadsheets as its software tracing matrices to maintain the traceability relationship between the software safety requirements and system hazards, and the flow down of software safety requirements into the design, code, and test. The spreadsheets have various sorting functions, for example, that will allow for the easy identification of all requirements that implement a particular hazard control. All spreadsheets are under CM control. System safety uses their own hazard tracking method, which points to the software tracking matrix for software-related controls, etc.

A.4.8 Discrepancy & Problem Reporting and Tracking (section 5.8 of Standard)

System and sub-system (including software) safety concerns or recommendations are tracked using the project's Problem Reporting and Corrective Action (PRACA) system. PRACAs are reviewed weekly by a panel consisting of project engineers, including software, and project assurance personnel. Any PRACAs that are not resolved within two weeks are brought to the attention of the PM.

A.4.9 Software Configuration Management Activities (section 5.9 of Standard)

The Configuration Manager requires that the safety-critical software modules are in separate files and flags them as safety-critical. Only specified developers can check out those files. Once the software is baselined, project software safety and system safety must both sign any change requests, and must verify that the changed software functions properly before it can be checked back into the CM system. All project plans and documents are placed in the CM system. CM provides formal releases of software for loading into the hardware, and testing (once the software is baselined).

A.4.10 Software Assurance Activities (section 5.10 of Standard)

The project SAE performs software assurance and software safety tasks as defined in the Software Assurance Plan, as part of the project assurance activities. The SAE coordinates with system safety and with the SMA organization to ensure that activities are carried out with appropriate independence and that minimal overlap occurs.

Prior to system delivery, all hazard-related activities must be completed. This requires formal documentation that all required activities were performed. The SAE will provide to the project system safety engineer a report for each hazard that includes software. This report will:

- Specify the hazard and the software element (e.g., control)
- List safety-critical software requirements derived from this hazard
- List, and provide the status of, verification activities for the safety-critical software requirements
- Provide references to related documents (e.g., test procedures and verification reports)
- Include a statement that all software assurance activities are complete for this hazard

The SSAE will review the reports, and may audit the project SAE activities to assure that all required actions were completed.

A.4.11 Tools Support and Approval (section 5.11 of Standard)

The tools used in this project include the compiler, operating system, libraries, language, code generators, and development environment. The project uses the following qualification process to ensure these tools do not inadvertently introduce software hazards into the software. This process is reviewed and approved by the Center SMA organization. The Center SMA may audit the project for compliance to this process.

1. Developer submits a tool usage request to the SAE for review. Any safety concerns are reported, along with recommendations for alternatives.
2. The SAE evaluates the tool using the established criteria: certification, use in other safety-critical systems, good product history (few bugs, few open issues, etc.), source-code availability, etc. These criteria are given a weighting factor and included in the Software Assurance Plan (safety section).
3. A Center SMA software safety engineer reviews the tool selection criteria.
4. If the tool is accepted, a formal statement to that effect, signed by the SAE, is appended to the Software Management and Development Plan.
5. If the tool is rejected by the SAE, either
    a. An alternative tool that meets the criteria is identified by the SAE or the software developer or
    b. The risk of using the tool is documented and the PM formally accepts the risk.
6. Accepted tools are maintained in the CM system for version control.
7. Defects in the tools are captured in a project-wide tool spreadsheet.

The project does not plan to change tools once they are approved. If a change is required, the qualification process is followed for this additional tool.

A.4.12 Off-the-shelf Software (COTS/GOTS/OTS) (section 5.12 of Standard)

The project intends to use an OTS operating system for the software controlling the vent valves. The project SAE will:

- Review the software developers' selection of commercial operating system and tools; make recommendations.
- Discuss with the software lead how the safety-critical software interacts with the operating system (OS). Develop possible failure modes that could cause the safety-critical software to fail (or to fail to operate correctly).
- Consider the reliability or robustness of the OS, in conjunction with the level of risk posed by failure of the safety-critical software.
- Evaluate the amount of code in the OS that is not used, and determine if it is insignificant and/or if it could cause a failure if the code was accidentally executed.
- In the same manner that safety-critical software is tested for response to off-nominal or stress conditions, verify that the safety task still runs properly by ensuring safety testing includes testing the OS under stress conditions.

A.4.13 Contract Management (section 5.13 of Standard)

The project plans to acquire software from another organization. Therefore, the SAE who covers software safety will review the contract provisions for:

- Inclusion of the NASA Software Safety Standard, the NASA Software Assurance Standard, and any applicable Center or program standards or processes.
- Inclusion of software assurance and software safety tasks in the Statement of Work.
- Specific inclusion of software in all safety analyses called for as deliverables.
- Additional software safety deliverables, such as audit reports, that are required.
- Delivery of all information required for the project to meet the certification requirements.
- Oversight and periodic audits of contractor safety activities.
- Reporting mechanism for safety problems or concerns.
- Mechanisms to control changes to safety-critical software.
- Safety verification being performed at the system-level by the project.

