Dryden Flight Research Center
Edwards, California

DCP-S-007
Revision: B

Dryden Centerwide Procedure

CODE S

SOFTWARE ASSURANCE

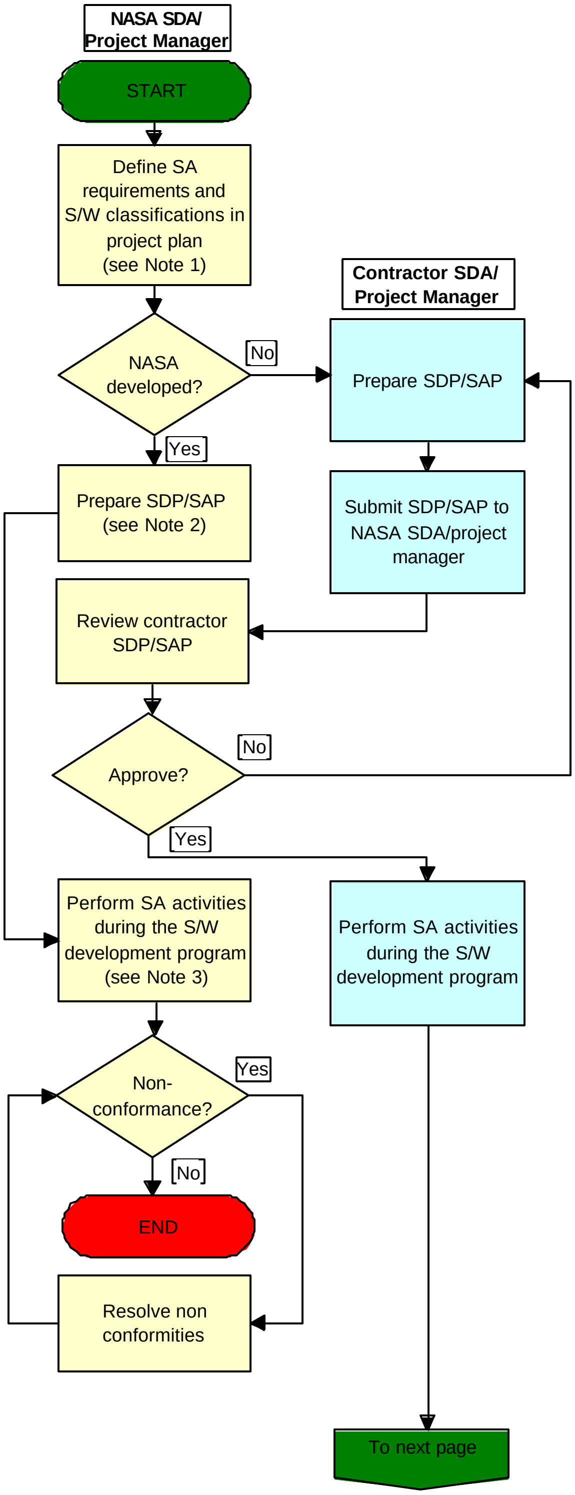
Electronically Approved by:
Assistant Director for Management Systems

CHECK THE MASTER LIST
VERIFY THAT THIS IS THE CORRECT VERSION BEFORE USE
PRINTED DOCUMENTS ARE FOR REFERENCE ONLY

DOCUMENT HISTORY PAGE

This page is for informational purposes and does not have to be retained with the document.

DATE APPROVED	ISSUE	PAGE	AMENDMENT DETAILS
1/10/01	Rev A	All	Entire document modified
See IDMS Document Master List	Rev. B	1 & 2	Corrected flowchart to remove the extra interface emanating from the last block-first page (and the first block on the second page) of NASA SDA/Project Manager so that the procedure ends when the non-conformities are resolved.


NASA SDA/ Project Manager
START
Define SA requirements and S/W classifications in project plan (see Note 1)
NASA developed?
No
Yes
Contractor SDA/ Project Manager
Prepare SDP/SAP
Prepare SDP/SAP (see Note 2)
Submit SDP/SAP to NASA SDA/project manager
Review contractor SDP/SAP
Approve?
No
Yes
Perform SA activities during the S/W development program (see Note 3)
Perform SA activities during the S/W development program
Non-conformance?
Yes
No
END
Resolve non conformities
To next page

Objectives

- To provide a planned and systematic process to verify that NASA Dryden software development processes and products conform to the approved requirements, standards and procedures
- To ensure that software acquisition and development at NASA Dryden satisfies its intended use and that the software is developed, used and maintained in accordance with sound engineering practices and principles
- To provide a tool for projects to use to determine the appropriate software assurance level-of-effort, with respect to the defined software criticality

Acronyms & Abbreviations

S/W	Software
SA	Software Assurance
SAP	Software Assurance Plan
SDA	Software Development Agent
SDP	Software Development Plan
SQA	Software Quality Assurance
SS	Software Safety

Note 1

- See Chapter 7
- Assign level A, B, or C to Flight S/W and Flight Support S/W
- Flight Systems Branch classifies Fllight S/W
- Systems Engineering Branch classifies Flight Support S/W

Note 2

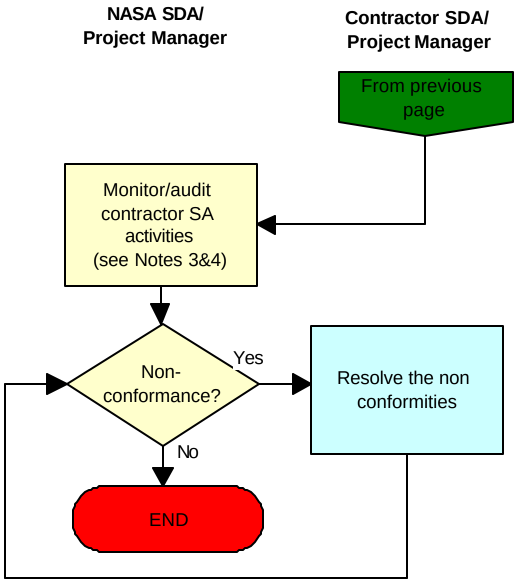
- For SDP see Attachment A
- For SAP see Attachment B

Note 3

- See Chapter 8 for Project SA activities
- Review Lessons Learned (Attachment L)

Note 4

- See Attachment I
- See Chapter 9 for SQA activities
- See Chapter 10 for SS activities


NASA SDA/
Project Manager
Contractor SDA/
Project Manager
From previous page
Monitor/audit contractor SA activities (see Notes 3&4)
Non-conformance?
Yes
No
Resolve the non conformities
END

SOFTWARE ASSURANCE

TABLE OF CONTENTS

SOFTWARE ASSURANCE

TABLES

SOFTWARE ASSURANCE

ATTACHMENTS

1 Introduction

Effective management of flight and flight-support software is a critical function at the Dryden Flight Research Center (DFRC). Consequently, it is the Dryden policy that each flight project or flight critical facility at DFRC shall apply the appropriate software management methods and processes, as specified in this Software Assurance Procedure.

Effective management of software development programs requires understanding and control of the development process through the use of Configuration Management, Test and IV&V, Software Quality Assurance and Software Safety.

This procedure describes the activities and methods available to the projects for control of the software development process. The following paragraphs are derived from the requirements of applicable NASA, IEEE, and other industry standard software development and Software Safety/quality standards, tailored to fit DFRC software development needs. Each project, through the Project Plan and/or the Software Development Plan, shall identify which of these activities apply to the project. The magnitude of each of these elements is dependent on the classification of the software and the complexity of the project.

Attachments A through H define the recommended minimum content for software documentation, identification, and configuration management. Attachment I describes the various reviews and audits applicable to the software development program. Attachments J and K define the terms and acronyms used in this procedure. Attachment L is a listing of various Software Development Lessons Learned. Attachment M is a handy Self-Evaluation Report.

2 Purpose

To establish a software management policy at Dryden Flight Research Center for all flight software or support software activities.

Primarily, two distinctive Software Assurance processes are detailed in this document, Software Quality and Software Safety. The combination of these two processes, in conjunction with a strong Software Management Organization, shall assure safe and effective software applications at DFRC.

3 Scope

This document shall apply to all levels of software for which DFRC is assigned primary responsibility for flight, ground and/or range safety and to specified support software. This includes:

a) flight software developed and controlled by DFRC,

b) flight software not developed at DFRC but maintained by DFRC,

c) flight software which is maintained for DFRC by another organization, either at DFRC or at a remote site,

d) support software which is identified as a Configuration Item (CI) in the Configuration Management Plan (CMP).

4 Responsibility

Responsibility for generating and maintaining this document rests with the DFRC Office of Safety & Mission Assurance, in coordination with the Research Engineering directorate.

5 References

The requirements contained herein address and satisfy the requirements stipulated in:

- NPG 7120.5A, NASA Program and Project Management Processes and Requirements
- NPD 2820.1, NASA Software Policies
- NASA-STD-8719.13A, Software Safety
- NASA-STD-2100-91, NASA Software Documentation Standard
- NASA-STD-2201-93, Software Assurance Standard
- NASA-STD-2202-93, Software Formal Inspections Standard
- NASA-GB-A201, Software Assurance Guidebook
- NASA-GB-A301, Software Quality Assurance Audits Guidebook
- NASA-GB-A302, Software Formal Inspections Guidebook
- NASA CM GDBK, NASA Software Configuration Guidebook
- NASA-GB-001-94, NASA Software Configuration Management Guidebook
- NASA-GB-001-95, NASA Software Process Improvement Guidebook
- NASA-GB-001-96, NASA Software Management Guidebook
- NASA-GB-001-97, Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems
- NASA-GB-002-95, Formal Methods Specification and Verification Guidebook for Software and Computer Systems

- NMI 2410.7, Assuring Security and Integrity of NASA Automated Information Resources
- DCP-P-016, Configuration Management of Flight Research Projects
- DCP-P-017, Configuration Change Process for Flight Project Critical Systems
- DCP-P-018, Discrepancy Report Process for Flight Project Critical Systems
- DOP-S-005, Software Quality Assurance Audit Procedure
- DOP-S-006, Software Safety Job Instruction
- Software System Safety Handbook, Joint Software System Safety Committee, Dec. ’99
- IEEE Std 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology
- IEEE Std 829-1983, IEEE Standard for Software Test Documentation
- IEEE Std 1012-1986, IEEE Standard for Software Verification and Validation Plans
- ISO 9001 (as further defined by ISO 9000-3, Guidelines for the application of ISO 9001 to the development, supply and maintenance of software)

Where appropriate in the attachments to this procedure, specific references are made to the NASA-STD-2100-91 Data Item Descriptions (DID) for each software document.

This procedure represents amplification and tailoring of various NASA standards. Where a conflict exists between a referenced standard and this procedure, this procedure shall take precedence.

6 Software Management Organization

Software Development Agent (SDA)

The SDA is the NASA organization or contractor responsible for software development, management, test and assurance.