A.4.14 Certification Process (section 5.14 of Standard)

The Center has established a certification process to ensure safety-critical software is produced in accordance with the requirements in the NASA Software Safety Standard. The process consists of the following steps:

1. Submittal of a formal letter (with supporting verification documentation) by the SAE to SMA, stating that all the 5.14.3 elements have been verified by the project safety organization.
2. Review of the documentation by a designated SMA system or software safety engineer. An audit can be performed to verify some or all of the documentation and the reported activities.
3. Submittal of all documentation (from steps 1 and 2) to the SMA director (or designee).
4. Production of a certification letter signed by the SMA director.

A.4.15 Waivers/Deviations (section 5.15 of Standard)

If a waiver or deviation of requirements is necessary, the SAE will assist the PM in preparing the documentation and justification for the waiver or deviation. A Center SMA software safety expert will review the waiver/deviation package and provide a written report to the SMA Director, who will approve or reject the waiver/deviation.

A.4.16 Security

The project software lead and SAE discuss security issues with Center network security personnel. They set up a secure sub-network that allows only authorized personnel to command the payload while it is in development. During flight operations, the control center has physical and network security in place to prevent unauthorized access.

A.5 Software Development and Safety Analyses (section 6 of Standard)

Throughout the project's life cycle, the project SAE will perform the following activities:

- Update the system safety plan to include software safety activities and processes.
- Review hazard and system safety analyses.
- Inform system safety of any issues with software requirements and design that may impact the safety functions.
- Review the project's risks monthly and add any safety-related risks during the weekly risk status meetings.
- Perform audits of the software development effort, especially for verification of the inclusion of safety features.
- Submit PRACA when an audit notes problems with processes relating to safety.
- Participate in the system and software CCB. Review the proposed software changes and submits a short written report to the CCB. Use the traceability matrix to see what else may be affected by the change. Review the proposed change for any possible impacts to the safety controls.
- Present the software safety status (i.e., documentation) to the responsible SMA system safety engineer and to the SMA Director if requested. Present the software safety process to the PSRP as part of that safety process.
- As part of a standard CM audit, verify that safety-critical software and associated information is in the CM system and correctly identified.
- Present software safety information and status at formal reviews.
- Work with the project manager to perform an IV&V self-assessment. If the project requires an Independent Assessment (IA) by the IV&V Facility, communicate current software safety activities with the IV&V Facility person assigned to the project. Incorporate the results from the IA in the software safety reporting process.

A.5.1 Software Safety Requirements and Analysis (section 6.1 of Standard)

During the requirements phase, the project SAE will:

- Review the software requirements against the system requirements, software standards, generic software safety requirements, hazard reports, and safety analyses and add any missing software safety requirements.
- Perform a requirements safety analysis as recommended by NASA-GB-8719.13. This analysis verifies both protection from negative behavior and implementation of proactive requirements to prevent problems from occurring. A short analysis report is written and submitted to the SSE.

- Report to the SSE any problems found or new hazards discovered. If system safety concurs, document these in PRACAs.
- Prepare a traceability matrix (spreadsheet) for all software safety requirements.
- Present the software safety assessment at the System or Software Requirements Review and the Phase 0/1 ISS payload safety review.

A.5.2 Software Design and Safety Analysis (section 6.2 of Standard)

During the design phase, the project SAE will:

- Utilize the traceability matrix to assure that all software safety requirements are incorporated in the design.
- Assure that safety features are identified in the software design and that they are separate from the rest of the software (i.e., in a separate task).
- Review the design with an eye to testability and suggest modifications that may improve the ability to test safety features at all levels of testing.
- Add to the design document the line "This function is safety-critical" to any part of the design that implements safety requirements.
- In flow charts, Unified Modeling Language (UML) diagrams, or other graphics, place a thick red border on safety-critical design elements.
- As part of the design safety analysis (as recommended by NASA-GB-8719.13):
    - Review the failure modes of the operating system and other tasks. (While failure of other tasks has a slight possibility of affecting the safety task, the risk is very slight.)
    - Determine that the other tasks are not safety-critical given that this is a "moderate" software safety effort.
    - Document this decision with a justification, and have the SMA system safety manager (per Center policy) approve the decision. (The operating system is determined to be safety-critical, since its failure would cause the safety task to fail.)
    - Consider the effects of interrupts, timing, event sequences, hardware failures, communication problems, etc., on the safety-critical task.
- Prepare a short analysis report and submit it and any associated data files to the system safety engineer, along with any problems found or new hazards discovered. If system safety concurs, document in PRACAs.
- Present the analyses and results at the project Preliminary and Critical Design Reviews and the Phase 2 ISS safety review.

A.5.3 Software Implementation and Safety Analysis (section 6.3 of Standard)

During the implementation phase, the project SAE will:

- Using the traceability matrix, assure that all safety-critical software requirements and design elements are incorporated in the code.
- Verify all source code files that implement safety-critical requirements/design have a comment block at the top that indicates these files are safety-critical.
- Review the coding standard against the appropriate language checklist in NASA-GB-8719.13. Provide any elements of the checklist not in the coding standard to the developers for inclusion in the coding standard.
- Assure through Formal Inspections and a development process audit that the coding standard is used by software developers.
- Perform a code safety analysis per NASA-GB-8719.13.
- Perform Formal Inspections for safety-critical code.

- Review unit test procedures and test logs for safety-critical units. Test procedures can be written in a logbook, and do not need to be formally controlled documents.
- Prepare a short analysis report and submit it and any associated data files to the system safety engineer, along with any problems found or new hazards discovered. If system safety concurs, document in PRACAs.

A.5.4 Software Test and Safety Analysis (section 6.4 of Standard)

During the test phase, the project SAE will:

- Verify that all software safety requirements are thoroughly tested.
    - Utilize testing to verify safety-critical software operates correctly in a variety of conditions, including load, stress, and sensor failures, while the system is performing multiple tasks, up to 100% CPU load, etc.
    - Verify the safety task in all experiment configurations.
    - Verify that the testing shows that failure of other system elements (e.g., tasks) does not compromise the safety task.
    - Use the traceability matrix, along with a test coverage tool, to verify all software safety requirements, design elements, and code components are tested.
- Verify by inspection of the source code, any requirements that cannot be verified by test. (For example, it is impractical to test a requirement that the system will move to an off-nominal state if the Power-on self test has a failure, due to the need to inject a fault in a very narrow time range.) Agree and approve with the deviation from test.
- Sign off on software safety verifications after reviewing the test plan/procedure, witnessing the test, and reviewing the test report and data.
- When the project is testing each module and task individually, review test results for each module and witness testing of each task.
- Verify test reports are prepared for module and task testing of safety-critical source code, and that these reports are put into the CM system.
- Verify software safety requirements are included in the system tests and that the acceptance test includes verification of the safety system. (System tests include simulated hardware failures and operator errors.)
- Witness system testing of software safety requirements. Verify that the software and test plans/procedures were generated from CM controlled documents prior to the start of test.
- Review system documentation and assure that the system test procedures incorporate testing of all identified failures. (While it is not practical to test for all possible failures, system documentation can provide potential failures that the software must handle correctly and safely.)
- If a hazard is discovered, work with the system safety engineer to analyze the hazard and determine what mitigations/controls are necessary to prevent it.
- Adhere to the ISO documented methodology for analyzing software and system test process and results; records what is done, data files used, and any deviations from the procedure.

A.6 Operational Use of the Software (section 7 of Standard)

Once the experiment is on-orbit, much of the project team moves on to other projects. The project SAE will be called back by the project, when required, to perform or update analyses. This activity is estimated to be 10% of the SAE's time for the first 6 months (while the experiment may experience operational problems that require updates to the software). It is estimated to drop to an average of 4 hours per month after that time.

The project SAE (or another person who can fulfill the functions) will:

- Approve any changes to on-orbit software and any data updates, as part of the CCB.
- Perform an impact analysis for any software changes that show the level of regression testing necessary.
- Witness the regression tests.
- Review the user manuals and operational procedures and assure that safety features (e.g., error message lists), or actions that can impact the safety features, are clearly identified.