When the SDA is a contractor, the contractor may use an internal organization of their choice, but shall designate an overall manager for the software development effort and shall establish responsibility for implementing the requirements of this document. The responsibilities and procedures shall be documented in a Software Development Plan (SDP) and/or Software Assurance Plan (SAP). The SDP and SAP may be combined into one document.

SDA activities:

a) Prepare the Software Development Plan to identify software design requirements, development environment and to define the analysis techniques (see Attachment A).

b) Prepare the Software Assurance Plan to establish and implement the Software Assurance program (see Attachment B).

c) Prepare the Software Configuration Management Plan or include this in the project Configuration Management Plan (CMP).

d) Appoint the Software Manager.

e) Serve as a member of the Configuration Control Board (CCB) for software related matters (Project Manager decision).

Project Manager

Project Manager Activities:

a) Define project priorities and milestones in the SDP, and ensure the allocation of resources in the form of schedule, staffing, and facilities.

b) Assure the Project Plan Request for Proposal (RFP) and the Statement of Work contains appropriate Software Assurance provisions (the Project Plan shall designate an SDA).

c) Approve all plans generated by the project, including the SDP, SAP and Configuration Management Plan (CMP).

d) Appoint the Configuration Control Board (CCB) membership.

e) Chair and preside over the CCB.

f) Act as approving authority for decisions made by the CCB.

Software Manager

Software Manager activities:

a) Take responsibility for directing and managing the Software Assurance Program (NASA-STD-2201-93, para 3.2.2).

b) Establish a reporting channel with the software provider's management that is independent of the providers project management and the software development function (NASA-STD-2201-93, para 3.2.2).

c) Act as the Technical Manager for the software.

d) Assure that the requirements specified in this document are addressed in the Project Plan and/or SDP, SAP and Software Verification and Validation Plan (SVVP).

e) Take responsibility for ensuring compliance with the software requirements of the CMP.

f) Oversee and coordinate the activity of the software design, coding, testing and documentation. For software developed by a contractor, subcontractor or vendor, this activity will include acting as the DFRC representative for dealing with the supplier on technical matters (ISO 9000-3, para 4.1.2.).

g) Establish software configuration baselines and any changes to the baseline (NASA-STD-2201-93, para 3.2.6).

h) Specify the flight software to be flown on a designated flight and document it on a Flight Media Release Form.

i) Act as a voting member of the CCB.

j) Define procedures for production of flight software media in the SDP.

k) Define procedures for control (physically and electronically) of flight software media in the CMP.

Software Engineers

Software Engineer activities:

a) Accomplish assigned tasks in accordance with the project's Software Development Plan.

b) Design, develop and test software in accordance with the approved DFRC and project procedures.

c) Implement and maintain software control in accordance with the Software Configuration Management Plan and/or the project's Configuration Management Plan.

d) Document and track all identified software problems in accordance with the project's Software Assurance Plan and the Software Verification and Validation Plan.

e) Identify potential hazards, associated with the software, throughout the software development program.

Operations Engineering

The Operations Engineering Branch has primary responsibility for overall surveillance of aircraft configuration and, in this capacity, is responsible for determining that the aircraft software has been properly generated, verified and validated, and therefore acceptable for flight.

Quality Inspection (QI)

The Quality Inspection Branch shall verify flight software and flight support software loading in accordance with established procedures.

7 Definition and Classification of Flight Software and Support Software

The level of Software Assurance activity is based on the designated criticality level of the software. The more critical the software, the more rigorous the planning, development, design, test, and assurance efforts should be.

Definitions

Flight Software

Flight Software (also known as Flight Critical or Safety Critical Software) is defined in this document as software that directly modifies vehicle operation, whether the software is installed in a system on-board an aircraft or installed in a ground-based system which modifies aircraft operation.

Support Software

Support software is defined in this document as software that could indirectly impact flight or test operations. Support software has been grouped into two categories, Flight Support Software and General Support Software. Flight Support Software is generally under configuration control, while General Support Software is generally not. Support Software that shall be under configuration control for a specific project shall be identified in the Project Plan or Software Development Plan (SDP). Specific software Configuration Items (CI's) shall be designated in the Configuration Management Plan (CMP).

Flight Support Software includes:

a) support software that directly supports flight test vehicles in flight (such as Control Room software).

b) flight simulation support software which includes real-time closed loop simulation.

c) support software that provides system test support (such as ground support systems for the aircraft and simulation).

d) support software that provides first generation post flight data processing.

General Support Software

General Support Software is defined as all support software that is not Flight Software or Flight Support Software. DFRC's control of this software is limited to assuring that the correct version of the software is used with the corresponding Flight or Flight Support Software. Instructions shall be included with the test procedures to assure that the appropriate (corresponding) support software and flight software are used together. General support software includes:

a) basic operating systems for non-flight computer equipment.

b) compilers, assemblers, linkage editors, builders, libraries and loaders required to generate machine code and to generate a complete computer program.

c) debugging programs.

d) analytic tools.

Classifications

Software at DFRC is classified according to the following levels:

Responsibility: Software classifications shall be determined by the Flight Systems Branch for "Flight Software" and by the System Engineering Branch for "Support Software".

Level A - Software failure could cause loss of life, life-threatening injury, compromise public safety, or result in loss of or substantial damage to the vehicle/system/facility.

Level B - Software failure could cause loss of flight research mission/test.

Level C - Software failure could cause inaccurate results or inefficient use of resources.

8 Project Software Assurance Activities

Table 8-1 provides project guidance on which activities, documents, reviews and audits are recommended for each software classification level throughout each phase of the software development process. Software classification levels for each project, facility, or activity shall be stated in the Project Plan and/or the Software Development Plan.

In addition to the project activities listed here, software quality and Software Safety analyses, audits and reviews will be performed, as required, by the Safety & Mission Assurance Office. An overview of software quality and Software Safety activities can be found in Sections 9 & 10. For specific step-by-step instructions see:

- DOP-S-005, Software Quality Assurance Audit Procedure
- DOP-S-006, Software Safety Job Instruction

Depending on the size and complexity of the project/software, the Project Manager or Software Manager may choose to combine some of the documents and activities listed here.

The various documents and plans should be reviewed at the end of each phase of the development process and revised as necessary. However, any deviation or modification from the baselined Software Assurance Plan shall be in the form of a formal change request and shall be accompanied by a risk analysis to identify the potential impact of the change.

Table 8-1 Project Activities and Recommended Outputs

		S/W Class				S/W Class		
Phase	**Activity**	**A**	**B**	**C**	**Associated Output**	**A**	**B**	**C**
Concept & Initiation	System Design Review	✓	✓	✓	Software Development Plan	✓	✓	✓
					Software Assurance/Safety Plan	✓	✓	✓
Requirements	Software Requirements Review/Software Specification Review	✓	✓		Software Requirements Specification (draft)	✓	✓	✓
Architectural & Preliminary Design	PDR/Architectural Design Review	✓	✓		Configuration Management Plan (draft)	✓	✓	
					Software Verification and Validation Plan	✓	✓	✓
Detailed Design	CDR	✓	✓		Software Design Description (draft)	✓	✓	
					Configuration Management Plan (final)	✓	✓	
					Software Requirements Specification (final)	✓	✓	✓
Implementation	Assist Quality & Safety Personnel with Formal Audits	✓	✓		Quality & Safety Formal Audit Reports	✓	✓	
Integration & Test	Formal Qualification Review	✓	✓		Software V&V Report	✓	✓	
	TRR/ORR	✓	✓	✓	Software Design Description (final)	✓	✓	
					Problem Resolution Reports	✓	✓	✓
					Test Reports	✓	✓	✓
Acceptance & Delivery	Functional Config. Audit	✓			Version Description Document(s)	✓	✓	✓
	Tech Brief/AFSRB/FRR	✓	✓	✓	Software Development File	✓	✓	
Sustaining Engineering & Operations	Same Activities as for Development	✓	✓	✓	Update of all Relevant Documents	✓	✓	✓

9 Software Quality Assurance (SQA)

Note: For specific audit process activities, see the Software Quality Assurance Audit Procedure available from S&MA.

Software Quality Assurance is the planned and systematic set of activities that ensures that software processes and products conform to the requirements, standards and procedures.

The DFRC Safety and Mission Assurance Office, through random monitoring and periodic audits and/or through active participation on selected projects shall:

a) assure that assurance requirements are documented and satisfied throughout all phases of the software life cycle using a strategy that emphasizes prevention, not correction (NASA-STD-2201-93, para 3.1.1),

b) assure that configuration documentation meets requirements,

c) assure that software is built to approved configuration documentation,

d) validate configuration prior to selected tests,

e) assure compliance to documented inspection procedures (NASA-STD-2202-93, para 3.2.3),

f) witness tests and validate results,

g) assure that all discrepancies are documented, and provide a process to assure corrective action is taken and completed (NASA-STD-2201-93, para 3.1.3),

h) assure that correction of discrepancies prior to acceptance of a configuration item.

The DFRC Safety and Mission Assurance Office shall conduct software management audits to assure the quality system for a project's software continues to be suitable and effective (ISO 9000-3, para 4.1.1.3). Safety and Mission Assurance will normally review each project's software management procedures prior to the PDR, the CDR and the Flight Readiness Review. Depending on the complexity of the software development program, further audits may take place throughout the software life cycle.

Software Assurance Life Cycle Phases, Activities and Outputs

The software life cycle is the period of time that starts when a software product is conceived and ends when the software is no longer available for use. The software life cycle traditionally has eight phases: concept/initiation, requirements, architectural design, detailed design, implementation & unit testing, integration and test, acceptance and delivery, and sustaining engineering and operations. Where software reuse is a consideration, the software, documentation and records may be retained under configuration control beyond the expected development life cycle.

The Preliminary Design Review and the Critical Design Review are applicable to all flight and flight support software developed for DFRC, either at DFRC or at the contractor's facility. Some of the reviews may be combined to form a review with a broader scope. Other reviews and audits listed below will be included or eliminated based on software criticality, as defined in the project plan. Table 8-1 provides a matrix associating typical activities and outputs to software criticality.

The following paragraphs provide more detail of the primary software life cycle activities and outputs (as listed in Table 8-1). More specific requirements can be found in the attachments to this document (as referenced in the following paragraphs). In addition, Attachment I provides detailed requirements concerning the various reviews and audits outlined below.

System Design Review (SDR)

The SDR is held to evaluate the optimization, traceability, correlation, completeness and the risk of allocated requirements, including the corresponding test requirements in fulfilling the functions defined in the operation concept.

Software Development Plan (SDP)

The SDP identifies all activities and processes necessary to develop the software. Documentation requirements, architectures, development environments and test process are all included. An outline for the SDP is shown in Attachment A.

Software Assurance Plan (SAP)

The SAP identifies Software Quality and Software Safety activities and the organizations responsible for those activities. It shall include provisions for defining standards for verification and validation requirements and test plans, reviews and audits, and problem reporting and corrective action. Attachment B shows an outline for a SAP.

Configuration Management Plan (CMP)

The CMP shall document the methods used for identifying project configuration items, including the software product items, controlling and implementing changes, and recording and implementing change implementation status.

The software CMP is further discussed in paragraph 12.1 and in Attachment H.

Software Requirements Review (SRR)

The SRR is held to assure the adequacy of the requirements stated in the Software Requirements Specification (see Attachment C). It is conducted to assure adequacy, technical feasibility and completeness of the requirements. Its purpose is to establish the allocated baseline for preliminary software design.

Software Requirements Specification (SRS)

The SRS identifies the objectives of the program, its environment, the configuration needed for its operation and the resources required for its support. The SRS will be reviewed by the Software Manager and approved by the Project Manager. An outline for the SRS is shown in Attachment C.

Preliminary Design Review (PDR)

The PDR is held to evaluate the technical adequacy of the preliminary design and to review the Software Development Plan. At this time, a determination can be made as to the adequacy of this document to satisfy the Software Requirements Specification and Software Design Description. Unless software is the only product, the software PDR may be held as part of the project PDR.

The PDR is also referred to as the Architectural Design Review in several NASA Standards. The term PDR is used in this procedure to mean both events.

System (or Software) Verification and Validation Plan (SVVP)

The SVVP may be included in the system-level plan. It shall describe the methods to be used to ascertain that the software products meet the specification and the design falls within limits of functionality. The SVVP shall describe methods to be used:

a) to verify that the requirements of the Software Requirements Specification (SRS) and the Software Design Description (SDD) have been implemented,

b) to validate that the software, when executed in a representative environment or on the actual hardware, complies with the specification expressed in the SDD,

c) to assure that the system will operate at off-nominal conditions in an acceptable manner,

d) to assure verification and validation independence, as required.

Attachment F lists verification, validation and test documents that may be required by the different projects.

Critical Design Review (CDR)

The CDR concludes the initial specification development phase of the software development program. The CDR is held to determine that the design is complete. Typically, the following documents are review: SDP, CMP, SAP, SRS & SVVP. Unless software is the only product, the software CDR may be held as part of the project CDR.

Software Design Description (SDD)

The Software Design Description is a technical description of how the software will meet the requirements set forth in the SRS. Its most important function is to describe a decomposition of the system as a whole into components (subsystems, segments, units/modules, etc.) that are complete and well bounded. The SDD will be reviewed by the Software Manager and approved by the Project Manager. An outline of the SDD is shown in Attachment D.

Test Readiness Review (TRR)

This review is held to insure that all systems are ready for acceptance testing. This review may be a contractor's internal review if testing is to be at the contractor's facility.

Software Development File

During development, the software development file is maintained and kept up to date by the developers and testers. At the end of the project, the software development file shall be included in the Software Library.

Formal Qualification Review (FQR)

The FQR is a formal review of test reports and test data generated during the formal qualification of a new Configuration Item (CI) (software or hardware). It is performed to ensure that all tests required by quality assurance provisions of the development specification(s) have been accomplished, and that the CI performs as specified by the requirements of the developmental specification(s).

Software Verification and Validation Report

Results from the Software Verification and Validation tests shall be described in a Software Verification and Validation Report. See Attachment F for specific requirements.

Functional Configuration Audit (FCA)

The FCA is the formal examination by the user organization of the functional characteristics based on specifications, test data, or other descriptive criteria of a configuration item to verify that the item has achieved the performance specified in the design documentation.

Version Description Document (VDD)

A VDD shall be written for each particular version of software released, baselined or modified. Attachment E provides an outline for a VDD.

Airworthiness and Flight Safety Review Board (AFSRB)

This review is generally held prior to first flight or first test of a major new configuration. The Dryden AFSRB will perform a final review with (at the boards option) the assistance of an Ad Hoc team normally referred to as an FRR committee, and specifically selected for this purpose. Hardware and software shall be reviewed together to determine if the combined system is ready for flight/test.

Informal Reviews

These reviews may be held on a special basis for the purpose of monitoring and assessing progress.

10 Software Safety (SS)

Note: Safety & Mission Assurance personnel should refer to the Software Safety Job Instruction for specific step-by-step instructions.

Software Safety shall be an integral part of the overall System Safety and software development efforts. It is the objective of the Software Safety effort to assure that safety is considered throughout the software life cycle. Therefore, Software Safety activities take place in every phase of the system and software development life cycle beginning as early as the concept phase and on through to the operations and sustaining engineering phase. Up-front participation, analyses, and subsequent reporting of safety problems found during the software development life cycle facilitates timely and less costly solutions.

Step one of an effective Software Assurance process should start with the completion of the Self-Evaluation Report. Preferably, this report should be presented to the SDA by the S&MA Office and completed early in the software development program; however, it may be used at any stage of the program to assess the overall level of compliance with NASA Software Assurance standards.

See Attachment M for a copy of the Self-Evaluation Report.

Software Safety requires a coordinated effort among all organizations involved in the development of the software. This includes Program Managers, hardware and software designers, safety analysts, quality assurance, and operations personnel. Those conducting the Software Safety effort shall also interface with personnel from disciplines such as reliability, security, Independent Verification and Validation (IV&V) (when applicable), and human factors.

The purpose of the Software Safety process is to ensure that software does not cause or contribute to a system reaching a hazardous state; that it does not fail to detect or take corrective action if the system reaches a hazardous state; and that it does not fail to mitigate damage if a failure occurs.

The Software Safety process shall:

a. assure that the system/subsystem safety analyses properly identify software that is considered safety-critical. Any software that has the potential to cause a hazard or is required to support control of a hazard, as identified by safety analyses, is safety-critical software.

b. assure that the system/subsystem safety analyses clearly identify the key inputs into the Software Requirements Specification (e.g., identification of hazardous commands, limits, interrelationship of limits, sequence of events, timing constraints, voting logic, failure tolerance, etc.).

c. assure that the development of the Software Requirements Specification includes the Software Safety requirements that have been identified by Software Safety analysis.

d. assure that the software design and implementation properly incorporate the Software Safety requirements.

e. assure that the appropriate verification and validation requirements are established to ensure proper implementation of the Software Safety requirements. This explicitly includes an assessment of the scope and level of IV&V to be planned and implemented based on the level of criticality and risk of the software application. A statement will be made in either the program/project plan or the software development plan as to the level of IV&V to be accomplished.

f. assure that test plans and procedures will satisfy the intent of the Software Safety verification requirements.

g. assure that the results of the Software Safety verification effort are satisfactory.

11 Software Verification and Validation

Software Verification and Validation are a subset of system Verification and Validation. Software Verification and Validation is concerned with ensuring that software being developed or maintained satisfies functional and other requirements, and that each phase of the development process yields the right products. Verification and validation activities shall be performed during each phase of the software life cycle.

Independent Verification and Validation (IV&V) is a process whereby the products of the software development life cycle are independently reviewed, verified and validated by an individual or organization that is neither the developer or the acquirer of the software. When IV&V is required, the IV&V activities duplicate the verification and validation activities step-by-step during the life cycle (with the exception that the IV&V agent does no informal testing). If there is an IV&V agent, formal acceptance testing may be done only once by the IV&V agent. In this case, the developer will do a formal demonstration that the software is ready for formal testing.

Verification is the process of assuring the software meets the specification. Typical elements included in the verification process include code walk-through, traceability analysis, unit/module level tests, integration tests and hardware-in-the-loop tests.

Validation is the process of assuring the design (including the hardware and full system) works for nominal and off-nominal conditions in a representative environment. Off-nominal conditions can include variations in performance, failures and design uncertainties. Typical validation tests include hardware-in-the-loop and/or aircraft-in-the-loop tests, failure modes and effects tests and frequency response tests.

Software verification and validation requirements (including the requirement for independence) shall be approved by the Project Manager and described in a Software Verification and Validation Plan (SVVP).

Attachment F contains detailed requirements for software Verification and Validation. This attachment shall be used as guidance for verification and validation and for IV&V.

12 Configuration Management

Configuration Management (CM) is the process of identifying and defining the configuration items in a system, controlling the release and revision of these items throughout the system life cycle, recording and reporting the status of configuration items and requests, and verifying the completeness and correctness of configuration items.

Configuration Management Plan (CMP)

The CMP shall describe how software control is to be implemented. The following shall be addressed:

Software Configuration Identification

a) Configuration Item

The CMP shall present criteria for selecting and identifying software configuration items (CI). A CI can be a subsystem, such as a vehicle management system, or can be a function within the subsystem, such as navigation.

b) Software Configuration Baseline (Software Release)

The CMP shall present guidelines for specifying software configuration baselines. Software control begins after the first baseline has been established.

Software Configuration Control

The CMP shall describe procedures and forms to be used for configuration control. These should be the same forms used for the hardware unless good reason dictates different forms. It shall define:

a) procedures and forms to request, review, and approve changes to the configuration,

b) procedures and forms to report and resolve a discrepancy,

c) procedures and forms to document and report that the changes have accomplished the desired objectives.

Software Identification

The CMP shall define procedures to ensure that in-house and deliverable software products are properly identified and consistent with approved documentation. The CMP shall define procedures to ensure in-house and deliverable software products submitted for testing are the correct version and incorporate authorized changes (see Attachment G).

Software Configuration Status Accounting

The CMP shall define specific project requirements for status accounting. If none are defined, the DFRC Configuration Control Office shall issue standard status reports (see Attachment H).

Media Control of Flight Software and Flight Support Software

Flight Software Production

a) When responsibility for production of the flight software rests with DFRC, production and physical control of the software is the responsibility of the DFRC Flight Systems Branch. A procedure for production and physical control of flight software shall be prepared by the responsible engineer.

b) When responsibility for production of the flight software rests with the contractor, DFRC shall be provided with the contractor's plan to assure conformance with the DFRC's requirements for production, identification and control of the flight software. After delivery of the flight software to DFRC, physical control will be the responsibility of the DFRC Flight Systems Branch.

d) When software is reproduced onto a tape, disk or chip, all flight software media shall be identified and physically labeled at the time of production (see Attachment G).

e) Prior to installation on the aircraft, a Version Description Document (VDD) shall be produced. The VDD shall contain a Flight Release Form and be attached to a Configuration Change Request (CCR) requesting installation on the aircraft (see Attachment E).

Flight Support Software Production

a) When the responsibility for production of the flight support software rests with DFRC, production and physical control of the software shall be assigned to a cognizant branch. A procedure for production and physical control of flight support software shall be prepared by the assigned branch.

b) When responsibility for production of the flight support software rests with the contractor, DFRC shall be provided with the contractor’s plan to assure conformance with DFRC’s requirements for production, identification and control of the flight support software. After delivery of the flight support software to DFRC, physical control will be assigned to a cognizant branch.

c) When software is reproduced onto a tape, disk or chip, all flight support software media shall be identified and physically labeled at the time of production (see Attachment G).