APPENDIX B. REQUIREMENTS COMPLIANCE MATRIX

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
Safety-Critical Software Determination	4.0	*Not a requirement*					
Determination of Safety-Critical Software	4.1	*Not a requirement*					
	4.1.1	If the system is safety-critical, evaluate the software.	SysSafety SwSafety				
	4.1.1.1	Use the criteria in this section to determine if the software is safety-critical	SysSafety SwSafety				
	4.1.1.2	Evaluate software during project planning	SysSafety SwSafety				
	4.1.1.3	Document the results of the evaluation	SysSafety SwSafety				
	4.1.1.4	SMA approves of the evaluation conclusions.	SMA				
	4.1.2	Evaluate all software in the system	SysSafety SwSafety				
	4.1.3	Apply this Standard even if using non-software hazard controls	SysSafety SwSafety				
Software and System Safety	4.2	*Not a requirement*					
	4.2.1	Participate in system safety analyses	SwSafety				
	4.2.1.1	Evaluate hazards for software's contribution (cause, control, etc.)	SwSafety				
	4.2.1.2	Conduct software safety analyses; coordinate with the system safety analyses.	SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	4.2.2	Create software safety requirements	SwSafety SwEng				
	4.2.2.1	Document software safety requirements, hazard contributors and controls. Software safety plan points to this document.	SwSafety				
Software Safety Management	5.0	*Not a requirement*					
Organization and Responsibility	5.1	*Not a requirement*					
Center S&MA	5.1.1	*Not a requirement*					
Program, Project, Facility Management	5.1.2	*Not a requirement*					
	5.1.2.1	Include software safety in project planning	PM				
	5.1.2.1.1	Consult with software safety personnel when acquiring safety-critical software	PM				
	5.1.2.1.2	Periodically evaluate the system for safety-critical software	PM				
	5.1.2.1.3	Provides adequate resources to the software safety program.	PM				
	5.1.2.1.4	Assign personnel for the software safety program	PM				
	5.1.2.1.5	Work with SMA management to resolve conflicts	PM				
	5.1.2.2	Plan and execute software safety throughout the entire software life cycle.	PM				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.1.2.3	Integrate software safety with system safety and software development.	PM				
	5.1.2.4	Create a process to document, trace, communicate, and close software safety concerns	PM				
Software Safety Personnel	5.1.3	*Not a requirement*					
	5.1.3.1	Assign sw safety mgr to develop and implement sw safety processes and plans	PM				
	5.1.3.1.1	Communicate software safety concerns directly to the project manager	SwSafetyMgr				
	5.1.3.1.2	Elevate software safety concerns that cannot be resolved within the project.	SwSafetyMgr				
	5.1.3.1.3	Assure software safety risks are captured, addressed, and managed as part of risk management processes	SwSafetyMgr				
	5.1.3.1.4	Participate in change control board for software modifications	SwSafetyMgr				
	5.1.3.1.5	Brief management on the selection of off-the-shelf or previously created (reused) software used in this system	SwSafetyMgr				
	5.1.3.1.6	Provide inputs to management regarding contactor requirements for safety-critical software.	SwSafetyMgr				
	5.1.3.2	Assign personnel to perform software safety activities	PMSwSafetyMgrSMA				
	5.1.3.2.1	Analyze and report software safety non-conformances to appropriate personnel	SwSafety				
	5.1.3.2.2	Review system hazard analyses for changes that impact the software subsystem.	SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.1.3.2.3	Inform system safety personnel of changes in safety-critical software	SwSafety				
	5.1.3.2.4	Support the system safety review process.	SwSafety				
	5.1.3.2.5	Participate in project reviews.	SwSafety				
Other Personnel	5.1.4	*Not a requirement*					
	5.1.4.1	Assign software assurance to verify that software safety is planned, approved, and implemented.	PM SMA				
	5.1.4.1.1	Verify software safety processes, product standards and procedures are followed.	SwAssure				
	5.1.4.1.2	Perform software safety assurance audits.	SwAssure				
	5.1.4.1.3	Report software safety process non-conformances to software and system safety personnel and/or appropriate management.	SwAssure				
	5.1.4.2	Perform configuration change control, status accounting, and change verification of safety-critical software requirements and software elements.	SCM				
Software Safety Planning	5.2	*Not a requirement*					
	5.2.1	Perform software safety assessment and planning for each software acquisition, maintenance activity, or change to legacy systems.	SwSafetyMgr				
	5.2.1.1	Plan and conduct safety program reviews	SwSafetyMgr SwAssure				
	5.2.2	Start software safety planning early enough to affect the software development and assurance activities.	SwSafetyMgr				
	5.2.3	Create a Software Safety Plan.	SwSafetyMgr				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.2.3.1	Cross-reference safety activities that are in multiple plans.	SwSafetyPlan				
	5.2.3.2	Put the Software Safety Plan under configuration control.	SCM				
	5.2.4.	Describe how the requirements of this Standard will be implemented .	SwSafetyPlan				
	5.2.5	Include activities, schedule, personnel, methods, and resulting products.	SwSafetyPlan				
	5.2.6	Define how system safety, software assurance, software development, and the Center or Program SMA organization works together.	SwSafetyPlan				
	5.2.6.1	Describe the role of IV&V and how IV&V will work with the software safety program and personnel.	SwSafetyPlan				
	5.2.6.2	Describe how safety-critical requirements are generated, implemented, tracked, and verified.	SwSafetyPlan				
	5.2.6.3	Define ithe procedures for resolving software safety concerns and recommendations.	SwSafetyPlan				
	5.2.6.4	Describe how software safety and project schedules are synchronized.	SwSafetyPlan				
	5.2.6.5	Specify the number and schedule of software safety assurance audits.	SwSafetyPlan				
	5.2.6.6	Document the conditions requiring software safety engineers to review a situation and proposed solutions or upgrades.	SwSafetyPlan				
	5.2.6.7	Define who monitors system during operation, & what procedures are followed when they feel safety may be threatened.	SwSafetyPlan				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.2.7	Review the Software Safety Plan periodically.	PM SwSafetyMgr SwAssure				
Personnel Qualifications and Training	5.3	*Not a requirement*					
	5.3.1	Describe the training requirements for all project software safety roles.	SwSafetyPlan				
Resources	5.4	*Not a requirement*					
	5.4.1	Document the resource requirements and allocation for software safety tasks.	SwSafetyPlan				
Software Life Cycles	5.5	*Not a requirement*					
	5.5.1	Describe how software safety is integrated with the chosen software life cycle.	SwSafetyPlan				
	5.5.2	Perform software safety activities throughout the software development life cycle.	SwSafety				
	5.5.3	Continue software safety activities in the operational phase.	PM SwSafetyMgr				
Documentation Requirements	5.6	*Not a requirement*					
	5.6.1	List the documents (and associated content) that are part of the software safety program in the Software Safety Plan.	SwSafetyPlan				
	5.6.2	Define the change and approval process for software safety related portions of all project documents.	SwSafetyMgr				
	5.6.3	Address safety-critical software in all appropriate project documents.	PM SwSafetyMgr				
Traceability	5.7	*Not a requirement*					
	5.7.1	Create a tracing system that maps software safety requirements to system hazards and traces the flow down of to design, implementation, and test.	PM SwSafetyMgr				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.7.1.1	Coordinate the software tracing system with the system-level hazard tracking system.	SysSafety SwSafetyMgr				
	5.7.2	Put the tracing system under configuration control.	SCM				
	5.7.3	Review the tracing system reports and outputs.	SwSafety				
Discrepancy and Problem Reporting and Tracking	5.8	*Not a requirement*					
	5.8.1	Create closed-loop tracking of discrepancies, problems, and failures	PM SwSafetyMgr				
	5.8.1.1	Trace identified safety-critical software problems back to the system-level hazard involved.	SwSafety				
	5.8.1.2	Approve safety-critical discrepancy report closures.	SwSafety				
	5.8.2	Regularly review all discrepancy reports for safety impacts.	SwSafety				
	5.8.3	Evaluate software changes for potential safety impact.	SwSafety				
Software Configuration Management Activities	5.9	*Not a requirement*					
	5.9.1	Configuration manage software, documentation, and associated data	SCM				
	5.9.1.1	Maintain all baselined safety-critical software and associated documentation, simulators, models, test suites, data, etc.	SCM				
	5.9.1.2	Provide and document the release of safety-critical software.	SCM				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.9.2	Evaluate all changes, modifications, and patches made to safety-critical requirements, design, code, systems, equipment, test plans, procedures, simulators, models, test suites, or criteria.	SwSafety				
	5.9.2.1	Approve changes to baselined safety-critical software.	SwSafety				
	5.9.3	Track and control incremental changes to the safety-critical software and its release to operations.	SCM				
	5.9.3.1	Configuration control routine reconfigurations and changes to operational software	SCM				
Software Assurance Activities	5.10	*Not a requirement*					
	5.10.1	Complete all assurance activities prior to acceptance or closure of any software-related system-level hazards.	SwAssure				
	5.10.2	Coordinate software safety tasks with software assurance.	SwSafety SwAssure				
Tool Support and Approval	5.11	*Not a requirement*					
	5.11.1	Define approach to preventing software tools from introducing hazards	SwEng SwSafetyMgr				
	5.11.1.1	Identify & assess project tools that could potentially impact safety-critical software and define mitigation strategies if necessary.	SwEng SwSafety				
	5.11.1.2	Document how project tools are selected, approved, and controlled.	SwEng SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.11.1.2.1	Document how approved tools are upgraded, what happens if an approved tool is no longer approved, and what limitations are imposed on tool use.	SwEng SwSafetyMgr				
	5.11.1.2.2	Ensure sufficient safety testing and analysis is performed when a project tool changes	SwSafetyMgr				
	5.11.2	Approve the safety analysis and testing approach for tool verification.	SwSafetyMgr				
Off-the-shelf Software	5.12	*Not a requirement*					
	5.12.1	Evaluate all off-the-shelf and reused software for its impact safety-critical functions.	SwSafety				
	5.12.1.1	Analyze safety-critical OTS and reused software for its ability to meet required safety functions, any safety impact of extra functionality, and interfaces to developed code.	SwEng SwSafetyMgr				
	5.12.1.2	Analyze the interactions of COTS software components with the developed software and any other COTS software in the system.	SwSafety				
	5.12.1.3	Verify safety-critical OTS or reused software to the same level required of in-house developed software to the extent possible.	SwSafety				
Contract Management	5.13	*Not a requirement*					
	5.13.1	Contract/MOA/MOU requires safety-critical software be developed according to this Standard.	PMSwSafetyMgr				
	5.13.1.1	Software safety deliverables are included in the contract/MOA/MOU.	PM SwSafetyMgr				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.13.1.2	Customer surveillance for software safety is included in the contract/MOA/MOU.	PM SwSafetyMgr				
	5.13.1.3	The contract/MOA/MOU defines how the contractor and customer report and resolve software safety problems.	PM SwSafetyMgr				
	5.13.1.4	The contract/MOA/MOU defines that customer agreement is required for any changes to baselined safety-critical software elements.	PM SwSafetyMgr				
Certification Process	5.14	*Not a requirement*					
	5.14.1	Establish a certification process for safety-critical software. Safety-critical software is certified prior to use or release.	SMA				
	5.14.2	Participate in program/project/facility certifications	SMA				
	5.14.3	5.14.3-(a-g) are items to be evaluated for certification	SMA SwSafetyMgr				
	5.14.3-a	All software hazards are identified.	SMA SwSafetyMgr				
	5.14.3-b	All hazard controls that require software implementation are identified.	SMA SwSafetyMgr				
	5.14.3-c	All software safety requirements and elements are identified and tracked.	SMA SwSafetyMgr				
	5.14.3-d	All software safety requirements and elements have been successfully validated, or waivers/deviations have been approved.	SMA SwSafetyMgr				
	5.14.3-e	All software safety requirements and elements have been properly verified, or waivers/deviations have been approved.	SMA SwSafetyMgr				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	5.14.3-f	All discrepancies in safety-critical software have been dispositioned with the safety organization's concurrence, per the certification process.	SMA SwSafetyMgr				
	5.14.3-g	All operational workarounds associated with discrepancies in safety-critical software have the concurrence of the Center or Program safety organization, per the certification process.	SMA SwSafetyMgr				
	5.14.4	Present the software safety process and results to an appropriate safety panel for certification.	SwSafety				
	5.14.5	Approve the results and reports prior to acceptance of the software and the system, with review and certification provided by SMA .	SMA SwSafetyMgr				
Waivers/ Deviations	5.15	*Not a requirement*					
	5.15.1	Request a waiver/deviation if a requirement cannot be met.	SwSafetyMgr				
	5.15.2	Document in a written request the justification to support the waiver/deviation.	SwSafetyMgr				
	5.15.3	Waiver/deviation is signed by the project and the Center Safety and Mission Assurance Director.	PM SMA				
	5.15.4	Keep copies of all variances to safety requirements.	SMA				
Software Development and Safety Analyses	6.0	*Not a requirement*					