Flight Software Installation

a) A procedure for flight software installation into the aircraft computer and for verification of correct loading shall be written by the SDA.

b) Flight software for a specific flight or block of flight(s) shall be designated by the Software Manager on a Flight Release Document.

c) Flight software installation shall be accomplished utilizing the DFRC Work Order in accordance with Process Specification 00-4 (or currently approved method).

d) Quality Inspection shall verify the correct flight software is loaded for the specified flight, according to approved procedures.

f) After the flight software has been installed in an aircraft computer and after verification of correct loading, no patches or tape overlays are allowed unless performed using an approved procedure and accompanied by a DFRC Work Order.

Flight Support Software Installation

a) A procedure for loading the flight support software into the computer and for verification of correct loading shall be written by the responsible facility.

b) Flight support software for a specific flight or block of flights shall be designated by the Facility Manager on a Flight Support Release Document.

c) After the flight support software has been installed in the computer and after verification of the correct loading, no patches or tape overlays are allowed.

Software Development File

Each project shall establish and keep up to date a software development file. At minimum, this file shall contain copies of all engineering notes, unit/module and integration level test procedures or test code and results of testing performed by the developer(s). Specific content shall be defined in the SDP.

The software development file is a living document, which normally follows the software through the various phases of the software development process. During development, the software development file is maintained and kept up to date by the developers and testers. At the end of the project, the software development file shall be included in the Software Library.

Software Library

All projects shall establish a Software Library. The purpose of the Software Library shall be to maintain the current status of the software program and to serve as an archive, keeping a record of the software history. The following shall be included in the Software Library:

a) Archive for formal released software and documentation.

b) Updated status of software.

c) Procedures generated to maintain the software.

d) Copies of the Software Requirements Specification, Software Design Description, Software Development Plan and other pertinent documents accepted under change control.

g) All autocode and autotest software.

13 Contractor, Subcontractor and Vendor Control

Procedures for assuring that all software, documentation and programming materials procured from contractors, subcontractors, etc. conform to the software specifications and to all applicable DFRC documents shall be included in the SAP (see Attachment B). The Software Manager will normally be assigned as the DFRC representative for dealing with the software supplier on technical matters (ISO 9000-3, para 4.1.2.).

14 Records

Each project shall define and keep up to date all required records. Records shall be defined in the SDP and SAP. At minimum, these records should include copies of all walk-throughs, inspections, design reviews and audits.

15 Training

Personnel developing and implementing the software assurance process and project personnel involved in the software development program (as determined by the Project Manager) shall be trained and/or experienced in the software assurance process. The Office of Safety and Mission Assurance provides the Software Assurance training program.

Attachment A – Software Development Plan (SDP)

(NASA-STD-2100-91: NASA-DID-M200)

The Software Development Plan shall be approved by the Project Manager.

Software Development Plan

The Software Development Plan (also called the Software Management Plan) defines the software development process and identifies all activities necessary for implementing a software project. The Project Manager shall approve the SDP. The following items shall be addressed:

Software Development Methodology

The software development methodology shall be identified. Traditional techniques include structured programming, while newer techniques include object oriented programming or rapid prototyping techniques. Standards and practices shall be defined as well as coding conventions. Documentation requirements will also be defined in this section.

Software Design Requirements

Overall goals and requirements shall be defined, including system components and their interfaces. This section shall define the detailed requirements of the documentation to be developed defining the detailed specifications.

Development Environment

The development environment shall be defined in the software development plan. The development environment is more than the software language used for the system. It includes design and autocode systems (such as MatrixX or MatLab). It also includes any debugging environment tools and the operating system definition (such as VXWorks or a similar system).

Analysis Techniques

Any analysis techniques or tools that will be used shall be identified. Tools that identify software complexity or flow, provide visualization of the software, or that identify potential trouble spots shall be described.

Test Philosophy and Process

The test philosophy to be used shall be defined in this section.

Traditional test techniques include unit/module level testing, integration testing, verification and validation. Also included in traditional techniques are code walk-throughs, requirements traceability evaluations, verification testing and validation testing. The intent of the traditional testing techniques is to check every path and element in the software

Each project will determine the test environment requirements. Test requirements will be defined in the Software Development Plan. Any deviations from the classical test techniques for flight critical software shall demonstrate that the resulting test effort meets the intent of this procedure in providing safe software.

Security (also see NMI 2410.7, Assuring the Security and Integrity of NASA Automated Information Resources)

The level of security required to safeguard the code and documentation (if applicable).

The SDP shall:

a) Define the development, integration, and testing activities for flight and support software.

b) Define the facilities, personnel, organizational structure, training requirements, supplies, services, and other resources required to design and test of the software.

c) Define the activity flow, schedules, and milestones for accomplishing planned activities, including the generation of all required documents.

d) Define the approach for identifying common software for cost effective utilization of existing or planned software and its documentation. Define the process to assure reused software (if any) is appropriate for this development.

e) Identify the standards, practices, and conventions to be used.

f) Identify the programming language(s) and any unique uses of the language.

g) Define the development environment.

h) Define the process for software integration, including:

1) Test build-up sequence.

2) Procedures for coordinating/controlling interfaces between each unit/module.

4) Unit/module build-up sequence.

5) Integration testing and evaluation.

i) State the data requirements, including inputs and outputs along with their source (scaling if required) and units/modules.

j) Describe the process for preliminary software coding and debugging.

k) Define the design approach used to satisfy software, system, operation and human performance requirements.

l) Define the test philosophy and the verification and validation requirements.

m) Define the facilities used for development and testing.

n) Include the preliminary Software Assurance Plan (if not in a separate document).

o) Specify the design characteristics at a program level in terms of sequencing control, displays, error detection and recovery, I/O control, diagnostics, timing characteristic, memory size and library utilization.

p) Describe any expected or potential hazards related to the software.

q) Describe any security issues/requirements associated with the software.

Attachment B – Software Assurance Plan (SAP)

(NASA-STD-2100-91: NASA-DID-M400)

The SAP (also called the Software Quality Assurance Plan) will be approved by the Project Manager and shall include the following:

a) Purpose

This section shall define the purpose and scope of this document. It shall list the name(s) of the software configuration items and their intended use. The SAP will include procedures for assuring all software, documentation, and programming materials procured from contractors, subcontractors, etc., conform to the software specifications and to all applicable DFRC documents.

b) Reference Documents

This section shall describe a complete list of all referenced documents.

c) Management

Organization, tasks and responsibilities shall be defined.

1) Organization

A description of the organizational structure and each major element of the organization that influences the quality of the software shall be defined. This shall include description of delegated responsibilities to each element and organizational dependencies or independence's.

2) Tasks (refer to DOP-S-005, Software Quality Assurance Audit Procedure)

The tasks associated with the portion of the software life cycle covered by this plan shall be described with emphasis on quality assurance activities.

3) Responsibilities

Key personnel responsible for publication, distribution, maintenance, and implementation of this plan shall be defined.

d) Documentation

This section shall:

1) Identify the documentation governing the development, verification and validation, and use and maintenance of the software.

2) State how the documents are to be checked for adequacy, including any reviews or audits.

e) Standards, Practices and Conventions

This section shall:

1) Identify the standards, practices and conventions to be applied and the associated phase of the life cycle.

2) State how the standards, practices, and conventions will be monitored for compliance.

3) Include documentation standards, coding standards, comment and header standards.

f) Reviews and Audits

This section shall define the technical and managerial reviews and audits to be conducted and state how they will be accomplished.

h) Software Safety (refer to DOP-S-006, Software Safety Job Instruction)

This section shall describe the overall approach to be used to perform the safety assurance activities for the software. Describe the specific activities with respect to analysis and review of specific aspects in terms such as:

a. Hazards
b. Fault tolerance
c. Safety criteria such as fail-safe, fail-soft, and fail-operational

h) Software Configuration Management

Include a description of the planned configuration management process, referencing the Configuration Management Plan.

i) Nonconformance Reporting and Corrective Action

This section shall address the following:

1. Nonconformance detection and reporting procedures.

2. Nonconformance tracking and management procedures.
3. Nonconformance impact assessment and corrective action procedures.
4. Interfaces to the Configuration Management process.

j) Tools, Techniques and Methodologies

Identify and describe any special tools, techniques and methodologies employed that support Quality Assurance.

k) Media Control

Define the methods and facilities used to maintain and store controlled versions of the software and the interactions with Quality Assurance.

Attachment C – Software Requirements Specification (SRS)

(NASA-STD-2100-91: NASA-DID-P200)

The SRS will be reviewed by the Software Manager and approved by the Project Manager. The following items shall be included:

a) Task Statement:
- A description of the program and its objectives.
- Identification of software as safety-critical, if applicable.

b) Functional Description:
- A functional description of the required software capabilities including the configuration needed for its operation.
- Define:
  1) The function to be performed by the software.
  2) The inputs.
  3) The outputs.

c) System Requirements
- Identify the computer system characteristics (computer hardware and its operating system) as well as any special peripheral equipment. Estimate memory size, (both program memory and system memory) and speed requirements.
  1) Interface between different elements of the system and between the system and its environment.
  2) General environment in which the system is to be used.
  3) Identification of support software, hardware and simulations necessary to accomplish software development.
  4) Requirements of language selection.
  5) Overall timing and memory requirements.
  6) Modifications required to elements not supplied with the system, but which will interface with it.
  7) Milestones on the critical path.

d) Implementation Schedule, Resource Estimate and Plan
- A project schedule of the system showing the phases of the development, the level of effort and the responsibilities of the organizations involved.

Attachment D – Software Design Description (SDD)

(NASA-STD-2100-91: NASA-DID-P400)

The SDD will be reviewed by the Software Manager and approved by the Project Manager.

This document shall:

a) Define software input/output

b) Define all software related functional, interface, and error recovery requirements.

c) Define software performance requirements in measurable/quantifiable terms and establish acceptance criteria for each requirement (including preliminary memory and timing requirements).

d) Identify key assumptions and constraints in defining external software interfaces requirements, such as sensor data, actuator commands, data rates, computational rates, etc.

e) Identify data flow and control flow.

f) Identify operational requirements that impact software design.

g) Define database structure, if required.

h) Identify equations, constants, functional flow charts, and key assumptions and constraints for which unique or critical formulation requirements are imposed.

i) Define unique software assurance requirements in terms of:

   1) Numerical accuracy characteristics,

   2) Special conventions and interfaces (paying close attention to sign convention and units/modules),

   3) Self-documenting aspects of the code.

j) Define computing resources including the type of computer system, the size of main memory, auxiliary storage, number of channels, etc.

k) Provide a detailed description of special data processing requirements or instructions for special format to accommodate testing, recording, simulation, necessary procedures and system growth.

Attachment E – Version Description Document (VDD)

(NASA-STD-2100-91: NASA-DID-P500)

The Version Description Document documents and identifies a new software release. It shall include:

a) Unique Release Number,
A unique version number shall be assigned to the software release and included in this document and on the media release form and media.

b) Summary of Change(s),
A description of all changes made between the new release and the previous release. The version used as a baseline shall be identified in this section.

c) Summary of Configuration Control Documentation,
A summary of all configuration control documentation associated with this release and their status shall be included in this section.

d) Test Summary,
A summary of testing completed and the results shall be included. The test report and test plan shall be referenced.

e) Hardware Configuration,
The configuration of the system's hardware and any interfaces shall be defined.

f) Media Release Form,

An unsigned media release form shall be included.

g) Operational Considerations,
Any operational considerations shall be described, including any restrictions.

h) Hazards,
Any hazards associated with the software shall be included in this section.