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
Software Safety Requirements and Analysis	6.1	*Not a requirement*					
	6.1.1	Create software safety requirements and include them in the software requirements specification.	SwEng SwSafety				
	6.1.1.1	Derive software safety requirements from system safety, environmental, interface, vehicle, and facility requirements; standards, program specification, system hazard reports and analyses	SwEng SwSafety				
	6.1.1.2	Clearly identify software safety requirements in the software requirements specification.	SwEng				
	6.1.1.3	Express and structure software safety requirements that are clear, precise, unequivocal, verifiable, testable, maintainable and feasible.	SwEng SwSafety				
	6.1.1.4	Include the modes or states of operation under which software safety requirements are valid or not applicable.	SwEng				
	6.1.1.5	Include hardware and software safety-related constraints in software requirements document.	SwEng SwSafety SysSafety				
	6.1.2	Analyze software safety requirements.	SwSafety				
	6.1.2.1	Ensure the analysis method or procedure meets the following requirements (a-e) and is documented:	SwSafety				
	6.1.2.1-a	Verify all software safety requirements meet the requirements of section 6.1.1 and sub-sections	SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	6.1.2.1-b	Examine the software safety requirements for ambiguities, inconsistencies, omissions, and undefined conditions	SwSafety				
	6.1.2.1-c	Verify all software safety requirements are traceable to higher-level requirements or external standards	SwSafety				
	6.1.2.1-d	Verify software safety requirements provide adequate response to potential failures.	SwSafety				
	6.1.2.1-e	Veirfy software safety requirements include positive measures to prevent potential problems and implement required “must work” functions.	SwSafety				
	6.1.2.2	Document results of the analysis. Any newly identified hazards are given to system safety.	SwSafety				
	6.1.2.3	Document project-level resolution for improperly decomposed requirements.	SwSafety				
	6.1.2.4	Include results of the software safety requirements analysis at project formal reviews and system-level safety reviews.	SwSafety				
Software Design and Safety Analysis	6.2	*Not a requirement*					
	6.2.1	Incorporate all functional software safety requirements into the software design.	SwEng SwSafety				
	6.2.1.1	Identify safety design features and methods in the software design.	SwEng SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	6.2.1.2	Allow for software safety features and requirements be thoroughly tested in the software design.	SwEng SwSafety				
	6.2.1.3	Designate design elements that implement safety-critical requirements as safety-critical.	SwEng SwSafety				
	6.2.1.3.1	Clearly identify all safety-critical design elements in software design documentation.	SwEng				
	6.2.1.4	Modularize the safety-related aspects of the design in the software design.	SwEng				
	6.2.2	Analyze the software design.	SwSafety SwAssure				
	6.2.2.1	Document the analysis methodology	SwSafetySwAssure				
	6.2.2.2	Ensure documented analysis method or procedure meets the following requirements (a-f):	SwSafety SwAssure				
	6.2.2.2-a	Verify software design meets the requirements of section 6.2.1 and sub-sections.	SwSafety SwAssure				
	6.2.2.2-b	Verify design does not compromise any safety controls or processes. Any additional hazard cause or contribution is documented. The design maintains the system in a safe state during all modes of operation.	SwSafety SwAssure				
	6.2.2.2-c	Verify safety features are adequate for their function.	SwSafety SwAssure				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	6.2.2.2-d	Determine design features necessary to prevent, mitigate or control failures and faults, and the partitioning of safety features between hardware and software.	SwSafety				
	6.2.2.2-e	Verify that any partitioning or isolation methods adequately isolate the safety-critical design elements from those that are non-safety-critical.	SwSafety SwAssure				
	6.2.2.2-f	Verify all safety-critical design elements are traceable to software safety requirements, and vice versa.	SwSafety SwAssure				
	6.2.2.3	Provide the documented results of the analysis and any newly identified hazards to system safety.	SwSafety SwAssure				
	6.2.2.4	Include the results of the software safety design analysis at project formal reviews and system-level safety reviews.	SwSafety SwAssure				
Software Implementation and Safety Analysis	6.3	*Not a requirement*					
	6.3.1	Implement all software safety design features and methods in the software code.	SwEng SwSafety				
	6.3.1.1	Incorporate software coding standards that prohibit unsafe language features and require commenting of safety-critical source code.	SwEng SwSafety				
	6.3.1.2	Utilize software coding standard in developing code.	SwEng SwAssure SwSafety				
	6.3.2	Analyze the software implementation (e.g., code).	SwSafety SwAssure				
	6.3.2.1	Document the analysis methodology	SwSafetySwAssure				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	6.3.2.2	Ensure the analysis method or procedure meets the following requirements (a-e) and is documented	SwSafety SwAssure				
	6.3.2.2-a	Verify safety-critical software code and data meets the requirements of section 6.3.1 and sub-sections.	SwSafety SwAssure				
	6.3.2.2-b	Verify design safety features and methods are correctly implemented in the software code.	SwSafety SwAssure				
	6.3.2.2-c	Verify the software maintains the system in a safe state during all modes of operation and does not compromise any safety controls or processes, nor create any additional hazards.	SwSafety SwAssure				
	6.3.2.2-d	Ensure code and data verification activities include software safety requirements, if a requirement can be verified at this level.	SwSafety SwAssure				
	6.3.2.2-e	Verify all safety-critical code units are traceable to safety-critical design elements.	SwSafety SwAssure				
	6.3.2.3	Provide the documented results of the analysis and any newly identified hazards to system safety.	SwSafety SwAssure				
	6.3.2.4	Include the results of the software safety design analysis at project formal reviews and system-level safety reviews.	SwSafety SwAssure				
	6.3.3	Verify unit testing and data verification is completed before the unit is integrated.	SwEng SwAssure				
Software Test	6.4	*Not a requirement*					

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
and Safety Analysis	6.4.1	Verify all functional software safety requirements and safety-critical software elements by testing.	SwEng				
	6.4.1.1	Verify via test that system hazards related to software have been eliminated or controlled to an acceptable level of risk.	SwEng SwAssure SwSafety				
	6.4.1.2	Include software safety testing in unit level tests and component level tests.	SwEng SwSafety				
	6.4.1.2.1	Configuration control unit test simulators, test drivers, stubs, and test data.	SCM				
	6.4.1.2.2	Configuration control component test simulators, test drivers, stubs, and test data.	SCM				
	6.4.1.2.3	Document the results of unit level and component level tests, plus the test procedures, simulators, test suites, drivers, stubs and data used.	SwEngSwSafety				
	6.4.1.3	Include software safety testing within system and acceptance tests.	SwEng SwSafety				
	6.4.1.3.1	Verify the correct and safe operation of the software in conjunction with system hardware and operator inputs prior to system acceptance.	SwEng SwSafety				
	6.4.1.3.2	Verify the correct and safe operation of the system in the presence of failures and faults.	SwEng SwSafety				
	6.4.1.3.3	Use safety analyses to determine which failures to test for, how many, and in what combinations.	SwEng SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	6.4.1.3.4	Verify the correct and safe operation of the system under system load, stress, and off-nominal conditions.	SwEng SwSafety				
	6.4.1.3.5	Verify the system operates correctly and safely in all anticipated operational and off-nominal configurations.	SwEng SwSafety				
	6.4.1.4	Newly identified hazardous states or contributors are analyzed prior to software delivery or use.	SwSafety SysSafety				
	6.4.2	Evaluate, inspect, or demonstrate requirements if testing is not feasible.	SwEng				
	6.4.2.1	Record the rationale for choosing evaluation, inspection, or demonstration over test.	SwEng				
	6.4.2.2	Record the evaluation, inspection, or demonstration methodology.	SwEng				
	6.4.2.3	Get the software safety engineer's concurrence with both the rationale and the methodology.	SwEng SwSafety				
	6.4.3	Analyze results from the software and system test process or requirements verification process.	SwSafety SwAssure				
	6.4.3.1	Document the analysis methodology	SwSafety SwAssure				
	6.4.3.2	Ensure the analysis method or procedure meets the following requirements (a-d) and is documented.	SwSafety SwAssure				
	6.4.3.2-a	Verify software and system tests data meet the requirements of section 6.4.1 and sub-sections.	SwSafety SwAssure				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	6.4.3.2-b	Verify that the requirements verification evaluation, inspection, or demonstration data meet the requirements of section 6.4.2 and sub-sections.	SwSafetySwAssure				
	6.4.3.2-c	Verify test coverage analysis shows that all safety requirements, functions, controls, and processes have been completely covered.	SwSafety SwAssure				
	6.4.3.2-d	Verify software safety requirements have been tested, or evaluated, inspected, or demonstrated.	SwSafety SwAssure				
	6.4.3.2-e	Verify all software safety functions are correctly performed and the software system does not perform unintended functions.	SwSafety SwAssure				
	6.4.3.3	Document the results of the analysis and provide any newly identified hazards to system safety.	SwSafety SwAssure				
	6.4.3.4	Document and report improperly implemented requirements for project-level resolution.	SwSafety SwAssure				
Operational Use of Software	6.4.3.5	Present the results of the software safety design analysis at project formal reviews and system-level safety reviews.	SwSafety SwAssure				
	7.1	This Standard applies to safety-critical software that has been released for operations.	PM SwSafetyMgr				
	7.2	When operational software is changed, specify, develop, analyze, and test all safety-critical software.	SwEng SwSafety				