Attachment F – Software Verification and Validation (V&V)

(NASA-STD-2100-91: NASA-DID-A000)

The Software Verification and Validation requirements shall address both the software initiation and development phases and the operational phase of the system.

Software Initiation and Development Phases

Two documents are required at this point in the software development project; the Software Verification and Validation Plan and the Software Verification and Validation Report.

Software Verification and Validation Plan (SVVP)

The SVVP shall describe the process and methods to be used for assuring the software contains no major errors. It is understood that it may not be cost effective to develop and test a system until the software is error free. The purpose of the test process is to reduce the risk that any major errors exist that can cause loss of life, vehicle, or mission to highly improbable. The test plan will describe the methods, tasks, and criteria for testing of the system. It shall address two (2) areas; verification and validation.

a) Verification

Verification is the process of assuring that the software or system meets the requirements as defined in the Software Requirements Specification and the Software Design Description. Verification may contain a number of different elements, including:

1) Requirements Traceability Matrix,

This element ties the test to the requirements in the SRS and SDD. This will assures that all the requirements have been addressed, or if not, provides an explanation of why not.

2) Code Walk-Through,

This element is a visual inspection of the source code by an independent person (someone who did not write the code) knowledgeable about software and the project, or a group of such people.

3) Unit/module Level Testing,

This element tests each path, gain or calculation of the software at the lowest level. This testing is often done on a machine other than the target processor.

4) Integration Testing,

This element is the testing performed when the software is hosted on the target system. The majority of paths, gains and calculations are tested at this time. Other tests will include timing or through-put checks, memory usage and external interface checks.

5) Closed-Loop Tests,

These tests are performed on a system as representative of the actual flight system as is practical. They can contain time history, frequency and mode transition tests.

b) Validation

Validation is the process of assuring the design will operate in the intended environment. Validation testing includes nominal and off-nominal conditions on as system as representative of the actual flight system as practical. Elements of validation testing include:

1) Off-Nominal Conditions,

Time histories, frequency response and mode transition tests are some of the tests performed with deviations in the models or conditions. It is assumed that the actual flight environment will not exactly meet the test environment, therefore deviations and variations on the nominal conditions are run. Tests are performed in conditions the vehicle is not expected to be operating at.

2) Failure Modes and Effects Tests,

Failures are introduced into the system to assure the system performs as expected and with no adverse effects. These tests are performed to assure that all worst case conditions are known and that the system will gracefully degrade with failures. Typically, all input and output to the system are failed in multiple ways, such as open, hardover or biased.

3) Pilot and Stress Tests,

These tests allow the pilot and tester to stress the system. Emphasis should be placed on maximizing computer speed and memory requirements while simultaneously maximizing data bus throughput. Maneuvers representative of test points at the edge of the test envelope are performed and failures are introduced as predicted by hazard analysis. In simulation, maneuvers outside the planned test envelope may be flown in order to maximizing computer and data bus throughput under extreme conditions.

4) On-Aircraft Tests,

These tests are performed on the actual flight vehicle to verify analysis and other test results. These tests can include functional checkout, ground vibration tests, structural mode interaction tests, electromagnetic interference tests (EMI), and combined system tests.

Note: Independent Verification and Validation

At this writing, NASA Policy Directive NPD 8730.XX, Software Independent Verification and Validation, was in a draft format and, therefore, the requirement for IV&V was as yet undefined. This document will be updated accordingly when NPD 8730.XX is released

.

If Independent Verification and Validation is specified in the Project Plan, Software Development Plan or Software Assurance Plan, the IV&V activities shall be performed by an individual or organization which reports directly to the project manager, and that is other than and independent of, the software developer. The software developer shall work closely with the IV&V provider and shall facilitate access to all software and software assurance products.

Software Verification and Validation Report (SVVR)

Results from the Software Verification and Validation tests shall be described in a Software Verification and Validation Report. The report shall contain:

a) Configuration,

The configuration under which each of the tests was performed shall be described in detail. If the configuration changed during the course of the testing an explanation for the change will be provided. This will include all tools used to test the software, the test environment, as well as the software itself.

b) Deviations,

Any deviations from the test plan and the rationale for the deviation shall be described.

c) Results,

Test results will be described, particularly any unexpected results of discrepancies. The reason for the discrepancies and/or unexpected results shall be described. An annotated copy of the test procedures may suffice as the test results document.

d) Operational Impacts,

Any operational impacts resulting from the testing shall be described in this section.

e) Hazards,

Any hazards associated with the software will be included in the report; particularly any hazards generated from the tests themselves.

Operational Phase

In this phase a Regression Test Plan should be written. The Regression Test Plan will define the tests that must be performed any time a change is made to the software system (this will include the host processing unit). A standard set of tests should be defined prior to any changes. This plan will be contain the same elements as the Software Verification and Validation Plan. Specific tests to address the changes will be defined at the time of the change.

A Regression Test Report for the new version of software shall be written and shall address the same elements as the Software Verification and Validation Report.

All changes to the software in this phase of the project will follow the same configuration control as defined in the CMP. The CMP shall address all phases of the software development project.

Attachment G – Software Identification

All flight software and flight support software media shall be identified and physically labeled at the time of production. A unique release number shall be assigned to it and when possible, denoted on the media itself.

All software released outside the developing organization shall be uniquely identified by part number, serial number or version number. The project CMP shall define the numbering system to be used for all project software releases.

The Software media is authorized by supporting documentation including:

a) CCR for changing or installing the software.

b) VDD defining the released software.

c) Media release document.

Labeling of the software media and documentation shall include:

a) Version Number.

b) Applicable CCR's, DR's and System Test Reports (STR's), if any.

c) Date of release and flight number of release.

The project's ground maintenance procedures shall include a means of identifying the software version loaded into the test vehicle. For software loads identifiable from the cockpit, pre-flight procedures or flight data cards shall include a process for the flight crew to identify and record the test software version.

Attachment H – Software Configuration Management Plan

(NASA-STD-2100-91: NASA-DID-M600)
(also see DCP-P-016, 017 & 018)

The purpose of the Software Configuration Management Plan is to define the configuration management process for the software and its associated products.

The Software Configuration Management Plan shall discuss the various activities and summarize the flow of information and products developed within the configuration management structure. Include a description of the process of incorporating products received into the baselines maintained by the preparing organization. Be sure to address any access restrictions.

Describe the configuration management information flow in terms of a flow chart or similar graphic. Show each review and control board in the context of the information flow. Summarized change control reports to be generated and how they are to be tracked.

If appropriate, describe special considerations for security that are to be supported by configuration management, such as analyzing proposed changes for adverse effects on security or recording each access to secure data under configuration control.

When utilized, the Configuration Control Office at DFRC shall prepare status reports containing the following items:

a) Historical configuration lists (total number of configuration control forms and documents, such as CCR's, DR's, STR's, etc.).

b) Status configuration control forms and documentation (such as lists of configuration control documents that have not been designated as complete by the Configuration Control Boards).

c) Current baseline (number of the software release issued or last designated baseline, including a list of configuration control documents implemented since the last baseline).

Attachment I – Reviews and Audits

(NASA-STD-2201-93)

The various software reviews are described in this section and are applicable to software developed for DFRC. The assumption has been made that software and hardware will be treated as a system and reviewed together and that baseline management will be used to track the system configuration.

The SDA shall be responsible for assigning cognizant project personnel to perform the software presentations.

System Design Review (SDR) (Concept & Initiation Phase)

The presenters are asked to demonstrate that the current Software Development Plan is adequate for continuing the program definition and that development activities with reasonable assurance that major revisions will not be necessary.

a) Demonstrate that the software design is consistent with the system requirements as defined in the System Requirements Specification.

b) Demonstrate that the Hardware/Software decomposition is effective and efficient.

c) Present test plans that will adequately demonstrate performance and conformance.

d) Present and evaluate the Software Assurance Plan (SAP).

Software Requirements Review (SRR)/Software Specification Review(SSR) (Requirements Phase)

The presenters are asked to show the adequacy, technical feasibility, and completeness of the requirements in the draft Software Requirements Specification.

a) Establish the need for the software and identify its objectives and its user and system environment, the configuration needed for its operation, and the resources required for its support.

b) Demonstrate that the remaining development activities may proceed under assurance that major revision of technical and management objectives will not be necessary.

c) Present evidence that the software and its use are feasible with respect to technical considerations, safety, staffing, schedule, and development costs.

d) Provide variance estimates or bounds for all planned resources to be experienced.

e) Review and demonstrate that the software requirements specified are adequate to identify the objective of the program, its environment, the configuration needed for its operation, the resources needed for its support, and safety.

Preliminary Design Review/Architectural Design Review (Architectural & Preliminary Design Phase)

The presenters are asked to perform the following tasks:

a) Identification of any safety critical software systems.

b) Present the updated Software Development Plan, which contains updated and detailed project staffing, schedule, and development cost estimates. The software design and development process shall also be presented as well as an implementation plan, including work priorities.

c) Present the program top-down software hierarchic functional definition and design architecture, using data flow diagrams, flowcharts, and explanatory narrative. Show that the program definition and design architecture are technically feasible and compatible.

d) Present implementation testing criteria, plans and procedures and show that they will fulfill requirements to establish program correctness.

e) Present preliminary plans for software integration.

g) Identify required software support and external program interfaces, and evaluate their impact on software delivery.

h) Review the draft Configuration Management Plan with emphasis on software control.

Critical Design Review (CDR) (Detailed Design Phase)

The presenters are asked to perform the following:

a) Demonstrate that the design is complete and acceptable for both Flight and Flight Support Software.

b) Demonstrate that the technical requirements have been satisfied and that all exceptions or remaining problems, and the plan for disposition of such items, has been identified.

c) Demonstrate that a detailed plan exists for the remaining elements of software development.

d) Demonstrate that software coding initiated is consistent with the Software Design Description (SDD).

d) Present evidence that the detailed Software Requirements Specification is complete.

e) Present evidence that the Configuration Management Plan is complete.

f) Present evidence that all hazards identified with the software have been addressed.

Test Readiness Review (TRR)/Operational Readiness Review(ORR)/Formal Qualification Review (Integration & Test Phase)

This review will be held to assure that:

a) All software is ready for acceptance testing.

b) Test plans and procedures adequately demonstrate performance and conformance (all validation, verification, and test requirements and plans shall be reviewed).

c) Verify that all V&V Reports, Problem Resolution Reports and Test Reports have been reviewed and resolved.

Formal Qualification Review (FQR) (Integration & Test Phase)

This review will certify that the program performs within acceptable limits of specified behavior. The FQR shall include:

a) A presentation of the program performance requirements, a set acceptance criteria relative to these requirements, and the means used to validate the measured performance relative to the Verification and Validation Plan.

b) Evidence that measured performance satisfies acceptance criteria.

c) Final Software Design Description (SDD), Software Verification and Validation Report (SVVR), and annotated code listing all approved audited by the CCB for completeness and conformity with project standards.

d) A final project management report delineating total staffing, schedule, and development cost figures. These should be broken down into detailed resources expended in definition, design, coding, checkout, testing, integration, and documentation areas.

Functional Configuration Audit (FCA) (Acceptance & Delivery Phase)

The Functional Configuration Audit includes:

a) Review of development test plans and procedures.

b) Review of all test results for compliance with requirements.

c) Listing of required tests not performed.

d) Listing of deviations and waivers.

e) Delivery of the Version Description Document(s) (VDD)

Flight Readiness Review (FRR)/Flight Readiness Review (FRR)/Tech Brief (Acceptance & Delivery Phase)

This review shall be held prior to first flight. The reviewing committee shall certify that all systems, both hardware and software and the vehicle are ready for flight. The review shall include:

a) A presentation of program performance obtained from the integrated system tests and the Verification and Validation tests.

b) Evidence that measured performance satisfies acceptance criteria.

c) Listing of any requested tests not performed.

d) Listing of any deviations and waivers.

e) Any hazards associated with the software.

Informal Reviews

These reviews may be held, as needed, for the purpose of monitoring progress and supervising developments.

Attachment J - Definitions

AIRWORTHINESS & FLIGHT SAFETY REVIEW BOARD (AFSRB) - The AFSR is the last review held prior to a first flight of a major configuration change or new vehicle. The purpose of this review is to certify that all systems, both hardware and software, and the vehicle are ready for flight.

ALLOCATED BASELINE - A statement of the detailed functional and design requirements for configuration items, sufficient to start detailed design. This is the initial decomposition of the system specification and is documented by the Hardware/Software Specifications Document.

ALLOCATED CONFIGURATION IDENTIFICATION (ACI) - Current, approved performance oriented specification governing the development of configuration items that are part of a system or higher-level CI, in which each specification;

a) Defines the functional characteristics that are allocated from those of the higher system or CHI.

b) Establishes the tests required to demonstrate achievement of its allocated functional characteristics.

c) Delineates necessary interface requirements with other associated configuration items.

d) Establishes design constraints, if any, such as component/part standardization, use of inventory items, and integrated logistic support requirements.

BASELINE (B/L) - As used in this document, a baseline is a configuration identified at a point in time and thereafter changed only by formal change control procedures.

BASELINE MANAGEMENT - Configuration management by control of progressively redefined baselines. Baseline redefinition continues though out the life cycle of the project.

COMPUTER PROGRAM - A series of instructions or statements in a form acceptable to computer equipment, designed to cause the computer equipment to execute an operation of series of operations.

CONFIGURATION - The functional and physical characteristics as achieved in a product. It includes hardware, software and firmware.

CONFIGURATION AUDIT - The formal examination of the functional characteristics of a configuration item to verify that the item satisfies contractual requirements: or the formal examination of the “as-built” configuration of a configuration item to verify agreement with its configuration documentation.

CONFIGURATION CHANGE REQUEST (CCR) - A CCR is a form that is used to request and initiate a change in the existing configuration. It includes a description of the change, the reason for requesting the change, an analysis of the impact the change will have on the system, and the impact to the project.

CONFIGURATION CONTROL - Configuration control is systematic evaluation, coordination, approval or disapproval and implementation of all changes in the configuration of an item after formal establishment of it configuration identification.

CONFIGURATION CONTROL BOARD (CCB) - The CCB shall perform all system review functions and is the focal point for change management. It is responsible for total assessment of change impact and is the source of directions for change implementation.

CONFIGURATION CONTROL FORMS - The DFRC approved forms required to implement Configuration Control. These are Configuration Change Request (CCR) form, Discrepancy Report (DR) form, Program Change Notice (PCN) form, Work Order (WO) form and the System Test Report (STR) form.

CONFIGURATION ITEM (CI) - An aggregation of hardware and software or any of its discrete portions which satisfies end user function and is designated by the government (or buyer) for configuration management. CI’s may vary widely in complexity, size and type, from aircraft. They can be an aircraft or electronic system, a test meter, or a three-axis attitude system.

CONFIGURATION MANAGEMENT - A discipline applying technical and administrative direction and surveillance to:

a) Identify and document the functional and physical characteristics of a configuration item.

b) Control changes to those characteristics.

c) Record and report change processing and implementation status.

d) Verify compliance with requirements.

CONFIGURATION MANAGEMENT PLAN (CMP) - A plan which identifies the configuration management requirements applicable to a program and how compliance with these requirements will be accomplished.

CONFIGURATION REPORT - A document that describes the “as is” configuration of a flight project at a specific point in time. It is selected jointly by the Project or Project

Chief Engineer and the Document Manager and is used to document changes in the configuration that occurred since the previous configuration report.

CRITICAL DESIGN REVIEW (CDR) - The CDR concludes the specification phase of the software development. In this review evidence shall be provided to prove that the design is compiler and that all the technical requirements have been satisfied.

DISCREPANCY - The events that occur whenever a system fails to operate according to specification or in a manner expected by knowledgeable project personnel. A discrepancy may occur due to a failure in the system, a design flaw, a lack of knowledge or a procedural flaw.

DISCREPANCY REPORT (DR) - The only approved document to report a discrepancy that could unduly impact safety, mission accomplishment, schedule, or cost of any flight project. Squawks that are normally covered by the DFRC workbook entry do not necessarily require a Discrepancy Report.

FLIGHT RELEASE DOCUMENT - A Flight Release Form or Media Release Form is a DFRC form which documents that a specific computer software configuration item has met all DFRC requirements and is suitable for flight. It is prepared by the Software Manager with the concurrence of the Operations Engineering Branch and Quality Assurance representatives.

FLIGHT SOFTWARE - Flight Software is software that directly modifies vehicle operation, whether the software is installed in a system on-board an aircraft, or installed in a ground-based system which modifies aircraft operation.

FLIGHT SUPPORT SOFTWARE - Software that could indirectly impact flight or test operations. Flight Support Software is generally under configuration control, while General Support Software is generally not.

FUNCTIONAL BASELINE - The statement of functional, performance, design, and interface requirements for configuration items in a system.

FUNCTIONAL CONFIGURATION IDENTIFICATION (FCI) - The current approved, or conditionally approved, technical documentation for a CI which prescribes:

a) All necessary functional characteristics.

b) Tests required to demonstrate achievement of specified functional characteristics.

c) Necessary interface characteristics associated with CI's.

d) CI's key functional characteristics and its key lower-level characteristics, if any.

e) Design constraints, such as envelope dimensions, component standardization, use of inventory items, and integrated logistics support policies.

GENERAL SUPPORT SOFTWARE - Support software that is not classified as Flight Software or Flight Support Software and includes software used for data analysis.

HARDWARE - Physical equipment, as opposed to computer programs, procedures, rules, and associated documents.

INDEPENDENT VERIFICATION AND VALIDATION – A process whereby the products of the software development life cycle phases are independently reviewed, verified, and validated by an organization that represents the acquirer of the software and is completely independent of the provider.

OVERLAY - A technique used to change an executable program by overwriting a specific portion of computer memory where the program resides. Overlays are usually incorporated when program constant values need modification with re-locating the program instructions in memory. Re-locating the program instructions in memory is considered a patch.

PATCH - A modification to an object program typically used to change the logic or structure of the executable program. A patch can be incorporated by modification to source code with re-assembly, or manually by use of computer ground support equipment.

PROCEDURE - Specific published steps taken to accomplish an activity. The who, what, when, where, and how in detail necessary to assure safe and proper operation.

PROJECT BASELINE - The "as-build" configuration of a configuration item or a flight project relating to its functional, performance and operating characteristics at a specified point in time.

PROGRAM CHANGE NOTICE (PCN) - A PCN is a document that gives detailed description of the implementation of a software change that had previously been requested and approved with a CCR. It gives the reason for making the change and any remarks that might be deemed helpful. It is not always used on a project.

PROJECT PLAN - The Dryden Project Plan is the basic planning document that describes the overall plan for proceeding with a project. Project plans are required for all DFRC projects prior to implementation. All project plans are approved by the Planning Council and signed by the Center Director.

REQUEST FOR PROPOSAL (RFP) - Requests for proposals are used in negotiated acquisitions to communicate government requirements to prospective contractors ad to solicit proposals from said contractors.

SOFTWARE (S/W) - A collection of associated computer programs and computer data required to enable the computer equipment to perform control or computational functions.

SOFTWARE ASSURANCE (SA) - SA is a planned and systematic process of assuring conformance of software products to established software requirements, approaches, and standards. SA consists of the activities of Quality Assurance, Quality Engineering, Verification & Validation, Nonconformance Reporting and Corrective Action, Safety Assurance, Security Assurance and Configuration Management. SA is distinct from, but supports the activities of, software management and software development.

SOFTWARE CONFIGURATION MANAGEMENT (SCM) - SCM is a discipline applying technical and administrative direction and surveillance to:

a) Identify and document the functional and physical characteristics of software configuration items and baselines.

b) Control changes to those characteristics.

c) Record and report change processing and implementation status.

SOFTWARE DESIGN DESCRIPTION (SDD) - This document shall define the software functional performance, requirements, design constraints, and standards necessary to insure proper development. It describes all of the software to be developed in sufficient detail to permit coding to proceed. Formal change control shall be required sometime after this document is released. An outline for the SDD is given in Attachment D.

SOFTWARE DEVELOPMENT AGENT (SDA) - The Software Development Agent is the NASA organization or contractor responsible for software management, development, and assurance. When the project plans have designated DFRC as the SDA, the SDA shall appoint a Software Manager.

SOFTWARE DEVELOPMENT FILE – The Software Development File is a collection of material pertinent to the development and support of software. Contents typically include design constraints, engineering notes, schedule and status information, lower level test procedures and results, and code listings. Specific contents are defined in the SDP.

SOFTWARE DEVELOPMENT PLAN (SDP) - This document defines the software development process and identifies all activities necessary for implementing a program. Its purpose is to define the scope of the work, to define schedule estimates, to formulate the design baseline, to define the development and testing philosophies and to initiate team selection, work planning and coordination activities. Software development standards and practices are define in this document. An outline for the SDP is shown in Attachment A.

SOFTWARE MANAGER – This is the person immediately responsible for the coordination, direction, documentation, and approval of all software development activities.

SOFTWARE QUALITY ASSURANCE (SQA) - SQA is a planned and systematic pattern of all actions necessary to provide adequate confidence that the item or product conforms to established technical requirements.

SOFTWARE QUALITY ASSURANCE PLAN (SQAP) / SOFTWARE ASSURANCE PLAN (SAP) - This document describes standards and procedures required to assure software quality. An outline for the SQAP/SAP is shown in Attachment B.