NASA-STD-8719.13B Requirements Compliance Matrix							
Section	**No.**	**Requirement**	**Role/ Responsibility***	**Compliance**			**Comments**
				Full	**Partial**	**None**	
	7.2.1	Evaluate proposed changes for their impact on system safety.	SwSafety				
	7.2.1.1	Assess the amount of regression testing needed.	SwSafety				
	7.2.1.2	Concur on any changes to basic, as built, or approved upgrades of the operational software.	SwSafety				
	7.3	Operational documents describe all safety related commands, data, input sequences, and options.	PM SwEng SwSafety				
	7.3.1	Operational documents include error message descriptions and corrective actions.	PM SwEng SwSafety				
	7.3.2	Review updates to user manuals and procedures for safety impacts, and verify software safety related operational workarounds are properly documented.	SwSafety				
	7.4	When a system or facility is retired, this Standard no longer applies.	PM SwSafetyMgr				
	7.4.1	A retirement plan will address the safe termination of operations, decommissioning, and retirement of the system or facility.	PM SwSafetyMgr SMA				

*** Role/Responsibility Definitions:**

SCM	Software Configuration Management
PM	Program/Project/Facility Management
SwAssure	Software Assurance personnel
SwSafety	Software Safety personnel

SwSafetyMgr	Software Safety Manager
SwSafety Plan	Requirements for what is included in the Software Safety Plan