SOFTWARE RELEASE - A Software Release is a formally issued new program version of flight code contained in a program media and which has been designated as a product baseline. It is the sole point for change. It is documented in the version Description Document. [ISO 9000-3: Software Product: Complete set of computer programs, procedures, and associated documentation pertaining to the operation of a data processing system.]

SOFTWARE REQUIREMENTS SPECIFICATION (SRS) - This document shall describe overall system characteristics including hardware and software functional requirements. The nature of the task is described, and background information needed for understanding is provided. A functional description of the software capability is presented which includes top-level system requirements to be performed by the software. The SRS shall be prepared prior to the Software Requirements Review. An outline for an SRS is shown in Attachment C.

SOFTWARE SAFETY - Software Safety is the application of the disciplines of System Safety engineering techniques, throughout the software life cycle, to ensure that the software takes positive measures to enhance system safety and that errors that could reduce system safety have been eliminated or controlled to an acceptable level of risk.

SOFTWARE UNIT - The smallest logical entity specified in the design of a computer software component and the actual physical entity in code that implements a testable aspect of the requirements. This is the smallest unit for which documentation may be required.
Note: For the purpose of this document, a ‘software unit’ is interchangeable with (and the same as) a ‘software module’.

SOFTWARE VERIFICATION AND VALIDATION (SVV) - SVV is a process of providing a systematic and objective technical evaluation of the software process and products (see Attachment F).

> VERIFICATION - is the evaluation process by which software is formally determined to accomplish what is specified. It is designed to ensure that the translation of the specification of the previous software life-cycle phase to the current phase is consistent and complete.
>
> VALIDATION - is a testing process that seeks to determine if the system, of which software is a part, performs adequately to accomplish the desired mission objectives. It is designed to assure that the system will perform safely at both nominal and off-nominal conditions.

SUPPORT SOFTWARE - Software that could indirectly impact test or flight operations and software used for the development of flight or test mission software.

TEST PLAN - A document outlining test strategy and coverage without detailed procedures.

TEST PROCEDURES - Step by step instructions for the test to establish inputs, outputs, measurements, and test sequence.

TEST REPORTS - Reports generated during or after tests to revel events accomplished, results, and significant findings.

VERSION DESCRIPTION DOCUMENT (VDD) - A document prepared for and delivered with each new version of a program after release which identifies the media, documentation, and changes applicable to the new program version. The document also identifies known or possible problems with the program. The VDD identification number is changed with each issue. An outline for a VDD is shown in Attachment E.

Attachment K – Acronyms and Abbreviations

ACI	Allocated Configuration Identification
AFSRB	Airworthiness and Flight Safety Review Board
B/L	Baseline
CCB	Configuration Control Board
CCR	Configuration Change Request
CDR	Critical Design Review
CI	Configuration Item
CM	Configuration Management
CMP	Configuration Management Plan
CSA	Code Safety Analysis
DFRC	Dryden Flight Research Center
DR	Discrepancy Report
EMI	Electromagnetic Interference
FCA	Functional Configuration Audit
FQR	Formal Qualification Review
HW	Hardware
IV&V	Independent Verification and Validation
PCA	Physical Configuration Audit
PCN	Program Change Notice
PDR	Preliminary Design Review
QI	Quality Inspection
RFP	Request for Proposal
SADA	Safety Architectural Design Analysis

SA	Software Assurance
SAP	Software Assurance Plan (same as SQAP)
SCCSC	Safety-Critical Computer Software Component
SCM	Software Configuration Management
SDA	Software Development Agent
SDD	Software Design Description
SDDA	Safety Detailed Design Analysis
SDP	Software Development Plan
SDR	System Design Review
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan (same as SAP)
SRR	Software Readiness Review
SRS	Software Requirements Specification
SS	Software Safety
SSRA	Software Safety Requirements Analysis
STR	System Test Report
SVV	Software Verification and Validation
SVVP	Software Verification and Validation Plan
S/W	Software
TRR	Test Readiness Review
VDD	Version Description Document
V&V	Verification and Validation

Attachment L – Lessons Learned

Note: The NASA Lessons Learned Information System (LLIS) and the International Safety Lessons Learned (ISLL) Information System can be found at:
http://llis.gsfc.nasa.gov/

DFRC Lessons Learned Data Base

All employees are encouraged to review and submit lessons learned.

The Dryden Lessons Learned home page is located at:
http://xnet.dfrc.nasa.gov/lessonslearned/

The Dryden Lessons Learned on-line submittal form can be found at:
http://xnet/cgi-bin/t3.cgi/lessonslearned/lldb/users_submit.taf

Definition

A lesson learned is knowledge or understanding gained by experience. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure. Successes are also considered sources of lessons learned. A lesson must be significant in that it has a real or assumed impact on operations; valid in that it is factually and technically correct; and applicable in that it identifies a specific design, process, or decision that reduces or eliminates the potential for failures and mishaps, or reinforces a positive result.

Dryden

- ♦ Guidelines:
  - No Global Variables
  - No Mallocs in Real Time
- ♦ Techniques:
  - Return ASAP from ISR
  - Use Real Time Objects in Architecture
    - Semaphores, Messages, Queses
  - Use Tasks to Turn On and Off Threads Quickly
- ♦ Philosophies:
  - Use In-Line Routines When Possible (Including Macros)
  - Use Common Data Types

Software System Safety Handbook (Joint Software System Safety Committee)

- ♦ The two person rule: At least two people shall be thoroughly familiar with the design, code, testing, and operation of each software module in the system.

- Changeover from hardware to a software implementation must include a review of assumptions, physics and rules.
- Testing should include possible abuse or bypassing of expected procedures.
- Design and implementation of software must be subject to the same safety analysis, review and QA as other parts of the system.
- Hardware interlocks should not be completely eliminated when incorporating software interlocks.
- Programmer qualifications are as important as qualifications for any other member of the engineering team.
- If the software-controlled implementation is not fully understood, the result may be flawed specifications and incomplete tests. Therefore, even though the software and subsystem are thoroughly tested against the specifications, the system design may be in error, and a mishap may occur.
- Changeover from hardware to software requires a review of design assumptions by all relevant specialists acting jointly. This joint review must include all product specifications, interface documentation, and testing.
- The test, verification and review processes must each include end-to-end event review and test.
- Specified equations describing physical world phenomenon must be thoroughly defined, with assumptions as to accuracy, ranges, use, environment, and limitations of the computation.
- When dealing with requirements that interface between disciplines, it must be assumed that each discipline knows little or nothing about the other and, therefore, must include basic assumptions.
- Boundary assumptions should be used to generate test cases as the more subtle failures caused by assumptions are not usually covered by ordinary test cases (division by zero, boundary crossing, singularities, etc.).
- System engineering must define the sequencing of the various states (dismantling, reactivating, shutdown, etc.) of all subsystems with human confirmations and re-initialization of state variables (e.g., site location) at critical points.
- System integration testing should include buffering messages (particularly safety-critical) and demonstration of disconnect and restart of individual subsystems to verify that the system always transitions between states safely.
- Training must describe the safety-related software functions such as the possibility of software overrides to operator commands. This must also be included in operating procedures available to all users of the system.

LLIS/ISLL

- Lack of fidelity between ground test system configuration and the flight system configuration can result in costly and damaging test failures to flight hardware during ground testing.
- End-to-end system checks, including software driven interfaces, should be performed to verify proper connection, wiring, signal level and function. If necessary, as in the case of pyrotechnic circuits, simulators should be used.

- ♦ Inappropriate fault protection actions can be as hazardous as the failure the system was designed to protect against.
- ♦ Fault protection software should be tested on the aircraft before flight.
- ♦ A continuous software looping operation, or "deadly embrace", can occur undetected in some flight software applications.
- ♦ Software packages that have worked in the past could prove to be faulty or inappropriate for the current aircraft state.
- ♦ All commands, whether stepping or explicit, have within them a potential for error. However, since explicit commands do not depend on the success of previous commands to be successful, they tend to be "fault-tolerant," in the sense that they do not perpetuate errors in the command sequence.
- ♦ Provide flight software with an independent watchdog timer to terminate operations that exceed the specified maximum time duration.
- ♦ Establish a software implementation plan early that outlines the basic strategy, including reviews, standards, processes, schedule, and deliverables.
- ♦ Make early, conscious, engineering decisions on the applicability of Computer-aided Software Engineering (CASE) tools. Beyond basic software development planning, their use can consume resources and prove counterproductive.
- ♦ Failure to perform IV&V for software projects could result in software system weaknesses, performance of unintentional functions, and failure of the system and the mission. Anything less than a methodical, systematic rigorous treatment of IV&V could cause loss of mission, life, and valuable resources.

<u>Marshall Space Flight Center (MSFC)</u>

Note: Some of the following items may not be applicable to flight research software development programs, but are included for your consideration.

- ♦ The failure of safety critical software functions shall be detected, isolated, and recovered such that catastrophic and critical hazardous events are prevented from occurring.

- ♦ Software shall perform automatic Failure Detection, Isolation, and Recovery (FDIR) for identified safety critical functions with a time to criticality under 24 hours.

- ♦ Automatic recovery actions taken shall be reported to the crew, round, or controlling executive. There shall be no necessary response from crew or ground operators to proceed with the recovery action.

- ♦ The FDIR switchover software shall be resident on an available, non-failed control platform that is different from the one with the function being monitored.

- ♦ Override commands shall require multiple operator actions.

- ♦ Software shall process the necessary commands within the time to criticality of a hazardous event.

- Hazardous commands shall only be issued by the controlling application, or by the crew, ground, or controlling executive.
- Software that executes hazardous commands shall notify the initiating crew, ground operator, or controlling executive upon execution or provide the reason for failure to execute a hazardous command.
- Prerequisite conditions (e.g., correct mode, correct configuration, component availability, proper sequence, and parameters in range) for the safe execution of an identified hazardous command shall be met before execution.
- In the event that prerequisite conditions have not been met, the software shall reject the command and alert the crew, ground operators, or the controlling executive.
- Software shall make available status of all software controllable inhibits to the crew, ground operators, or the controlling executive.
- Software shall accept and process crew, ground operator, or controlling executive commands to activate/deactivate software controllable inhibits.
- Software shall provide an independent and unique command to control each software controllable inhibit.
- Software shall incorporate the capability to identify and status each software inhibit associated with hazardous commands.
- Software shall make available current status on software inhibits associated with hazardous commands to the crew, ground operators, or controlling executive.
- All software inhibits associated with a hazardous command shall have a unique identifier.
- Each software inhibit command associated with a hazardous command shall be consistently identified using the rules and legal values.
- If an automated sequence is already running when a software inhibit associated with a hazardous command is activated, the sequence shall complete before the software inhibit is executed.
- Software shall have the ability to resume control of an inhibited operation after deactivation of a software inhibit associated with a hazardous command.
- The state of software inhibits shall remain unchanged after the execution of an override.
- Software shall provide error handling to support safety critical functions.
- Software shall provide caution and warning status to the crew, ground operators, or the controlling executive.
- Software shall provide for crew/ground forced execution of any automatic safing, isolation, or switchover functions.

- Software shall provide for crew/ground forced termination of any automatic safing, isolation, or switchover functions.
- Software shall provide procession for crew/ground commands in return to the previous mode or configuration of any automatic safing, isolation, or switchover function.
- Software shall provide for crew/ground forced override of any automatic safing, isolation, or switchover functions.
- Software shall provide fault containment mechanisms to prevent error propagation across replaceable unit interfaces.
- Hazardous payloads shall provide failure status and data to core software systems. Core software systems shall process hazardous payload status and data to provide status monitoring and failure annunciation.
- Software (including firmware) Power-On Self Test (POST) utilized within any replaceable unit or component shall be confined to that single system process controlled by the replaceable unit or component.
- Software (including firmware) POST utilized within any replaceable unit or component shall terminate in a safe state.
- Software shall initialize, start, and restart replaceable units to a safe state.
- For systems solely using software for hazard risk mitigation, software shall require two independent command messages for a commanded system action that could result in a critical or catastrophic hazard.
- Software shall require two independent operator actions to initiate or terminate a system function that could result in a critical hazard.
- Software shall require three independent operator actions to initiate or terminate a system function that could result in a catastrophic hazard.
- Operational software functions shall allow only authorized access.
- Software shall provide proper sequencing (including timing) of safety critical commands.
- Software termination shall result in a safe system state.
- In the event of hardware failure, software faults that lead to system failures, or when the software detects a configuration inconsistent with the current mode of operation, the software shall have the capability to place the system into a safe state.
- When the software is notified of or detects hardware failures, software faults that lead to system failures, or a configuration inconsistent with the current mode of operation, the software shall notify the crew, ground operators, or the controlling executive.

- Hazardous processes and safing processes with a time to criticality such that timely human intervention may not be available, shall be automated (i.e., not require crew intervention to begin or complete).
- The software shall notify crew, ground, or the controlling executive during or immediately after execution of an automated hazardous or safing process.
- Unused or undocumented codes shall be incapable of producing a critical or catastrophic hazard.
- All safety critical elements (requirements, design elements, code modules, and interfaces) shall be identified as "safety critical."
- An application software set shall ensure proper configuration of inhibits, interlocks, and safing logic, and exception limits at initialization.

Attachment M – Software Assurance Self-Evaluation Report

Each Dryden software development program should complete the Software Assurance Self-Evaluation Report as early as possible in the software development process.

For each checklist question, indicate a *yes* or *no* answer in the appropriate column. Provide any additional comments or explanations in the "Remarks" column. If a question does not apply to your project, so indicate by writing "N/A" in the "Remarks" column.

Handwritten responses are acceptable. If you prefer, the Safety and Mission Assurance Office will provide you with an MS Word version of the checklist.

Return the completed checklist to the DFRC Safety and Mission Assurance Office. Retain a copy of your responses for your project files.

NOTE

The inclusion of applicable document titles and document numbers (as requested in the following pages) will save you a great deal of time when more extensive software audits are performed in the future.

Please provide the following general information to identify the project and key project personnel.

Project Name ______________________________

Project Manager ______________________________

Software Manager / Software Development Agent (SDA) ______________________________

Configuration Mgmt Rep ______________________________

S/W Program Start Date ______________________________

S/W Program Due Date ______________________________

System Design Review (SDR) Date ______________________________

PDR Date ______________________________

CDR Date ______________________________

Formal Qualification Review Date ______________________________

Tech Brief / AFSRB / FRR Date ______________________________

Completed by ______________________________

Date ______________________________

Dryden Flight Research Center

Element A: Software Organization and Management

This element addresses the organization and management structure of your project. The term 'manager' is used to refer to the individual in charge of a function. For example, if there is no designated software manager, the lead software engineer is considered the 'software manager' for audit purposes. Please include the name of the process owner in the "Remarks" column.

Activity	Yes	No	Remarks
Does your project have an organizational chart that clearly identifies the software development and Software Assurance functions?			
Do you normally assign a designated software manager to control all software, developed or purchased, for the project (including supplier software)?			
Do you have, in use, a Software Development Plan (SDP)?			
Do you have, in use, a mechanism for creating and maintaining detailed software schedules?			
Do you have, in use, a mechanism for identifying and reducing technical and schedule risks?			
Do you regularly monitor your suppliers' software activities?			
Do your software management techniques include software size and cost estimating?			
Does your project generate and use management indicators?			
Has an independent software assessment been performed on your project?			

Remarks:

Element B: Software Quality Assurance

This element addresses the Quality Assurance Plan and the responsible individuals for your project. Please include the name of the responsible individual in the "Remarks" column. Also note the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Do you have, in use, a documented Software Quality Assurance Plan (SQAP)?			
Does the SQAP conform to DOD, NASA or other standards?			
Has an individual been formally assigned the responsibility for implementing the SQAP for this project?			
Do you regularly perform software product evaluations?			
Do you regularly perform software process evaluations?			
Does your project use software quality measures?			
Do SQA personnel regularly audit software supplier activities?			

Remarks:

Element C: Software Design and Development

This element addresses the software design environment for your project. Please include the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Do you have a documented set of software development standards and procedures?			
Do your software development standards and procedures require formal documentation, records and configuration control?			
Do your software development standards and procedures conform to DOD, NASA or other standards?			
What software language(s) is currently used for software design and coding?			
Do you have, in use, a system/software engineering environment using automated tools?			
Are you generating technical performance metrics?			
Are formal walkthroughs (or equivalent) used to verify design and code?			

Remarks:

Element D: Software Test, Verification and Validation

This element addresses the software test environment and the responsible individuals for your project. Please include the name of the responsible individual in the "Remarks" column. Also note the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Do your software development standards and procedures include software testing?			
Do you have, in use, a documented software testing methodology?			
Does your software testing methodology conform to DOD, NASA or another standard?			
Do you regularly produce formal test documentation and reports?			
Do you have a mechanism for performing formal qualification testing of the software?			
Does your project have an independent test organization to perform acceptance and qualification testing?			
Are automated tools used to help generate test cases and perform tests?			

Remarks:

Element E: Software Qualification and Certification

NOTE

If your software project has no requirement for FAA certification, indicate this by writing "N/A" in the "Remarks" column and skip to Element F.

This element is generally applicable only to those projects developing software that may see commercial air transport application. The air transport industry develops and certifies software according to the current version of FAA standard RTCA/DO-178. The FAA uses software Designated Engineering Representatives (DER) to verify the software was developed according to acceptable standards.

While the Dryden Software Assurance procedure (DCP-S-007) does not mandate use of DO-178, it is strongly recommended for commercial aircraft software development.

Activity	Yes	No	Remarks
Do you have a plan for obtaining product certification from a certification agency (e.g., FAA)?			
Do you have a plan for addressing the software aspects of certification?			
Do your software plans include developing and maintaining a software accomplishments summary or software configuration index?			
Does your project have access to a Designated Engineering Representative (DER) for software?			
Has an individual been formally assigned to address certification issues and interface with the certification agency?			

Remarks:

Element F: Software Configuration and Traceability

This element addresses the software configuration issues and the responsible individuals for your project. Please include the name of the responsible individual in the "Remarks" column. Also note the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Do you have, in use, a documented Configuration Management Plan (CMP)?			
Do your software development standards and procedures include software configuration control?			
Do you have, in use, documented procedures for managing and controlling software documentation and associated revisions?			
Do you have, in use, documented procedures for managing and controlling software and software changes?			
Has an individual been formally assigned the responsibility for software configuration management for this project?			
Are you using automated configuration control tools?			
Do you maintain software development files throughout the software life cycle?			
During the software life cycle, do you use a secure software development library as a repository for all development and product baseline software and documentation?			
Do you have, in use, a mechanism for accomplishing requirements traceability to all software elements?			
Are you using an automated tool to help accomplish requirements traceability?			

Remarks:

Element G: Software Problem Resolution and Correcti ve Action

This element addresses the corrective action and change request process for your project. Please include the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Do your software development standards and procedures include software problem resolution and corrective action?			
Do you have, in use, documented procedures for managing and controlling corrective actions?			
Are you formally documenting (and tracking to closure) corrective actions to baseline software and documentation?			
Are you using an automated tool to help document and track problems and corrective actions?			
Are problems and corrective actions associated with prime contractors and sub-tier suppliers documented and tracked to closure?			
Are you performing problem trend analyses?			

Remarks:

Element H: Software Supplier Requirements Flow Down Control

NOTE

If your software project has no software supplier(s), indicate this by writing "N/A" in the "Remarks" column and skip to Element I.

This element addresses the flow down of requirements to software suppliers and the responsible individuals for your project. Please include the name of the responsible individual in the "Remarks" column. Also note the document numbers of any guidelines or standards applicable to your project.

Activity	**Yes**	**No**	**Remarks**
Do your software development standards and procedures include monitoring supplier software activities?			
Has an individual been formally assigned responsibility to oversee suppliers?			
Do you have, in use, a documented procedure for approving software suppliers?			
Are suppliers required to conform to DOD, NASA or other standards?			
Do you have, in use, a mechanism for controlling changes to supplier software?			
Do you have, in use, a mechanism for resolving and tracking problems with supplier software?			
Do you regularly perform on-site evaluations of supplier's software activity?			

Remarks:

Element I: Software System Safety

This element addresses Software Safety and the responsible individuals for your project. Please include the name of the responsible individual in the "Remarks" column. Also note the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Does your project have a System Safety program?			
Does your project have a Software Safety program?			
If yes, does it conform to DOD, NASA or other standards?			
Does your System Safety program include performing a Software Hazard Analysis?			
Does your project have a function dedicated to addressing and tracking Software and System Safety issues as discrete hazard items in the hazard analyses?			
Has an individual been formally assigned responsibility for Software Safety activities for this project?			
Do you have documented criteria for evaluating Software Safety characteristics?			
Does your Software Requirements Analysis include: 1) accurate translation of safety specification requirements; and 2) identification of high criticality software and safety related requirements?	1. 2.	1. 2.	
Is software testing conducted under abnormal environmental and input conditions (as well as normal conditions) to ensure it performs properly and safely?			
Are you using automated tools to help prepare safety analyses?			

Remarks:

Element J: Software Continuous Quality Improvement

This element addresses continuous quality improvement initiatives and the responsible individuals for your project. Please include the name of the responsible individual in the "Remarks" column. Also note the document numbers of any guidelines or standards applicable to your project.

Activity	Yes	No	Remarks
Do you have a mechanism for improving software development and Software Assurance processes?			
Does your project have, in use, a Total Quality Management program that addresses software?			
What Continuous Quality Improvement tools and mechanisms are you using?			
Are you using process measures to help improve your software development and Software Assurance processes?			
Does your project participate in a software engineering process group or equivalent?			

Remarks: