DEPARTMENT OF THE AIR FORCE
Headquarters Air Force Life Cycle Management Center (AFMC)
Engineering Directorate
Wright-Patterson AFB OH 45433-7101

AC-17-01
23 MAR 2017

AIRWORTHINESS CIRCULAR
Verification Expectations for Select Section 15 Criteria

PURPOSE:

This Airworthiness Circular (AC) identifies expectations for showing compliance to a set of criteria found in MIL-HDBK-516C, Section 15, *Computer Systems and Software (CS&S)*, which is used in the United States Air Force (USAF) airworthiness certification process.

SCOPE:

This AC applies to all USAF air systems, including those operated by the Air National Guard and USAF Reserve.

ATTACHMENTS:

(1) Detailed Guidance Regarding Safety Critical Function (SCF) Identification
(2) Detailed Guidance Regarding SCF Thread Analysis (SCFTA)
(3) Detailed Guidance Regarding System and Software Integration Methodology
(4) Detailed Guidance Regarding Failure Modes and Effect Testing (FMET)
(5) Detailed Guidance Regarding Safety Interlock Design Mechanization
(6) Detailed Guidance Regarding CS&S Development Process and Product Attributes
(7) Abbreviations, Acronyms, and Glossary of Terms

CANCELATIONS:

Not applicable. This is the first issuance of this AC.

REFERENCED DOCUMENTS:

[1] MIL-HDBK-516C, *Airworthiness Certification Criteria*, 12 December 2014
[2] MIL-STD-882D/E, *System Safety Program*, 10 February 2000/23 April 2012
[3] AWB-1011A, *Airworthiness Expert Endorsement*, 4 September 2014
[4] IEEE 12207, *Systems and Software Engineering – Software Life Cycle Processes*, 2008
[5] MIL-STD-498, *Software Development and Documentation*, 5 December 1994
[6] DOD-STD-2167A, *Defense System Software Development*, 29 February 1988
[7] DO-178B/C, *Software Considerations in Airborne Systems and Equipment Certification*, 1 December 1992 / 13 December 2011
[8] JSSG-2000A, *Joint Service Specification Guide: Air System*, 8 October 2002
[9] JSSG 2001A, *Joint Service Specification Guide: Air Vehi*cle, 22 October 2002
[10] MIL-HDBK-61A, *Configuration Management Guidance*, 7 February 2001
[11] SAE EIA-649_1, *Configuration Management Requirements for Defense Contracts*, Nov 2014
[12] *DOD Unmanned Aircraft System Airspace Integration Plan*, Version 2.0, March 2011

BACKGROUND:

With the advent of integrated computer system architectures, air system functionality is often dependent on sensor information, data buses, subsystem processing, backplanes, output signals, data/system latency requirements, software partitioning, etc. This has led to an increased reliance on executing Safety Critical Functions (SCFs) with integrated computer system architectures. To provide the requisite safety assurance, the USAF airworthiness certification process has recognized that it is necessary to adhere to a rigorous standard of safety verification for these systems, referred to as System Processing Architectures (SPAs). The USAF airworthiness certification process utilizes MIL-HDBK-516 Section 15, *Computer Systems and Software*, to establish the airworthiness verification criteria for SPAs.

Over the past few years, the majority of airworthiness certification activities have involved development of, or modifications to, computer systems and software. During the review of many of these efforts, certain criteria within Section 15 have frequently been found to be non-compliant. As a result, this AC has been written to clarify expectations associated with airworthiness criteria within Section 15 experiencing repeated non-compliance. The guidance in this AC elaborates on particular airworthiness certification requirements that focus on design contributions that the hardware and software must provide to the system architecture in support of Safety/Flight Critical functionality, as well as key verification activities that are needed to evaluate the safety risk associated with the system design. An overview of the verification expectations focused upon in this AC involve:

a) Identification of Safety Critical Functions (SCFs),
b) Identification of all hardware/software supporting SCFs,
c) Redundancy as required to support the criticality,
d) Fault tolerance and redundancy management to handle faults,
e) Verification and validation (V&V) activity expected from lower level testing through system integration (including Failure Modes Effects Analysis/Testing (FMEA/T),
f) Evaluation of interlock design mechanizations that provide de-confliction of functional modes and are used to ensure safe operation,
g) Strict adherence to a robust development process and sound V&V approach,
h) Full qualification of software requirements for every flight release of software supporting an SCF.

Meeting the above requirements is vital for demonstrating compliance to certain Section 15 criteria.

DISCUSSION AND RECOMMENDATIONS:

The following focus areas are addressed to ensure a better understanding of the design, development, integration, and V&V expectations for showing compliance to Section 15. Each focus area contains a discussion paragraph and a recommendation paragraph.

a) Safety Critical Function (SCF) Identification
b) SCF Thread Analysis (SCFTA)
c) Integration Methodology: System, Software, & Levels of Testing
d) Failure Mode and Effects Testing (FMET)
e) Safety Interlock Design
f) SPA and Software Development Processes
g) Full Qualification of Software

SCF Identification

Discussion: The term Safety Critical Function is defined in both MIL-STD-882 and MIL-HDBK-516C as: a function whose failure to operate or incorrect operation will directly result in a mishap of either Catastrophic or Critical severity. Per MIL-STD-882, SCFs are to be identified as part of the initial activity associated with the system safety process. Once identified, the SCFs are used in the Functional Hazard Analysis (FHA), which lays the foundation for identifying hazards within the system. The identification of SCFs is critical to understanding the focus area of airworthiness-oriented functionality. In Section 15 one criterion, 15.1.1, requires that SCFs have been identified for the air system. All criteria in Section 15 indirectly rely on SCF identification since the criteria are to be applied to equipment supporting SCFs.

Recommendation: To demonstrate compliance with MIL-HDBK-516 Section 15 criteria 15.1.1 involving the identification of SCFs, the System Safety process (supported by functional engineering teams) should identify the system's applicable SCFs, which should then be used as the foundation for performing SCFTAs.

NOTE: Other MIL-HDBK-516 sections (4, 6, 7, 8, 9, 10, 11, 12, 13, 14, and 17) are also dependent on this information.

See Attachment 1 (page 9) for *Detailed Guidance Regarding SCF Identification*.

SCFTA

Discussion: An SCF thread is defined in MIL-HDBK-516C as: the combination of elements/components within a system and the required interfacing and interaction of those elements/components whose overall contribution is necessary for the operation of a given SCF. An SCFTA's purpose is to:

1) Identify all the elements, hardware and software components, and interfaces that are necessary for the safe execution of all identified SCFs,
2) Ensure the identified elements and components are developed at Computer System Integrity Levels (CSILs) appropriate for SCF applications, and that safety critical interfaces are identified as such, and
3) Verify that end-to-end V&V coverage is achieved by the tests used to verify the SCF functionality (includes: component level test and review; subsystem level test; through system integration test).

A key purpose of the airworthiness process is to ensure the design is safe to operate within its intended envelope of operation. It stands to reason then that verifying the end-to-end functionality of the portions of the system that contribute to airworthiness safety risk (i.e., SCFs) is essential to establishing confidence in the airworthiness of the design. The SCFTA is considered to be a foundational tool for providing evidence that the end-to-end SCF functionality has been verified.

Recommendation: To demonstrate compliance with the various MIL-HDBK-516 Section 15 criteria involving SCFTAs and demonstrating that the end-to-end V&V of the air system's SCFs have been adequately achieved, the SCFTA activity should be performed as an integral process to the development and V&V activities. An SCFTA should be performed on all SCFs that are supported by computer systems and software. An SCFTA is considered to be satisfactorily completed when all the SCF threads have been fully identified (i.e., all supporting elements, components, and interfaces identified with associated CSIL) and complete test coverage of all SCF threads is verified and documented. The SCFTA should be reviewed and updated as necessary for every flight release of software supporting an SCF and every modification impacting an SCF.

See Attachment 2 (page 14) for *Detailed Guidance Regarding SCFTA.*

System and Software Integration Methodology

Discussion: Within Section 15, paragraphs 15.2.3 and 15.6.1 evaluate the adequacy of the integration methodology associated with the development of computer systems supporting SCFs. Section 15 seeks to verify the system integration methodology, the software integration methodology, and assure that complete test coverage is obtained through all levels of testing.

The system integration methodology is the systematic process that is employed to bring the subsystem elements of a system together as a functional system. The system integration level of V&V testing stresses the SPA architectural design. The system integration activity includes the informal demonstration and test of subsystems integrated into a nearly complete functional system as well as the activity to verify the system level requirements.

The software integration methodology is the systematic process that is employed to bring the constituent components of a Computer Software Configuration Item (CSCI) together ***AND***

integrate that CSCI into the complete system. The software integration level of V&V testing stresses the software architectural design. The software integration activity includes the informal testing related to the integration of software components that make up the CSCI as well as the activity to verify the software level requirements.

Recommendation: To demonstrate compliance with MIL-HDBK-516 Section 15 criteria involving the system and software integration methodologies, the integration and test plans show that the following is achieved:

a) Complete V&V coverage of requirements, functions, and failure conditions (to include boundary and out-of-bounds test conditions)
b) End-to-End functional test coverage of SCF threads over all levels of testing
c) Test methodologies include proper levels of testing and that the testing focus is appropriate at each level
d) All essential functionality is included and fully verified for intended flight operations
e) Unmodified SCFs have been appropriately regression tested and are proven to be unaffected by the modification effort

See Attachment 3 (page 21) for *Detailed Guidance Regarding System and Software Integration Methodologies*.

FMET

Discussion: Understanding the system's susceptibility to errors and faults is essential in determining that a system is safe. Analyzing the system through Failure Mode and Effects Analysis (FMEA), Failure Modes, Effects and Criticality Analysis (FMECA), and other analyses identify areas in the system where failures can result in unsafe operation. FMET is needed to perform specific testing based upon fault risk areas identified in the FMEA/FMECA, combination failure, and various safety hazard analyses. FMET specifically injects failure conditions into the system at each level of the design to verify that the implemented design mitigates the identified failure conditions safely while revealing to what degree the functionality is impacted by a given failure. The completeness of FMET test coverage is assessed against: the set of failure conditions tested against those that are identified in the design; and the thoroughness in evaluating the system response, system performance, and crew responses under tested failure conditions.

Recommendation: To demonstrate compliance with the various MIL-HDBK-516 Section 15 criteria involving FMET, a comprehensive suite of failure mode tests should be developed and executed at each integration level of the design. The FMET test cases should be based upon analysis results that identify fault conditions (including both loss and degraded modes) that could potentially occur in the system and impact SCF operation. FMET test cases are not to be constrained (eliminated from the FMET test set) simply due to an unlikely probability of occurrence (except as noted in Attachment 4, page 32), or for factors such as design mitigations that are in place to manage

the risk of the failure (Attachment 4, page 32 provides guidance on determining the scope of FMET coverage). Failure modes that cannot be fully tested in the available laboratory environments should be documented and communicated to USAF program offices' Section 15 Subject Matter Experts (SMEs)/points of contact.

See Attachment 4 (page 28) for *Detailed Guidance Regarding FMET*.

Safety Interlock Design Mechanization

Discussion: Interlocks are defined in MIL-HDBK-516C as system design mechanization to enable or disable systems, functions, subsystems, or modes at given times and conditions. A safety interlock is defined in MIL-HDBK-516C as an interlock that is necessary for the operation of one or more SCFs. Understanding the use and the design of safety interlocks is critical to ensuring the safe operation of an SCF. For Airworthiness purposes, safety interlocks provide control over the functional operation of an SCF to ensure safe operation is maintained with proper mode engagement (or enabling of functionality) and disengagement (or disabling of functionality). Safety interlocks are therefore designed with mode controlling logic and this logic is often complex in nature. Analyzing and testing all safety interlocks' logical designs is critical to establishing airworthiness assurance.

Recommendation: To demonstrate compliance with MIL-HDBK-516 Section 15 criteria 15.2.6 and 15.5.4 involving safety interlocks, all safety interlocks associated with an SCF thread should be identified. Once identified, the design of every safety interlock should be analyzed for nominal and off-nominal operation, considering all coupling influences of the control parameters utilized within the interlock. In turn, the testing performed on safety interlocks needs to be designed to dynamically exercise the nominal and off-nominal conditions identified in the safety interlock analysis.

See Attachment 5 (page 33) for *Detailed Guidance Regarding Safety Interlock Design Mechanization*.

SPA and Software Development Processes

Discussion: Numerous criteria in Section 15 evaluate the suitability of the development and V&V processes used for producing a system's SPA and software. A tenet in the design of Section 15 criteria is that poor processes equate to low safety integrity. If the software is supporting an SCF (and per definition the software is then a Safety Supporting Software Element (SSSE) as defined in MIL-HDBK-516C), then ensuring the integrity of the software and the SPA it runs on is essential to establishing the safety assurance provided by the software products. SPA and software processes are captured in a vast array of documents that focus on specific aspects of the SPA/software development and V&V activity. These documents will range from system level documents down to software specific documents. Consideration should be given as to what program process

documents will be needed to show compliance to SPA and software process criteria, keeping in mind that Section 15 views the verification activity for software starting at the software unit level and continuing through to system level verification. The established processes should also ensure that product attributes expected for items supporting SCFs are instilled within SPA equipment.

Recommendation: To demonstrate compliance with the various MIL-HDBK-516 Section 15 criteria involving SPA and software development processes, programs should develop SPA and software process documentation that addresses all Section 15 Computer System & Software (CS&S) process verification criteria. The CS&S process should ensure a CSIL is assigned that establishes development and V&V processes that are designed to instill an acceptable level of safety integrity into the SPA and software products. Programs also should ensure contract clauses exist with suppliers and vendors requiring them to provide CS&S process documentation supporting Section 15 airworthiness verification criteria when those suppliers and vendors are developing SPA Safety Supporting Hardware Elements (SSHEs) and SSSEs for the air system. The documented software processes should be supported by lower level work instructions that describe in step-by-step detail how tasks supporting the processes are to be executed.

See Attachment 6 (page 35) for *Detailed Guidance Regarding CS&S Development Process and Product Attributes*.

Full Qualification of Software

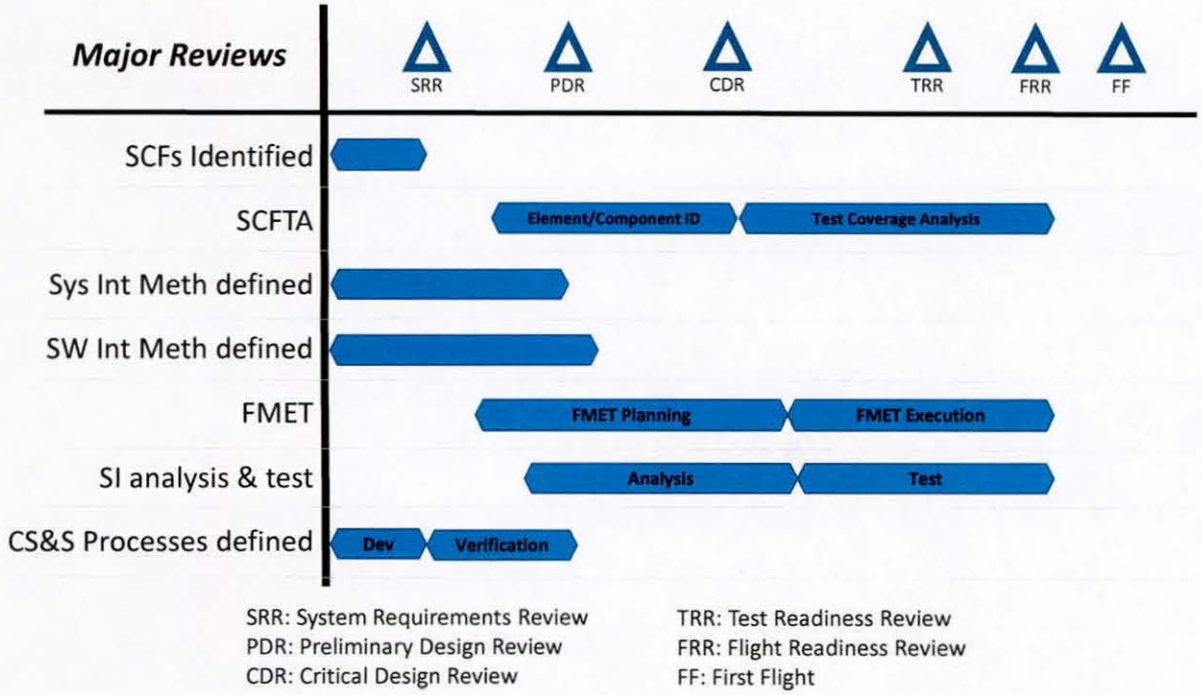
Discussion: Full qualification of software is achieved when 100 percent of the software-level requirements are tested before the software is released for flight. Software-level requirements are the requirements that are written against the CSCIs. For airworthiness purposes, a CSCI typically identifies a collection of software routines that are functionally related. When software routines are functionally related there is some degree of functional coupling that is expected to exist. The airworthiness criteria presume that this functional coupling exists and requires the verification of all software-level requirements for a new flight release of software to ensure:

a) That all changed software meets requirements
b) That all unchanged software continues to meet requirements

Recommendations: To demonstrate compliance with MIL-HDBK-516 Section 15 criteria involving full qualification of software, for each SSSE CSCI (i.e., a software entity with a given set of requirements) perform a systematic verification of every software requirement on the target processing hardware configuration installed on the air system before releasing for flight operation. The verification activity should ensure failed test cases are properly dispositioned to ensure all safety concerns are properly mitigated. If for some reason the program has failed to designate CSCIs against the software supporting SCFs, then each software load image that supports an SCF will be considered the equivalent to a CSCI.

Ideal phasing for performing the activities identified above are shown in the chart on the next page.

Ideal Phasing of Section 15 Activity Identified In This Circular



Section 15 AC Activity Ideal Phasing Chart

POINTS OF CONTACT:

The Office of Primary Responsibility (OPR) for this AC is the Integrated Avionics, Computer & Software Systems Branch (AFLCMC/EZAS). Comments, suggestions, or questions on this AC should be directed to: Mr. Dana Springer DSN 312-785-9407, commercial (937-255-9407), or email at dana.springer@us.af.mil; or Mr. Christopher Jackson DSN 312-785-2711, commercial 937-255-2711, or email at christopher.jackson.1@us.af.mil. General airworthiness questions should be directed to the USAF Airworthiness Office (USAF.Airworthiness.Office@us.af.mil).

JORGE F. GONZALEZ, SES
Director, Engineering and Technical
Management/Services
USAF Technical Airworthiness Authority

Attachment 1
DETAILED GUIDANCE REGARDING SCF IDENTIFICATION

Safety Critical Functions (SCFs) are defined at the weapon system or air system level so they are necessarily high level functions. SCFs should be identified by the program's System Safety activity with support of engineers from relevant technical discipline areas. From an airworthiness perspective, identification of SCFs is essential to the process of verifying all functionality that contributes to airworthiness risk. The specific set of SCFs for a given system will be unique to each platform. All criteria in Section 15 indirectly rely on SCF identification since the criteria are only to be applied to equipment supporting SCFs, however there is one criterion (15.1.1) that verifies that SCFs have been identified for the system. In addition to 15.1.1, there are 32 criteria (including their associated standards) that directly reference or per definition (i.e., make reference to SSEs or flight critical functionality) rely on SCFs being identified in order to properly perform the verification. Those criteria are:

15.1.3	15.1.11	15.3.1	15.5.4
15.1.4	15.2.1	15.3.2	15.5.6
15.1.5	15.2.2	15.3.3	15.5.7
15.1.6	15.2.3	15.4.1	15.5.8
15.1.7	15.2.5	15.4.2	15.5.10
15.1.8	15.2.6	15.4.3	15.6.1
15.1.9	15.2.7	15.5.2	15.6.2
15.1.10	15.2.8	15.5.3	15.6.3

In identifying the air system's SCFs it is recommended to generate a program controlled list of these functions called an SCF List. The SCF List should be generated by the system safety process activity. System safety takes the functional definition for the air system that is produced by system engineers and analyzes the functions for meeting the MIL-STD-882 definition of SCFs. This system safety analysis should be supported by engineers from the technical disciplines associated with each function analyzed and can be performed in conjunction with the Functional Hazard Analysis required by MIL-STD-882. For application to Section 15 airworthiness verification activities, the SCF List should be reviewed to identify which SCFs are supported by CS&S equipment.

Identifying an SCF is based upon the definition of an SCF, which is defined as: A function whose failure to operate or incorrect operation will directly result in a mishap of either Catastrophic or Critical severity. The development community has differing interpretations for the phrase "will directly result in". In an attempt to eliminate the confusion over differing interpretations we

recommend interpreting the phrase "will directly result in" to include the following: When the function is lost or corrupted it creates the conditions, without consideration to probability, to produce a Catastrophic or Critical mishap; when the function is expected to operate under a certain set of preconditions (e.g., an emergency condition, a specific failure condition), then when evaluating the function for applicability as an SCF, the preconditions should be assumed to already exist; when the loss or degradation of the function could lead the crew to make decisions that would cause a Catastrophic or Critical mishap; and when a function is needed to mitigate circumstances that could cause a Catastrophic or Critical mishap, then the function should be considered an SCF. SCF implementations require the capability to detect, notify (when crew notification is critical to safe operation), manage redundancy, and accommodate failures, which are incorporated into the system architecture to ensure the safe operation of an SCF, and therefore should be considered part of the SCF.

When performing the analysis to identify an SCF, it is common that a developer will assess the function's criticality based upon the equipment design supporting the function. This approach is not a pure functional assessment and should not be done. A function should be evaluated for applicability as an SCF purely on the basis of being able to potentially cause a catastrophic or critical consequence (in MIL-STD-882 parlance, the hazard severity) when the function is lost or degraded. If the answer is yes, it is an SCF per definition. If it is an SCF then certain design considerations and process actions need to be employed because of the safety related nature of the function. If a developer is examining the design employed to identify if a given function is an SCF or not, then that developer is not properly assessing the criticality of the function. Likewise, if a developer is considering the probability of failure (and thus considering the design of the equipment supporting the function) to make an SCF determination, then the developer is not properly assessing the criticality of the function.

Below are a collection of SCFs that have been identified on various USAF platforms. This list can be used to help identify functions that may meet the definition of an SCF for an air system under development or modification. To provide some organization to the list, the SCFs have been grouped into 5 categories titled: Flight Critical, Operation Critical, Emergency Critical, Indication Critical, and Avoidance Critical. The only purpose for the 5 categories is to help convey the variety of functions that can be identified as SCFs. Below are descriptions for each of the categories.

1. Flight Critical functions are functions used to achieve and control flight (loss or degradation could directly lead to loss of aircraft).
2. Operation Critical are SCFs that are used for supporting a non-Flight Critical function that has inherent safety functionality associated with its operation (loss/degradation could directly lead to a consequence of Catastrophic or Critical hazard severity).

3. Indication Critical are SCFs needed to provide indications to pilot/crew necessary for maintaining safe operation.
4. Emergency Critical are SCFs that exist purely for the purpose of mitigating risk associated with emergency conditions.
5. Avoidance Critical are SCFs needed purely to mitigate a potential safety risk.

The specific functions listed may not always be an SCF on every type of air system configuration and may be referred to with different nomenclature than what is listed in this AC. Functions in the list below that are followed by the label "[UAS]" are functions that are often considered SCFs for unmanned aircraft systems, but may not be for manned aircraft systems.

FLIGHT CRITICAL

- Flight Control
  - Level III Flying Qualities
  - Stability in Pitch, Roll, Yaw
  - Air Data Sensing
  - Pitot Heating
  - Autopilot
- Lift System Control
- Supply and Control of Critical Utilities
  - Electrical
  - Hydraulic
  - Thermal Management
  - Fuel/Center of Gravity (CG) Control
- Automated Landing
- Automated Takeoff
- Automated Terrain Following
- Structural Integrity
  - Withstand Flight Loads
  - Withstand Ground Loads
  - Freedom from Flutter
  - Structural Damage Tolerance
- Thrust/Propulsion Control
- In-flight Restart/Reset
- Landing Gear Extension/Control
- Ground Steering

OPERATION CRITICAL

- Ground Deceleration
  - Differential Braking
  - Arresting Hook
  - Thrust Reversing
- Navigational Control/Auto-Pilot Operation
- Fuel Tank Pressurization & Temperature Control
- Fuel Feed to Propulsion System
- Mid-Air Refueling/Boom Control
- Anti-Icing
  - Control surfaces
  - Engine/Inlet
- Mission Planning [UAS]
- Communication Link Control [UAS]
- Pilot Override Control [UAS]
- Crew Visibility/Canopy Defog
- Life Support (Crew O_2 Supply & Pressurization)
- Pilot Altitude and G protection
- Ground Operation for Armament /Expendables/Counter Measures
- Separation/Deployment of Armament /Expendables/Counter Measures
- Door Collision Prevention
- Gun Door Operation

INDICATION CRITICAL

The display of:

- Primary Flight References
  - Altitude
  - Airspeed
  - Attitude
- Heading
- Fuel Quantity
- Engine Health Monitoring
- Caution and Warnings
- Manual Terrain Following
- Communication Link Status [UAS]
- Flight Termination [UAS]
- Weight Off Wheels [UAS]
- Thermal Management Status [UAS]
- Electrical System Status [UAS]
- Navigation Status [UAS]

AVOIDANCE CRITICAL

- Bleed Air Leak Detection
- Ice Detection
- Ground Collision Avoidance System
- Air Collision Avoidance System
- Emissions Control
  - Toxins
  - Lasers
  - RF Energy
- Environmental Protections
  - Bird Strike Protection
  - Lightning Protection
- Turbulence Detection [UAS]

EMERGENCY CRITICAL

- Fire Detection
  - Engine
  - APU
- Fire/Explosion Suppression
- Crew Ejection
- Emergency Egress
- Cockpit Smoke/Fume Evacuation
- Fuel Shutoff
  - Engine
  - Lift Fan
  - APU
- Containment of Highly Energized Displaced Parts
- Flight Termination [UAS]

Activity Phasing Guidance

SCFs should be identified early in the development of a new air system. Likewise, if an SCF is being incorporated into an air system through a modification effort the SCF identification should be made very early in the development timeline. SCFs should be identified and agreed to no later than the System Requirements Review (SRR).

Attachment 2
DETAILED GUIDANCE REGARDING SCFTA

A proper SCFTA consists of three key steps that are applied to every SCF identified for the air system (for the purposes of Section 15 the focus is on SCF threads that are supported by CS&S). Each step identifies information about the SCF thread being analyzed as follows:

STEP 1 – Decompose: Identify all elements, components and interfaces that support the operation of a given SCF

STEP 2 - Classify: (1) The Computer System Integrity Level (CSIL) of the Safety Supporting Elements (SSEs) that are supporting the SCF; (2) The identification and designation of safety critical data and control interfaces supporting the SCF

STEP 3 - Analyzing V&V Coverage: The evidence that complete test coverage has been achieved from end-to-end across the SCF thread

An SCFTA needs to be performed for a new development and for every modification/update that impacts an SCF. Details for performing each STEP are provided below. Failure to follow the guidance provided will likely result in an insufficient SCFTA and contribute to non-compliance of Section15 criteria dependent upon SCFTA.

Certain airworthiness criteria in Section 15 focus specifically on Flight Critical functions. Flight Critical functions will be a subset of the functions identified on the program's SCF List. An airworthiness certification applicant may want to consider identifying SCFs that are Flight Critical during the SCFTA in order to efficiently address the criteria with a Flight Critical focus.

Guidance regarding SCFTA STEP 1 - *Decompose*

The analysis activity of this STEP involves decomposing the SCF in order to identify elements, components and interfaces supporting the SCF. The activity of identifying elements, components and interfaces requires that the SCF under analysis be functionally decomposed into the sub-functions that make up the SCF operation. The sub-functions are further decomposed until the level of detail is sufficient to uniquely identify all the components that support the SCF. The complexity of the SCF will drive the number of levels of decomposition needed to clearly identify the supporting components. As the decomposition is performed, elements (Safety Supporting Elements (SSEs), Safety Supporting Hardware Elements (SSHE), and Safety Supporting Software Elements (SSSEs)) of the system along with interfaces between supporting elements will be identified that support the SCF. The identification of these elements and interfaces supporting the SCF should be documented as part of the SCFTA. Once the functional decomposition is completed and the components and low level interfaces supporting the SCF are identified, these too are documented as part of the SCFTA and this information is then utilized for STEP 2. Ideally

this documentation process will organize the information by sub-function and include thread diagrams containing the thread's elements and interfaces. Elements and components that only provide source data to the SCF processing, or only receive data produced by the SCF processing, should be identified as SCF Data Sources or SCF Data Receivers accordingly (i.e., the beginning and end points of the SCF thread). The thread diagrams should also identify if there are redundant elements or interfaces in the thread architecture. It is recommended that the information be captured in a database system that can link the interconnectivity of the elements, components and interfaces. In STEP 3 it will also be useful if the database system can capture the traceability to test cases that are used to fully test the SCF thread.

Summary of the process for STEP 1 of SCFTA

1) Functionally decompose the SCF
2) Identify elements supporting the SCF, including any SCF Data Sources or SCF Data Receivers
3) Identify element interfaces supporting the SCF
4) Continue functional decomposition of SCF until all supporting components can be identified
5) Identify all components supporting the SCF, including any SCF Data Sources or SCF Data Receivers
6) Identify all component level interfaces supporting the SCF
7) Document all elements, components and interfaces identified during the analysis (recommend a database system that can link the identified items together to define thread)

Guidance regarding SCFTA STEP 2 - *Classify*

In STEP 2 of the SCFTA there is a focus on ensuring that each identified SSE and supporting components (both hardware and software) have been assigned an appropriate CSIL classification, which drives the set of safety assurance processes that are applied to an item's development and V&V (see page 36 in Attachment 6 for more on CSILs). Secondly, this step makes sure that the interfaces (buses, bus messages, data signals, etc.) are designated as supporting an SCF. Any parts of the SCF thread (e.g., elements, components, interfaces) not developed and verified to well-defined safety assurance processes (i.e., a CSIL classification) need to be clearly identified in the SCFTA. If parts of an SCF thread are not designed and developed in accordance with appropriate safety assurance processes this will result in several Section 15 criteria being non-compliant and the airworthiness applicant will have to provide risk assessments regarding those non-compliances.

The USAF Airworthiness Office has encountered numerous occasions in which it considered the CSIL assignment of an item to be inappropriately classified. Unique approaches for assigning a CSIL have been seen across the various aircraft system developments. This document will not attempt to establish a standard, however there are some given principles that we expect to be applied to any CSIL assignment process. Those principles are:

1) **CSIL assignments are applied entirely to the lowest level configuration items (CIs) identified in the system** – In most cases the lowest level CI identification is made at an LRU (or LRM) level for hardware, while software CI identification is typically made against a single (or set of) load image(s) that has a particular functional focus. Any given development effort however, may take a different approach in designating their lowest level CIs at a higher (e.g., subsystem level) or lower level (e.g., sub-assembly level or low level software sub-function). If a program has chosen to designate their CI assignment in some manner that doesn't make sense for CSIL assignments (i.e., there is a poor correlation between CI assignments and the scope of critical functionality provided by the CI relative to the overall SCF supported), it is recommended that the CSIL assignments be made against hardware assemblies that are typically considered LRUs/LRMs and against software items such that the identified item will result in a specific load image that will run on a given processing node (or if applicable, duplicated on several nodes). All components within a given element are to be assigned the same CSIL level as the element that they reside within has been designated, unless those components are sufficiently segregated from any influence or support to an SCF. Sufficient segregation is achieved by eliminating all functional coupling and direct electrical coupling to the components that do support an SCF. For software, all software components should be developed under the same CSIL as the CSCI CSIL designation unless that CSCI represents two unique load images in which one image is an SSSE and the other image is not. The non-SSSE load image software components can be developed at a lower level CSIL than the CSCI is designated. For hardware, if a given assembly is an SSHE and is assigned a CSIL all components of that assembly should be developed at that assembly's CSIL unless the assembly has been designed such that some components are fully isolated from, and have no potential influence (i.e., physically, functionally, electrically) on, the components supporting the SCF.

2) **CSIL assignments for elements supporting an SCF will always be categorized with an assignment that establishes processes designed to provide mitigation to safety risk associated with their functional operation** – Another way to say this is that an element's CSIL assignment will prescribe processes that are adequate for mitigating the inherent safety risk associated with the safety criticality of the element's function. Any element in an SCF thread must be treated as an SSE. As applicable to hardware or software, the attributes identified in Attachment 6 of this document are expected to be associated with the identified process definitions of any CSIL assigned to an SSE.

3) **CSIL assignments to elements/components are commensurate with the criticality of the functional support provided to the SCF** – When identifying the items in an SCF thread there needs to be a clear method for identifying what supports the SCF and what does not. An SCF is typically a high level function that is supported by numerous lower level sub-functions. All sub-functions that contribute to an SCF's nominal operation; fault detection and accommodation including redundancy management; functional, timing, and data integrity; mode control including interlock functionality; and health monitoring and status functionality crucial to functional operation are to be considered as providing support to the SCF. All elements/components that support an SCF should be assigned a CSIL that is intended to mitigate safety risk associated with SCFs (i.e., Catastrophic or Critical severity). When assigning a CSIL to an element/component the criticality of the SCF sub-function supported by the element/component should drive the criticality of the CSIL assigned to the element or component. If a program has only established one CSIL level that is suitable for hardware and software supporting an SCF, then every element and component in an SCF thread must be assigned that one CSIL. When a program has established two or more CSIL levels that are intended for hardware and software supporting an SCF, then there needs to be clear definitions for how those levels will be applied to the elements/components. All of the levels established for hardware and software supporting an SCF should possess at a minimum the applicable attributes that are defined in Attachment 6 of this document. Additionally, the USAF Airworthiness Office has observed varying approaches for identifying the criticality of a given hardware or software item. Some variation in approach is acceptable, however we expect certain attributes to drive the criticality assessment. Key drivers expected to be used for establishing criticality are: the functional criticality supported (hazard severity); applicable functional and safety oriented requirements; and if the function has been implemented with a fully independent secondary/backup functional channel (fully redundant and independent secondary/backup channels can potentially be assigned a lower CSIL see page 54 in Attachment 6). Design attributes that help mitigate safety risk associated with the functional support provided to an SCF should not be used as a driver in evaluating the safety criticality of hardware or software items in order to establish CSIL assignments for these items. The CSIL assignment is what should drive the process to utilize designs that mitigate the associated safety risk.

4) **CSIL assignments are not influenced by design mechanizations incorporated to mitigate safety risks** – A CSIL assignment, which is driving the level of process rigor, design, and verification activity, should not attempt to take design mechanizations that are incorporated into a system to mitigate the associated safety risk and then attempt to also justify that the process rigor can be lowered to a less

rigorous level. The mitigating design mechanizations were incorporated due to the functional criticality and the process rigor is also to be determined by that same functional criticality, not the combination of functional criticality and design choices. The functional criticality should drive the choices for the applied development process rigor, the design robustness, as well as the verification rigor equally.

As in STEP 1, it is recommended that a database system be utilized to capture the classification of these items. When modifying a system this information can be particularly useful in providing verification evidence that the full scope of the change impact to any SCF is fully understood. By utilizing the SCFTA thread information and associated safety assurance classification of all parts impacted by the modification, evidence can be provided that the scope of impact to airworthiness safety has been completely assessed.

Guidance regarding SCFTA STEP 3 – *Analyzing V&V Coverage*

The final step of the SCFTA is to analyze the multiple levels of functional and failure mode testing to ensure that all items in the thread have been adequately tested to achieve end-to-end V&V coverage of the SCF. This analysis step includes performing traceability analysis that ties the identified testing to the identified sub-functions of the SCF, while also showing that the components and SSEs supporting the SCF have also been fully tested. The testing identified needs to be at the system integration level, subsystem integration level, and at the box/LRU/LRM level. Not all testing identified needs to be formal testing. Much of the testing used to validate functional operation is expected to be non-formal testing. The SCFTA should continue to analyze the V&V coverage down to the component level (e.g., software integration and unit level testing, hardware component testing and analysis) and ensure the coverage relevant to the operation of the SCF is fully covered.

The component development process that underlies the SCF thread development and V&V has a substantial influence on the SCFTA product. Every effort should be taken to ensure a rigorous process is put in place and that the process is prudently followed. The attributes of a development process with sufficient rigor at the component level are as follows:

1. Requirements implemented (partially or wholly) through components that also support one or more SCFs, are tagged as such and traced to the SCFs they support
2. The traceability analysis has identified supporting components
    a. This traceability is necessary for safely supporting the system once released. Traceability enables identifying any components that may be contributing to an SCF in the event there are issues or anomalous behavior in the operation of the SCF.
3. Peer review instructions include guidance on inspecting SCF functionality
4. For hardware: component level acceptance, reliability and life analysis and/or testing is performed; traceability exists from a component to the analysis/testing performed on it

5. For software: all requirements, design, models used to generate auto-code, hand generated source code, and unit level test and qualification test products supporting SCFs are completed and peer reviewed
6. Software component (i.e., unit) level analysis and test is completed and sufficiently robust
    a. All test cases run and evaluated as passed, or if failed, adequately dispositioned for safety risk
    b. Software component design elements supporting SCFs are identified
    c. Adequately tested (e.g., nominal, off-nominal, robustness/stress, out-of-bounds)
    d. Component (i.e., unit) level testing is designed to achieve 100% functional coverage and required structural coverage
    e. Traceability exists from software component to all analysis/testing performed on it
    f. Flight Critical software is independently tested at the component (i.e., unit) level
7. Safety interlocks are properly identified, analyzed and tested.

If an SCFTA does not perform test coverage analysis at the component level and the component level processes are found to be inadequate against the above list, an airworthiness SME will likely find the SCFTA to be insufficient and this will impact the program's ability to be compliant with criteria reliant upon the SCFTA data.

If the analysis of testing identifies gaps in the test coverage of the SCF thread then resolving those gaps should be included in the SCFTA activity.

It is important that the SCFTA STEP 3 activity ensure that adequate functional test coverage of the SCF has been achieved. Adequate coverage should be evaluated on the basis that the SCF (and all the sub-functions that the SCF decomposes into) is functionally tested over the range of operation and performance expected of the system, including failure mode functionality. Types of testing that typically support non-component level testing for STEP 3 include: system/subsystem level testing (with and without operator-in-the-loop); functional requirements testing; functional validation testing; safety specific testing; FMET; and safety interlock testing. A portion of these tests will stress the dynamic and critical timing operations of the SCF considering as necessary any transitory responses and mode switching interactions. Component level testing coverage would typically be provided by: component level requirements testing, functional coverage testing, and robustness/stress testing (including failure and out-of-bounds testing). It is impossible to test every single test point that is possible to test, however coverage expectations do anticipate that the range of test points exercised will thoroughly cover the full scope of functional operation through the use of complimentary test coverage strategies. This is due to the fact that any given single strategy is typically unable to address all desired coverage expectations.

When making a modification to the system, one important aspect of airworthiness verification is ensuring unmodified SCFs were not unintentionally impacted. Regression testing is performed to provide the evidence that the unmodified functionality was not impacted. It is recommended that

when initially identifying the SCF thread testing that satisfies end-to-end V&V coverage, test cases that should be run as regression V&V should also be identified at that time. The set of regression tests that cover SCF regression are typically referred to as Core Tests or Core Regression Tests because they become part of a core, or repeated, test activity to formally release flight software.

Guidance on integration test coverage can be found in Attachment 3, which begins on page 21 of this document.

Activity Phasing Guidance

SCFTAs should be started at the point SCFs can be associated with the supporting elements (i.e., assemblies). The functional decomposition and identification of supporting elements and components should begin at some point between SRR and Preliminary Design Review (PDR). The activity of analyzing test coverage of the SCF thread should begin by Critical Design Review (CDR) and continue until testing has been performed and assurance of thread coverage achieved (this will likely take you near the point of an Flight Readiness Review (FRR) or equivalent release activity).

Attachment 3

DETAILED GUIDANCE REGARDING SYSTEM AND SOFTWARE INTEGRATION METHODOLOGY

Integration involves bringing less complex items together to into a more complex architecture and/or functional operation. For very complex systems the integration activity is broken into several levels. The lowest level is focused on component testing. Components are integrated together to form elements. Elements continue to be integrated together until a subsystem is formed. At the highest level the subsystems are integrated together to form the system. This pattern of integration activity is typified in the Engineering "V". A term generated from the V-shaped diagram used to show a system's development beginning with requirements development, leading to fabrication, and followed by relevant V&V activity performed before being delivered to customers/users for operational use. See Figure A3-1 below.

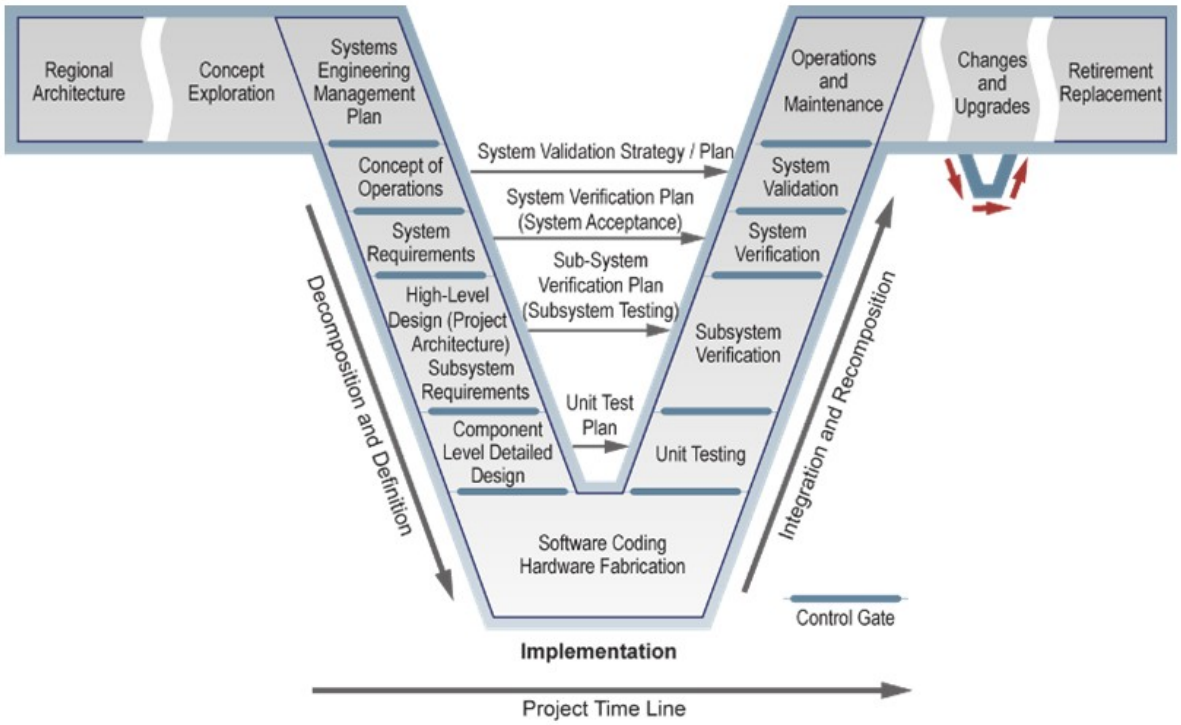


FIGURE A3-1: THE SYSTEM ENGINEERING "V"

Take note of the small "V" with the burgundy colored arrows going around it, which is located on the upper right side of Figure A3-1. This "V" is representative of the fact that changes and upgrades to the system need to have the System Engineering "V" applied to them like the system did when going through initial development.

The integration activity is typically focused on either verification or validation. Verification is the term used for the set of testing (or other accepted techniques such as analysis, simulations, etc.) that verifies that the product being built is built as specified. Verification testing uses formal testing and analysis activity to provide proof that a program requirement has been met. Formal testing has customer involvement as the customer will be accepting the results of the testing as evidence of some level of completion of the contracted work. Verification activity typically has a period of informal testing in order to prepare for the formal test activity. The validation activity consists of testing (or other accepted techniques such as analysis, simulations, etc.) that ensures that the product will meet the intended operational need or mission. If the specification did not adequately capture how the product was to be built in order to meet the operational need, it is possible to pass verification, but fail validation activities. Validation typically focuses on functional operation and key performance characteristics that are crucial to operational use.

Each level of the System Engineering "V" must focus on the relevant integration concerns appropriate for that level. For example, it is inappropriate to focus on component level operation when performing system level integration. The testing activity of each level must keep the focus for that level in mind and design the testing to address the focus of that level. See Figure A3-2 for a listing of testing levels and Figure A3-3 for a summary of the type of focus that is expected for various types and levels of software testing.

Level #	Software	Hardware
8	Unit Testing	Component Testing & Analysis
7	Software Integration	Hardware Integration
6	SW Qualification	HW Qualification & Acceptance
5	SPA/Subsystem Integration	
4	SPA/Subsystem Qualification	
3	System Integration	
2	System Qualification	
1	Operational Use Validation	

FIGURE A3-2: LEVELS OF TESTING

Type of Testing	Focus of Testing	Lvl#
Unit Test	Unit design (nominal, boundary, off-nominal conditions)	8
Component Test	Validation of Integrated design of software architecture	7
CSCI Test	Verification of software requirements	6
CSCI Integration Test	Validation of integrated design of SPA architecture & software & Verification of SPA requirements.	4-5
Subsystem Test	Verification of subsystem requirements/validation of performance	4-5
SCF Thread Test	Functional testing ***at every level*** focused on SCFs (up to system lvl)	3-8
Systems Integration Test	Validation of integrated design of air vehicle subsystems	5
Iron Bird Test	Validation of integrated design of air vehicle with real HW	3-4
Flying Test Bed	Validation of subsystem functionality & performance characteristics	3-4
Air Vehicle Level Test	Verification of system level requirements	2
Ground Test	Verification of system/user requirements & Validation of system function & performance	1-2
Flight Test	Verification of system /user requirements & Validation of system function & performance	1-2

FIGURE A3-3: TESTING FOCUS FOR VARIOUS TYPES OF TESTING

Regarding SCFs, each level of testing provides a unique set of verifications and validations in assuring the safety of the SCF thread. Every level needs to have FMET/fault injection/robustness/off-nominal condition testing along with testing the nominal operating conditions. A given level of testing will focus on one of the following areas:

- Functional validation
- Requirement verification
- Design V&V

Test coverage for the focus area of a given level of testing is the degree of completeness achieved by the scope of the test cases utilized. Adequate or ***complete test coverage*** is achieved when the area of test focus for a given level has been fully addressed by the scope of testing performed at that level. The following more specifically expresses how complete test coverage should be considered for each test focus area:

- Functional validation: Also known at times as system validation testing, this testing validates that the functional performance meets all of the expected functional operation and does not provide any unintended functional operation. Expected functional operation is defined in a variety of ways depending on the level of testing being performed, but all definitions of functional operation should be related to the intended system operation or the operational needs the system is intended to support. Nominal and off-nominal conditions must be addressed throughout the entire envelope of operation. Out-of-bounds conditions should also be tested along with FMET conditions. Testing of this type usually utilizes black box testing techniques to stress and validate the system.

- Requirement verification: This testing verifies that the item under test at a given level meets all the requirements established against that item. Ideally this type of testing utilizes procedures that use a black box design approach.

- Design V&V: The testing verifies that the design implemented matches the design documented, and that the design can support requirements. This testing will also validate key design attributes and ensure that the design provides acceptable performance and operation, while not introducing undesirable functionality. Design V&V should also include testing that confirms the robustness of a design by testing any potential weaknesses that may be associated with the design. The scope of testing should address nominal and off-nominal conditions, and include FMET. Testing with this focus will typically utilize a white box design approach.

Within Section 15 of MIL-HDBK-516C paragraph 15.2 addresses system integration and 15.6 addresses software integration. These two subsections are not separate and unique, but actually overlap each other. Software is viewed as being generated at a unit level and systematically integrated into more complex Computer Software Components (CSCs) until a Computer Software Configuration Item (CSCI) is formed. Software is typically viewed as having at least 3 levels of testing (i.e., unit, CSC, and CSCI) before the CSCI is ready for formal qualification and integration at higher levels. As higher levels of integration are performed, Section 15 takes the perspective that software is still undergoing validation testing. See Figure A3-4 for details on the type of software testing that is expected at each level.

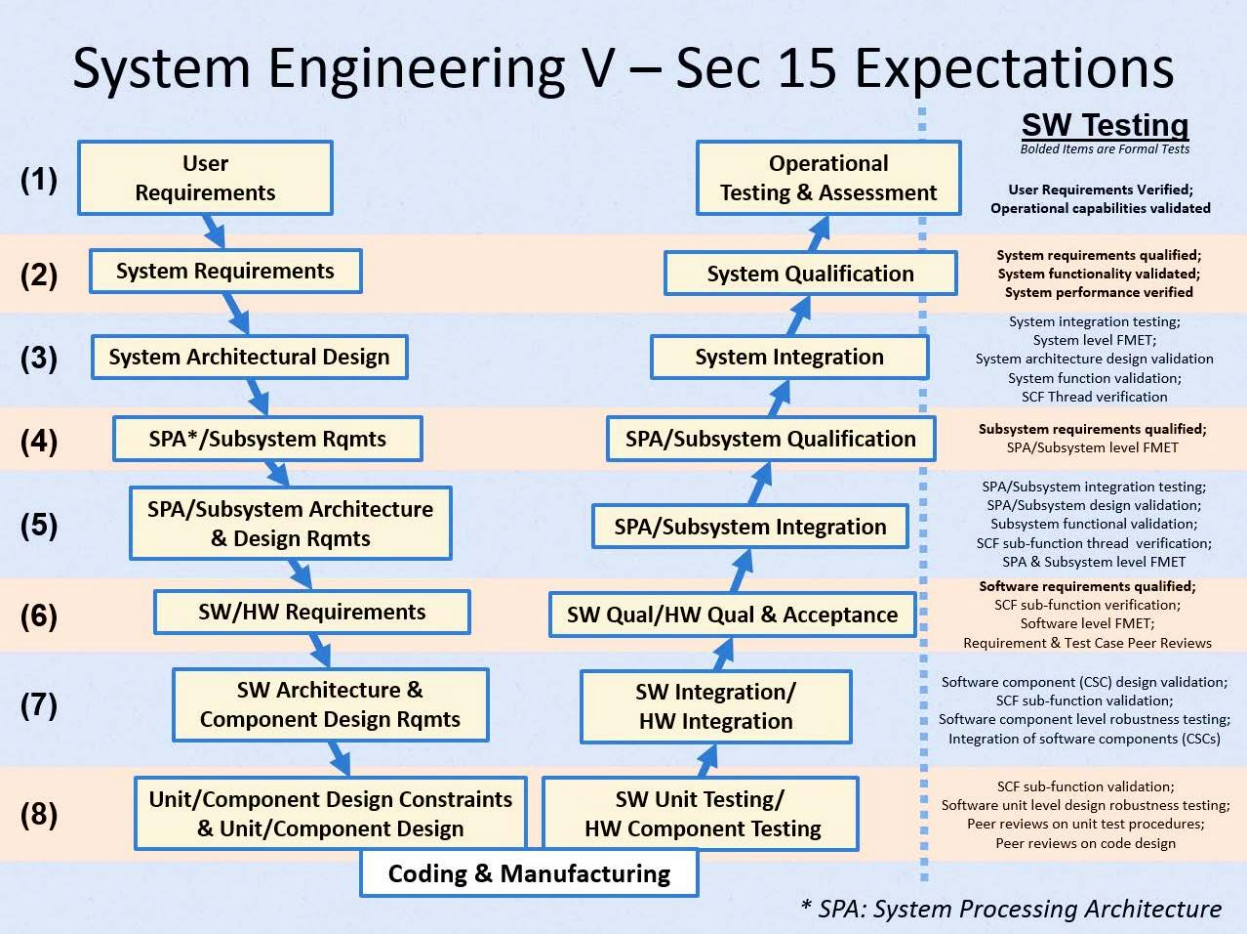


FIGURE A3-4: SYSTEM ENGINEERING V WITH SOFTWARE TESTING AT EACH LEVEL

For each level of testing, the test environments need to be tailored to the focus of testing for the level that the environment will be used to test. Below are a set of environments and the levels that are typically associated with them. For a given development effort not all environments may be utilized (e.g., iron bird tests may not be utilized for an avionics upgrade), but every level must be completely tested by the environment(s) used. Below is a list of test and integration environments that are typically used at certain levels during the integration phase of the System Engineering "V".

System/Software Integration Environments and

Levels of Testing Typically Performed within Each Environment

Software engineering test environment with simulated hardware

- Unit testing
- Initial software CSC integration

Software integration test environment with actual target hardware

- Software CSC integration
- CSCI testing and verification
- Software partitioning validation

SPA Integration Facility/Subsystem Integration Lab

- CSCI integration (e.g., multiple CSCIs)
- SPA architecture validation
- SPA requirement verification
- Subsystem design validation
- Subsystem requirement verification

System Integration Lab

- Systems integration testing, including operator-in-the-loop testing
- System requirements verification
- End-to-End SCF Thread testing

Iron Bird

- Aircraft subsystems (flight control, electrical, hydraulic, etc.) hardware/software integration

Flying Test Bed

- Aircraft mission systems testing

On Aircraft Testing

- Ground testing
- Flight testing

Modification Test and Regression Policy

When a modification is incorporated into a system the testing performed can be categorized into two categories. The first is the testing needed to verify and validate the modification being made. The second is to perform regression testing to ensure that the unmodified parts of the system were not unintentionally impacted by the modification effort.

When testing the modification it is necessary to perform an impact analysis that identifies the SCFs impacted by the modification. The impact analysis needs to identify the components of software and hardware that are being modified and the associated data flow, control flow and functional

coupling that will be touched by the modification. Once these items are identified, corresponding testing needs to be planned and executed, which completely tests the modified parts of the system.

When performing regression testing at the subsystem level or higher it is acceptable to perform a subset of all integration tests to demonstrate that no functional or performance regression has occurred in the unmodified portion of software. During initial development this subset of tests should be identified as Core Regression Tests. The release process should require that the Core Regression Tests are run for every flight release of software. The scope of coverage provided by the Core Regression Tests should provide comprehensive coverage of all SCFs and any other essential functions deemed critical to mission operation. When modifications are made that drive the need for additional testing, these new tests need to be analyzed to see if any should be added to the Core Regression Tests.

Activity Phasing Guidance

Integration methodologies should be defined by PDR.

Attachment 4
DETAILED GUIDANCE REGARDING FMET

Failure Mode and Effects Testing (FMET) is testing that intentionally injects (or causes) a failure in a system in order to observe the effect of the failure on the system including its response. FMET is typically performed to ensure that a system has adequate performance and is safe under the tested fault condition. FMET is a key verification technique for demonstrating Section 15 compliance regarding the implementation of SCFs and is associated with the following 22 criteria:

- Elimination of Single Point Failures (SPFs) in design (15.1.3)
- Accommodating data channel loss (15.1.6)
- Adequate separation of SSEs and non-SSEs (15.1.10)
- Operator notification of loss of critical processing (15.1.11)
- Fault Tolerance to power failure & interruption (15.1.12)
- Ensuring the necessity of functional coupling with SSEs (15.2.1)
- Avoiding dependence on single source safety critical data (15.2.2)
- Utilizing a sound integration methodology (15.2.3)
- Fault resilience associated with safety interlocks (15.2.6)
- Testing Single Event Upset (SEU) mitigations techniques (15.2.7)
- Hardware fault detection (15.3.1)
- Safety critical components segregated from non-critical components (15.3.2)
- Redundancy Management of safety critical data buses (15.3.3)
- Adequacy of software development process verifications (15.4.1)
- Software architecture & design supports sufficient level of safety (15.5.1, 15.5.2)
- Sufficient fault detection, Failure Management, Redundancy Management, operational flight modes, safety interlocks, SCF & health status interfaces, and reconfiguration under dynamic conditions (15.5.4 & 15.5.5)
- Accommodation of digital system failures (15.5.6)
- Sufficient restart and/or reset provisions (15.5.7)
- Software stress testing (15.6.1)
- Software load process can detect loading errors (15.6.5)

Other sections within MIL-HDBK-516, such as sections 6 and 8, also utilize FMET as an important verification technique. The expectations associated with FMET for these sections should be obtained from SMEs endorsed for the relevant verification criteria from those specific sections.

Well-designed FMET test cases provide V&V evidence that requirements based testing alone cannot provide. The FMET results should be used to perform the following:

- Verify injected failure is detected and reported
- Verify that resulting transients are acceptable
- Verify that the failure affects the system as expected
- Verify pilot receives the proper cautions and warnings
- Validate system design or reveal weaknesses/errors in the system design
- Assess that flying qualities and situational awareness are adequately maintained
- Identify improvements in the fault detection, isolation, and reporting coverage to give the pilot the best possible indication of mission capability
- Evaluate that the system design maximizes fault detection, isolation, and reporting to meet system supportability requirements and enhance maintainers' ability to identify required repair and the verification of repair after maintenance action is performed
- Minimize false or erroneous failure indications
- Determine and validate required pilots actions after the occurrence of failures

In figure A4-1 a flow diagram is provided to show how a typical FMET process flow is performed. FMET tests should be developed using requirement and design based drivers. These drivers are typically developed from system requirements, system architecture and design attributes, and various safety and failure analyses. Allocations need to be made at the various test levels and then test cases developed and run.

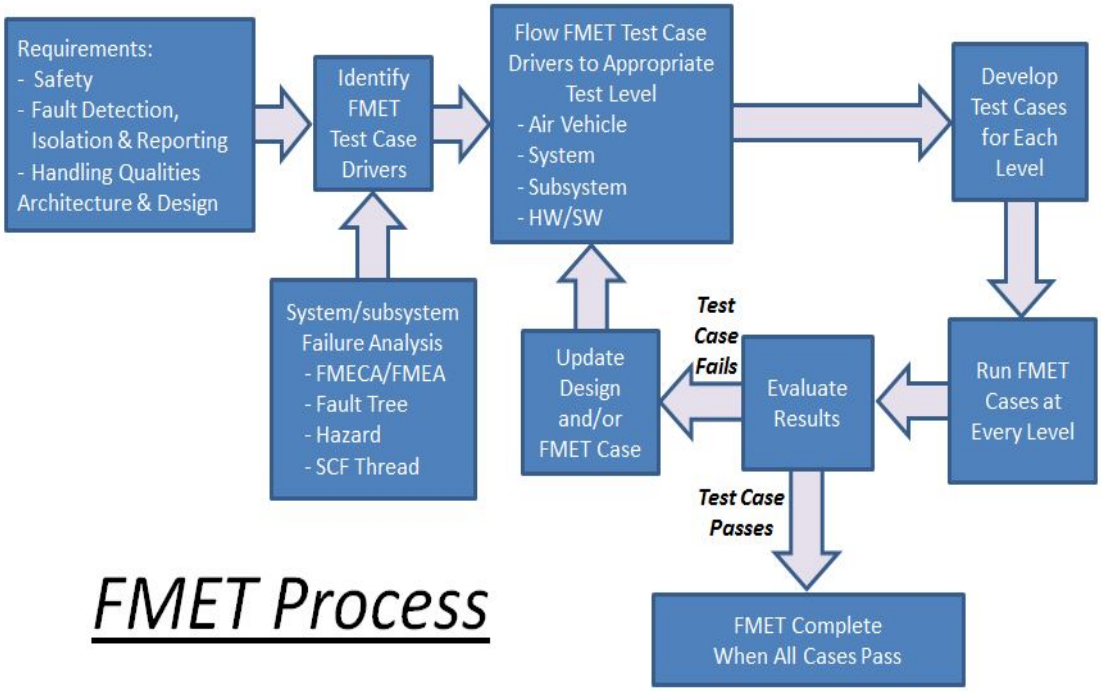


Figure A4-1: FMET Process

The FMET process begins by identifying the FMET test case drivers from system/subsystem requirements, architectural and detailed design of the system, and various failure analyses. FMET drivers define the basic areas of focus that the FMET tests need to be designed to stress. The list below provides examples of the type of system/subsystem requirements that feed into identifying FMET drivers:

- Safety
- Fault detection
- Fault isolation
- Fault reporting
- Probability of Loss Of Control (PLOC)
- Fail Operational/Fail Safe
- Design constraints such as no SPFs in the design of SCF thread
- Handling Qualities under fault conditions
- Minimal functional performance under failure safe conditions
- Transient response

Specific design choices in the architecture and detailed design will influence the FMET testing needed. Additionally, various failure analyses need to be used to generate the failure conditions that the FMET testing should inject into the system. The following provides example analyses that should be used to identify FMET drivers:

- FMEA/FMECA
- Fault Tree Analysis
- Hazard Analysis
- Safety Critical Function Thread Analysis
- System Processing Architecture Analysis
- Hardware And Software Architecture Design Analysis

Once the FMET drivers are identified, the scope of testing that they represent needs to be allocated to the various levels of testing in which FMET will be performed.

Robust FMET testing is performed at all levels of the development (see Figure A3-2 for levels of software testing). At the lowest level failure mode testing may be called stress testing or robustness testing. At this level an example may involve a software unit using failed data or out-of-bounds data to see how the code responds. For hardware an example of low level FMET would be an input/output (I/O) card that is fault tested by failing components on the card. This testing should also be supported by supplier failure data for all components used on the card. At the hardware integration and software integration level failure testing can begin to see any cascading effect in the integrated components. This is an important aspect of FMET all the way through system level testing. At the subsystem and system levels of FMET it is possible to stress the system to verify performance, system coupling, subsystem interaction, and compliance to requirements that fed the

FMET drivers. Additionally, the subsystem and system level FMET should also include some degree of pilot-in-the-loop testing to evaluate crew notification and response to fault conditions.

The injection of failures into a system can be done by a variety of ways. Test equipment is often designed to allow for signal disruption or to drive signals to specific values. Sometimes FMET cases may be conducted by actually introducing hard component failures into the system. This is referred to as destructive FMET because hardware components are actually destroyed in the process of conducting the test. Due to the cost of such testing, destructive FMET is only expected to be performed on new Flight Critical hardware when there is some uncertainty as to how an item will fail and/or be detected when it fails (e.g., failure analysis is insufficient, confidence or ability to detect failure is lacking). Non-destructive failure injection techniques are recommended to be used for Flight Critical FMET if those techniques can fully characterize the failure condition as well as the complete system impact/response. Destructive FMET is used it is not expected to be performed on subsequent modifications of a Flight Critical design if it is reasonable to assume that the results of the previously performed destructive FMET continue to apply to the modified design. Fault injection techniques utilized in the FMET of any SSE need to ensure that a full range of failure mode characteristics is introduced into the system.

FMET test cases should be designed to cover a broad spectrum of failure conditions. Below is a list of various conditions that should be covered by the complete set of FMET test cases:

- single point failures
- reset/restart cases
- dual failure cases
- triple failure cases with reset (if possible)
- quad failure cases with reset (if possible)
- combination of single point unlike failures
- order dependent failures
- loss of function
- loss of component
- loss of subsystem
- loss of redundancy
- loss of power
- degraded power
- loss of cooling
- loss of sensor
- loss of feedback
- loss of communication/CCDL
- degraded signals/communication
- system transient response regardless of the probability
- generic software error/SEU condition (if system susceptible)

FMET test cases should be traced to relevant SCF threads via the SCFTA. Once the tests have been run and results evaluated as acceptable the FMET activity is complete. For tests that do not pass, an assessment must be made as to why, and whether or not mitigation is necessary such as a design modification or change of system usage.

For a modestly complex system, the number of FMET test conditions needing exercised may rise to thousands of conditions. The question arises as to where a reasonable boundary can be drawn to limit the problem of "endless testing". The following guidance is provided as a means to define what should be included at a minimum within the necessary scope of all FMET type testing:

1. All single fail conditions (i.e., components; functions; communication signals, messages, and channels)
2. Sequential, like-fail conditions resulting in all-channel failure scenarios, repeated to address all order dependent possibilities
3. All critical software faults and digital system failure conditions identified by analysis
4. The full range of supply power errors to include loss, transient and degradation conditions (independent of probability of occurrence due to failure in power system)
5. Any error condition, regardless of probability, that is known to introduce a transient into control or communication associated with a SCF
6. Single bit error conditions for all critical data stored in volatile memory during runtime (i.e., SEU)
7. In addition to all the above, for multiple unlike failure conditions in which a probability of occurrence can be calculated, a probability of occurrence threshold can be established that defines which conditions must be tested based upon how likely it is to occur. It is recommended that this threshold be 2 magnitudes lower in probability than what is required for PLOC or the probability requirement for loss of critical functions (i.e., SCFs) For example, if PLOC is 10^{-7} losses per flight hour then the threshold should be set to 10^{-9}. If no PLOC or probability of loss requirement is specified then the threshold should be set to at least 10^{-9} losses per flight hour for vehicle control functions and 10^{-8} losses per flight hour for all other SCFs.

Activity Phasing Guidance

FMET planning should begin at the point that a top level design is in place and initial requirements and failure analysis is available that will be used to identify the FMET drivers. This should begin at some point between SRR and PDR. The activity of executing FMET test cases should begin around CDR and continue until testing has been completed (this will likely be near the point of an FRR or equivalent release activity).

Attachment 5

DETAILED GUIDANCE REGARDING SAFETY INTERLOCK DESIGN MECHANIZATION

Analysis and testing of Safety Interlocks (SIs) for airworthiness verification compliance must focus on the interlock designs and systematically examine and test the implemented mechanization. The basic process expected with the airworthiness verification activity for SIs is 1) Identify the SIs, 2) Analyze the SIs, and 3) Test the SIs.

Identifying SIs

By definition SIs are associated with SCFs. The utilization of the SCFTA can help scope where SIs reside within the design. Per the MIL-HDBK-516C's definition of interlock, SI identification will involve system design mechanization to enable or disable systems, functions, subsystems, or modes at given times and conditions. Measures should be taken to clearly identify SIs within the design. For Section 15 compliance, areas of the system design that utilize signals (analog or digital via discrete or multiplexed communication paths) to enable and disable functionality or modes that support or perform SCFs, need to be included in the identified list of SIs. Simple on-off switches that are not utilized in any additional control or interlock logic do not need to be identified as SIs unless these switches are able to be controlled by an automatic mechanization. It is recommended that a traceability database be utilized to capture identified SIs, which will help meet the traceability expectations in the following analysis and testing steps.

Analyzing SIs

The expectation associated with analyzing SIs involves examining the interlock logic, signals utilized within the interlock logic, and understanding the coupling the interlock logic and signals have with other aircraft functionality and modes. Complete condition tables/state diagrams should be generated for each SI. All signals utilized by the SI should be included in a condition table/state diagram as well as all timing or logic conditions that the interlock responds to. Coupling analysis should also be performed to understand how external modes and conditions may influence the SI logic. Coupling analysis should focus on two areas: (1) direct coupling influences that may occur through the utilized signals; (2) any possible indirect coupling influences that can exist through functional dependencies. All functional users of the SI output(s), and how each functional user uses the output(s) of the SI, need to be accounted for in the SI analysis. It is recommended that the analysis results be stored in the database used for the SI verification activity and be traceable to the specific interlock design mechanization that implements the SI.

Testing SIs

Stressing the SI design with test cases focused on each user of the SI output(s) is the final step in the SI verification activity. The SIs should be tested under all the possible logic and timing conditions that have been identified in the analysis for that SI. It is recommended that the test

cases used to validate the SI designs be traceable to the specific interlock design mechanization that implements the SI.

Activity Phasing Guidance

SI analysis should begin at the point interlock designs are available to be examined. This should typically be at some point between SRR and PDR. The activity of executing tests on SIs should begin shortly after CDR and continue until testing has been completed (this will likely be near the point of an FRR or equivalent release activity).

Attachment 6

DETAILED GUIDANCE REGARDING CS&S DEVELOPMENT PROCESS AND PRODUCT ATTRIBUTES

When considering how all of the computer system and software processes for development and V&V are documented for a given program it is typical to find that they are in a collection of documents involving various plans, processes, procedures, and instructions. Airworthiness efforts that must show compliance to Section 15 should begin considering very early in the program how their collection of software process documentation will support their airworthiness Compliance Report (recommended no later than System Requirements Review). At this same time consideration should be given to assure that the processes being employed will satisfy airworthiness verification criteria, since it is incredibly difficult (if not impossible) to address a process non-compliance at the end of a product's development. This attachment has been developed to help identify expected computer system and software development process and product attributes for Safety Supporting Elements (SSEs). However, it should be noted that this attachment is not attempting to define all process and product attributes, but only those that have a unique relevance to achieving airworthy computer systems and software.

Process specific criteria found in Section 15 are summarized below:

15.1 – System Processing Architecture (SPA)

- 15.1.1 – Safety Critical Functions (SCFs)
- 15.1.4 – SCF Threads
- 15.1.7 – Computer System Integrity Levels (CSILs)
- 15.1.8 – CSIL processes

15.2 – Design and Integration of SPA Elements

- 15.2.3 – Integration Methodology
- 15.2.4 – Critical Discrepancies
- 15.2.5 – Simulations, Models & Tools

15.4 – Software Development Processes

- 15.4.1 – Software Processes
- 15.4.2 – Traceability
- 15.4.3 – Configuration Management

15.5 – Software Architecture and Design

- 15.5.4 – Dynamic Operation
- 15.5.8 – Unsafe Techniques
- 15.5.10 – Safety Supporting Software Element (SSSE) Performance

15.6 – Software Qualification and Installation

- 15.6.1 – Software Test Methodology
- 15.6.2 – Full Qualification of Software
- 15.6.3 – Software Build Process
- 15.6.4 – Software Load Compatibility
- 15.6.5 – Software Load Process

EXPECTED CS&S PROCESSES & ACTIVITIES SUPPORTING AIRWORTHINESS

Software Identification: USAF utilizes the terminology of *Computer Software Configuration Item* (CSCI) (note: Software Configuration Item (SWCI) may also be used) to identify a specific portion of executable code, and any associated data files, as a unique configuration item. CSCIs, whether developmental or non-developmental software, need to be uniquely identifiable. Some software may simply be data files that are used by a CSCI, but not considered part of a CSCI. These data files should have a unique identification established for them that clearly identify what they are and how their configuration will be maintained. Programming instructions and associated data loaded into a firmware device are considered software and should also have their own unique identification if the firmware's programming instructions are not treated as a CSCI.

Other software that requires unique identification is Off-The-Shelf Software (OTSS). The development pedigree of OTSS could range from having originally been developed as a CSCI to something that was commercially developed and is used in a completely different manner than it was originally intended. Once software has been identified, key attributes about the software should also be associated with the software's identification. Two essential attributes for airworthiness are Development Pedigree and Computer System Integrity Level (CSIL). The attribute of Development Pedigree captures whether the software was a developmental item or a non-developmental item. Software that was created in-house or contracted to be developed by a supplier would be developmental software. OTSS that was modified would also be considered developmental software. OTSS or reused software that was not modified for use with the project would be non-developmental software. CSIL is a term defined in MIL-HDBK-516C that generically represents any approach for defining software safety criticality/integrity levels (also known as design assurance levels) and applying certain development and V&V processes based upon a given criticality/integrity level. Each software item identified in a project (developmental or non-developmental) should have a CSIL classification assigned to it that is commensurate with the functional criticality of the system supported by the software.

The software associated with a given CSCI is an arbitrary decision. The scope of content of a CSCI can be based on any factors the customer purchasing the system thinks are important. There are some things that should be given some consideration when designating the scope of a CSCI if you want to minimize costs associated with airworthiness verification. MIL-HDBK-516C Section 15 requires an SSSE CSCI (a CSCI supporting an SCF) to have a full FQT successfully performed before being released to flight. It is recommended that SSSE CSCIs minimize their non-safety related functionality. This recommendation is driven by the fact that when non-safety functionality of an SSSE CSCI is modified, Section 15 criteria will drive a full FQT to be performed even though the modification is just for non-safety functionality. Additionally, functional coupling with the SCF may also drive the modification to be Reportable when it otherwise would not be. Limiting the CSCI's scope helps to limit what software functionality has to undergo a full FQT, but this must be balanced with the fact that each CSCI carries with it the overhead cost of documentation and configuration management processes. Physically segregating SSSE and non-SSSE CSCIs,

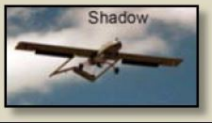
either on different processors or with a partitioning scheme, also helps to meet Section 15 verification criteria while limiting the verification cost impact when making a modification. Additionally, using the SCFTA to understand functional coupling through the system and smartly limiting CSCI scope to functionality within the SCF (or related SCFs) will help to reduce support costs related to verification as well.

SCF and SCFTA: SCFs for the program need to be established, typically by the System Safety organization (see *Attachment 1 – Detailed Guidance Regarding SCF Identification,* page 9 in this document). Software supporting SCFs needs to be identified as Safety Supporting Software Elements (SSSEs, as defined in MIL-HDBK-516C) and classified with an appropriate CSIL. Once identified as supporting an SCF, the software needs to be analyzed to identify the thread of software components that provide support to the SCF Thread. This analysis is called an SCF Thread Analysis (SCFTA) and additionally identifies the testing/V&V that is associated with verifying and validating the safeness of the thread's operation (see *Attachment 2 – Detailed Guidance Regarding SCFTA,* page 14 in this document). The SCFTA should be an integral part of the software development and safety hazard analysis processes. When evaluating test coverage or when identifying impacted portions of the software due to a modification, the SCFTA should be utilized in identifying functional coupling that needs to be evaluated in testing and V&V activity associated with the modification.

CSILs: The USAF has not yet established a standard set of design assurance levels (also known as defined levels of rigor) for computer system hardware and software development. The USAF has found that it is acceptable to have different processes supporting airworthiness safety for different types of developments. This permits developing unique process approaches that meet safety requirements while providing flexibility for balancing cost and schedule constraints. Computer System Integrity Level (CSIL) is the umbrella term that the USAF uses to generally describe the various safety process categories (for both hardware and software) that a development effort establishes to manage the applied processes that drive safety within development, design, and support airworthiness verification (see Attachment 2 page 14, under the section titled "Guidance regarding SCFTA STEP 2 – *Classify"* for guidance on CSIL assignment principles). All SSEs, SSHEs, and SSSEs need to be developed and verified with processes that meet acceptable safety standards in order to be compliant with MIL-HDBK-516 Section 15 criteria. Non-developmental SSEs, SSHEs, and SSSEs also need CSIL assignments that are commensurate with their use within the system. If the pedigree of a non-developmental item does not support the CSIL classification that is appropriate for its use within the system, process and/or design actions need to be implemented to mitigate the risk associated with using these non-developmental items. The down side to not having a standard design assurance approach is that there are times when programs choose to implement a process approach that meets cost and schedule constraints, but falls short of supporting airworthiness safety expectations. Below are a collection of expectations for CS&S development and V&V. For Reportable certification efforts, Section 15 Endorsed AW SMEs (Level II and III) will personally evaluate the applied CSIL processes for adequacy based upon the process expectations identified below as well as aspects of the implemented architecture

and lower level design. Programs are encouraged to have early and robust communication with Section 15 Endorsed AW SMEs to make sure their applied software development and V&V processes appropriately address all the items identified below.

Flight Critical CS&S Equipment: For manned air vehicles, or Group 3, 4, or 5 UASs (as defined by the *DOD Unmanned Aircraft System Airspace Integration Plan*, see Figure A6-1 below) that have computer hardware (anything supporting SPA operation) and software supporting SCFs that are critical to achieve and/or maintain controlled flight (including take-off and landing) of an air vehicle, this equipment will be considered Flight Critical CS&S items (i.e., SSEs, SSHEs, SSSEs, components). The Flight Critical category drives the most rigorous set of processes and V&V for equipment. Development programs need to ensure that that development and V&V processes supporting Flight Critical SCFs meet the expected rigor and discipline for these applications.

UAS Groups	Maximum Weight (lbs) (MGTOW)	Normal Operating Altitude (ft)	Speed (kts)	Representative UAS	
Group 1	0 – 20	<1200 AGL	100	Raven (RQ-11), WASP	
Group 2	21 – 55	<3500 AGL	< 250	ScanEagle	
Group 3	< 1320	< FL 180		Shadow (RQ-7B), Tier II / STUAS	
Group 4	>1320		Any Airspeed	Fire Scout (MQ-8B, RQ-8B), Predator (MQ-1A/B), Sky Warrior ERMP (MQ-1C)	
Group 5		> FL 180		Reaper (MQ-9A), Global Hawk (RQ-4), BAMS (RQ-4N)	

A6-1 - DOD UAS Group Definitions

Below is a list of process and product attributes that are expected for SSSE products, SPA products supporting SCFs, and their associated development and V&V processes. The attributes identified below have been written as if they are being applied to a production system seeking a type certification. Some attributes may be in conflict with the needs of pre-production systems. If there is any uncertainty in how these attributes should be assessed for a given system it is recommended that consultation be made with Section 15 Airworthiness SMEs in AFLCMC/EN-EZ. The list of attributes is long, so for the sake of readability the attributes have been grouped together under various group titles. A brief explanation of the group titles is provided to help with understanding the list.

SSSE Process and Product Attribute Groupings

1. Development Foundation: This grouping contains expected process and product attributes that are needed to establish a sound process foundation for supporting airworthiness safety in the development and V&V of SSSEs.
2. Basic Development: This group collects the expected process and product attributes associated with the common activities involved in the development of an SSSE.
3. Interface Control: This group identifies the expected process and product attributes that are to be associated with ensuring the integrity of the interface designs of SSSEs.
4. Avoiding Unsafe Attributes: This group identifies process and product attributes that the development and V&V process should ensure are not part of an SSSE.
5. Failure Detection and Accommodation: This group establishes the process and product attributes expected of SSSEs with regards to failure detection, accommodation, and/or redundancy management.
6. Configuration Management: This collection establishes the expected process and product attributes associated with configuration management processes supporting SSSEs.
7. Traceability: This is the grouping of expected process and product attributes associated with traceability activities/products for SSSEs.
8. Software V&V: This collection identifies the software test and analysis attributes expected in the V&V activities for SSSEs.
9. System V&V: This grouping contains the expected system (or subsystem) level V&V process and product attributes that the USAF believes should support SSSE airworthiness verification.
10. Building & Loading: This grouping identifies the expected build and load process attributes for SSSEs.

SPA Process and Product Attribute Groupings

1. SPA Level Development Foundation: This grouping contains expected process and product attributes that are needed to establish a sound process foundation for supporting airworthiness safety in the development and V&V of SPAs supporting SCFs.
2. SPA Level Basic Development: This group collects the expected process and product attributes associated with the common activities associated with the development of a SPA supporting SCFs.
3. SPA Level Interface Control: This group identifies the expected process and product attributes that are associated with ensuring the integrity of the interface designs of SPAs supporting SCFs.

4. SPA Level Failure Detection & Accommodation: This group establishes the process and product attributes expected of SPAs supporting SCFs with regards to failure detection, accommodation, and/or redundancy management.
5. SPA Level System V&V: This grouping contains the expected system (or subsystem) level V&V process and product attributes that the USAF believes should support airworthiness verification for SPAs supporting SCFs.
6. System Flight Clearance: This grouping of process attributes focuses on expectations associated with providing a flight clearance for the equipment of SPAs supporting SCFs.

Regarding any SSSE: The software product along with the associated development and V&V processes are expected to have the following attributes at a minimum for supporting airworthiness safety (Note: Some attributes are only expected to be applied to Flight Critical SSSEs and are noted with the phrase "FOR FLIGHT CRITICAL SSSEs:":

Development Foundation

1. All SSSEs are identified and documented
    a. All system software (executable code and data files) supporting an SCF needs to be identified. Typically documented in an SDP and the SCFTA.
2. SSSEs are developed with a rigorous, well documented software and system level development process suitable for producing and verifying high integrity software (i.e., at least addresses the process attributes identified in this attachment)
3. SSSEs are identified and documented as developmental (e.g., new hand code, auto-code, modified reuse), or non-developmental (e.g., off-the-shelf, reused, legacy)
    a. This helps identify what type of process and V&V activities are applied to the software
4. All SSSEs loaded in firmware are identified and documented
    a. Programmable logic instructions which are executed as a logic routine and support an SCF need to be identified. This could be done in the SDP or the SCFTA.
5. SSSEs are developed with a sound process for defining requirements, creating a design, coding the software, integrating the software product, analyzing for safety risk, validating algorithms and code design, and verifying requirement compliance
    a. Development processes that streamline to the point of eliminating these essential development activities are not considered suitable for developing and V&Ving SSSEs

6. CSILs properly assigned to all SSSEs
    a. The CSIL assignment, at a minimum, should include an associated process that addresses all the process and product attributes identified in this attachment
    b. When utilizing DO-178B/C, SSSEs should only be assigned Software Levels of A or B. Level C and lower are not considered by the USAF suitable for software supporting SCFs. (Note: DO-178B/C alone (Level A or B) will not address all the expected process attributes for SSSEs that are identified in this appendix.)
    c. FOR FLIGHT CRITICAL SSSEs: software is given a CSIL assignment that establishes processes that include all unique Flight Critical process and product attributes identified in this attachment
7. SSSE safety requirements identified and documented
    a. This should include all requirements that define the safety qualities required for the system and requirements that drive the design to meet the required level of safety. Examples include airspace class requirements, operational capability after failures, percent fault detection of critical failures (where a critical failure is one that impacts an SCF), reset capabilities, processing redundancy and associated redundancy management, critical timing performance requirements, etc.
8. Well defined and controlled software engineering environment established
    a. The development and the V&V equipment can be trusted to produce airworthiness verification data
    b. The hardware and software equipment are well defined and configuration controlled
    c. The software engineering environment is suitable for supporting and providing evidence of support to the process attributes identified in this appendix
    d. The software engineering environment supports designing and V&Ving the unique system performance essential to airworthiness safety
9. Performance requirements identified and documented
10. Requirements are robust
    a. Accurate, consistent, unambiguous, quantifiable, verifiable, with appropriate detail
    b. Software requirements are established from a clear allocation of system/subsystem requirements
    c. Software requirements trace to no more than, and no less than, one parent requirement
    d. Requirements are clearly identified and delineated from design
11. Software is integrated and tested in a multi-level approach
    a. A minimum of three levels should be utilized: unit level, software integration level (including hardware-software integration), and CSCI/requirements qualification level testing

Basic Development

12. For a given SSSE, the number of SCFs supported by it is accurately documented and reflected in the SCFTA.
13. All SSSEs are analyzed against all supported SCFs in the SCFTA
    a. The number of SCFs supported by an SSSE should equal the number of unique SCF threads that make reference to the SSSE
14. SQA performed
    a. The role of SQA is expected to provide assurance that the processes defined for the SSSE development are being faithfully followed
15. Coding standards supporting safety are utilized
    a. Coding standards should identify unique risks associated with the programming language/environment utilized and provide mitigations to those risks that are followed in the coding stage of development
    b. Coding standards should identify unsafe coding techniques that will not be permitted to be used in SSSE code
    c. Coding standards should identify coding practices that will facilitate consistency, readability, maintainability and safety in the code
    d. Use of a static code analysis tool is recommended to verify compliance with coding standards
16. Software safety process performed
    a. System safety process applied to software with code analyzed for hazards
    b. CSIL processes identified are appropriate for the criticality of assignment and meet the process and product attributes expressed in this appendix
    c. The identified CSIL process assigned to a given SSSE is shown to have been faithfully followed
17. Peer reviews conducted on requirement, design, code and test products and results documented
    a. The peer review process includes checklist guidance that requires safety attributes to be reviewed
    b. FOR FLIGHT CRITICAL SSSEs: peer reviews should include at least 2 independent peers with expertise in the software functional design area relevant to the product under review
18. Standard management practices utilized for managing software
    a. Process competency demonstrated for managing development progress, manpower support, etc.
    b. Risk management process accounts for airworthiness safety risks
    c. Metric tracking employed during development for appropriate airworthiness verification items (e.g., processor throughput utilization)

19. Software development files (SDFs) are sufficiently maintained
    a. Files are clearly associated with specific portions of the software
    b. Development, architectural, design, and implementation decisions
    c. Known software design limitations and constraints
    d. Developer's personal engineering analysis created in support of developing the software
    e. Information that must be considered when testing the software
    f. Informal test results
    g. Developer specific information generated in day-to-day development activities
20. Hardware-software integration testing performed
    a. Fully evaluates the compatibility of the software with the target hardware processing environment
    b. Target hardware is evaluated for ability to meet system/SCF performance requirements
21. Software problem reporting process is in place
    a. The problem reporting process should assess the safety impact of problems reported
    b. Reporting of problems identifies where the problems were found (type of environment), under what conditions, and what products are affected by the problem
    c. Problems are categorization based on the problem's impact to safety and mission operation. Categories are ranked in such a way that safety impacts are given the highest criticality rating. The following types of categories are established at a minimum:
        i. Safety impact no workaround
        ii. Safety impact with a workaround
        iii. Mission impact no workaround
        iv. Mission impact with a workaround
    d. Problem reporting process is able to reflect the historical safety categorization of problems through the development timeline
        i. For example, the change history of a problem report that was first categorized as "Safety impact-no workaround" then changed to "Safety impact with workaround" will not be lost by the problem reporting system
22. Comprehensive change impact analysis is performed when making software modifications
    a. A comprehensive change impact analysis will evaluate: any potential impact to SCFs; all data and control flow interfaces for coupling impacts; and analyze for any functional dependencies with other software internal and external to the software being modified.
    b. Applies to software that has been previously released for operational use.

23. Software architecture selection process accounts for providing safe SCF operation
24. Software design mechanizations are sufficient to support safe SCF operation and performance
25. Software architecture demonstrates compatibility with target hardware architecture
26. Model Based Design (MBD) tools and processes are suitable to support SCF
    a. Requirements are clearly segregated from design
    b. All MBD models trace to at least one requirement
    c. No problem reports exist against the auto-coding functionality of the MBD tool that would impact safety or code performance to the SCFs supported
    d. Models generated with a MBD tool are adequately peer reviewed
    e. Guidelines exist for unit size and complexity
        i. Unit size needs to allow for unit testing to be understandable and peer reviews to not be overly complex
        ii. Complexity thresholds are established and when met either drive effort to simplify design or apply enhanced processes to ensure integrity of design
    f. MBD tool emulators are qualified if used to produce evidence of airworthiness verification compliance
    g. Code generated by MBD tool is analyzed for functional and structural coverage as stipulated by defined CSIL processes
    h. Models generated by MBD tool are properly baselined and CM controlled
27. Execution rates are consistently obtainable
28. Execution rates support safe SCF operation
29. Process structure is safe
30. Process/task prioritization structure is sufficient to support safe SCF operation
31. Process execution is deterministic
32. Processes execute within deadline limits
33. Frame rate design meets real time system performance requirements
34. Interrupt structure fully documented
35. Interrupt structure and prioritization is sufficient to support safe SCF operation
36. Control structure is sufficient to support safe SCF operation
37. Executive control/operating system software is developed to the same CSIL (or higher) of any SSSEs it supports
38. Design requirements for execution rates, frame rates and latency tolerances defined
39. All initialization conditions are defined and documented
40. All synchronization requirements are defined and documented
41. Redundant processing tasks, data channels, and multi-sourced data are properly synchronized

42. All design timing requirements are defined and documented
43. Data structures are sufficient to support safe SCF operation
44. Process data dependencies are defined and documented
45. Safety interlock mechanizations are sufficient for safe SCF operation under all dynamic conditions anticipated
46. Static code analysis performed to evaluate data flow, control flow, complexity, and standards compliance
    a. Complexity thresholds should be established such that when the thresholds are exceeded the development policy either drives effort to simplify design or apply enhanced processes to ensure integrity of design
47. Processing throughput does not exceed 90% utilization under peak/worst case conditions
    a. The peak/worst case test condition should be established from an analysis of potential scenarios that could actually occur in the system. Unrealistic or impossible stressing scenarios do not need to be considered when demonstrating worst case utilization.
    b. Multi-core processing measurements are to be made per core
48. Real-time volatile memory utilization does not exceed 90% utilization under peak/worst case conditions
49. Memory bus utilization does exceed 90% utilization under peak/worst case conditions
50. Data channel throughput utilization does not exceed 90% utilization under peak/worst case conditions
51. Performance deficiencies identified and documented in software release documentation

Interface Control

52. All control flow analyzed and documented
    a. All synchronous and asynchronous control flow that influences the execution of an SCF is identified and analysis of design integrity and robustness is performed
    b. Use of a static code analysis tool is recommended
53. All data flow analyzed and documented
    a. All data utilized in support of executing an SCF is identified and analyzed for the functional utilization and/or transformation of the data throughout the entire SSSE
    b. Use of a static code analysis tool is recommended
54. SCF functional and health status interfaces are sufficient for supporting safe SCF operation under all dynamic conditions anticipated
    a. SCF interfaces are SSSE interfaces enabling communication with entities external to the SSSE

Avoiding Unsafe Attributes

55. Design does not have any unacceptable hazards
56. Does not have patched executable code
    a. The intent of this attribute is to prevent the flight use of software containing modifications to executable code or modified data files that have not fully undergone the approved CSIL process requirements before it is released for flight
57. Does not have deactivated code
    a. Deactivated code does not refer to software that disables access to portions of code based upon operating conditions (e.g., ground maintenance mode vs flight mode). Deactivated code does refer to code that exists within the loaded software configuration that has been disabled and is never intended to be used during planned operation of the configuration being certified.
58. Does not have flight test unique software
59. Does not have dead code
60. Does not have training system hooks
61. Does not have lab unique functionality
62. Does not contain code reused from software with unknown or questionable pedigree
    a. Software code with questionable pedigree would be:
        i. Code from an application whose development and V&V processes are unknown
        ii. Code from a non-safety application
        iii. Code that was not developed to the CSIL level that is required for the current SCF supported
        iv. Code in which the requirements, design, or source code is not known or is unavailable
        v. Code where the software testing at all levels is unknown, not accomplished or cannot be accomplished
63. Does not have undocumented functions
    a. An undocumented function that is in a given configuration of software is one in which the functional capability does not have requirements or design detail properly documented or traced
64. Does not utilize dynamic memory allocation
    a. This is to mitigate against the risk of memory leaks or memory mismanagement that could cause memory resource issues during run-time operation
65. Does not have unverified source code statements and structures
66. Does not alter source code to achieve requirement verification

Failure Detection and Accommodation

67. Software fully analyzed for potential software fault conditions and response to system failure conditions
68. FMET fully performed to evaluate software contribution to fault detection, isolation and accommodation
    a. FMET should be performed in a manner that complies with the guidance of Attachment 4 of this document
69. Failure monitoring is sufficient for safe SCF operation under all dynamic conditions anticipated
    a. Failure monitoring refers to the scope of monitoring capability to detect various types of failure conditions. The software should be designed to detect all types of failure conditions applicable to the system if practicable. Special failure conditions due to operational performance dynamics or variations in envelope of operation should be accounted for in the failure monitoring V&V activity.
70. Failure detection techniques are sufficient for safe SCF operation under all dynamic conditions anticipated
    a. Failure detection techniques refer to the manner employed to detect a given failure condition. The techniques utilized need to provide reliable performance over the full range of dynamic operation expected of the system.
71. Failure management functions sufficient for safe SCF operation under all dynamic conditions anticipated
    a. Failure management refers to the system's capability to react to a detected failure and then provide appropriate response and notification. Appropriate response involves ensuring that the system does not rely upon the failed item and any health monitoring that may permit the failed item back into use after being declared failed. Appropriate notification included any necessary notification to air crew to maintain safe operation and maintenance personnel to ensure repair before next use.
72. Redundancy management capability is sufficient for safe SCF operation under all dynamic conditions anticipated
    a. Redundancy management refers to the system's ability to manage redundant capability of the system, especially in the event that some part of the redundant capability fails. This includes voting logic used to evaluate the health of redundant items, output logic to ensure integrity of command signals, and the control capability to continue safe operation when a failure reduces the available redundant capability.
73. Voting schemes sufficient for safe SCF operation under all dynamic conditions anticipated

74. Built-In-Test (BIT) and self-checks are sufficient for safe SCF operation under all dynamic conditions anticipated
75. Reconfiguration capabilities sufficient for safe SCF operation under all dynamic conditions anticipated

Configuration Management

76. The configuration management (CM) process controls the configuration of system products, processes and related documentation and is fully documented
    a. The CM process provides complete and adequate configuration status accounting
    b. The CM process archiving capability provides for off-site archival from the primary development site to provide for recovery of flight released configurations in the event of disaster
77. CM is performed on essential development and V&V data for hardware, software, and firmware
78. HWCIs and CSCIs have been identified for all hardware, software, and firmware supporting an SCF
79. HWCIs and CSCIs supporting SCFs are identified as such (i.e., SSHE and SSSE respectively)
80. The CM process provides for a complete audit trail of decisions and design modifications
81. CM process establishes fully defined functional and product release baselines that identify unique configurations and performance capabilities for all hardware, software, and firmware supporting SCFs
82. The CM process shall make provisions for archival, retrieval, and release of software products and CM data
    a. CM process supports reconstruction of all historical baselines
83. Change control is managed off of defined baselines
84. The CM process provides for adequate technical oversight to the change control process
85. Composition of the Software Review Board (SRB) (also known as a software Configuration Control Board (CCB)) is made up of appropriate stakeholders with adequate authority
    a. The SRB membership has at a minimum the Software Project Lead, Project Management Lead, Lead Functional Engineer, Test Team Lead, and System Safety as representatives
86. CM process directly supports the software build process
87. Product baselines identify all known deficiencies with the product
88. CM process establishes a compatibility matrix/database for all installed air system software

89. CM process ensures all versions of installed air system software are functionally compatible
    a. This expectation applies to line loadable and depot loadable software
90. CM process supports the air system in providing a means to verify the correct software configuration is installed (executable code and data files)

Traceability

91. Traceability database is utilized that facilitates linking of traceable objects
    a. All traceable items (e.g., requirements, design, SCFs) can be captured in the database as a unique object that can be traced to multiple objects
92. Bidirectional traceability established from software requirements to parent requirements up through system requirements
93. Bidirectional traceability established from software requirements to design
94. Bidirectional traceability established from design to source code
95. Bidirectional traceability established from source code to test cases
96. Bidirectional traceability established from software requirements to test cases
97. Bidirectional traceability established from test cases to test procedures
98. Bidirectional traceability established from software requirements to supported SCFs
99. Bidirectional traceability established from test cases to supported SCFs
100. Bidirectional traceability established from source code to SCF threads
    a. The trace to source code will support the SCFTA verification activity
101. All source code in a software flight release traces to a software requirement

Software V&V

102. Unit level testing performed when created (or modified) and results documented
    a. Testing provides 100% functional coverage of the unit
    b. Testing stresses the logic design of the unit
    c. Testing uses nominal and off-nominal conditions
    d. Testing includes robustness cases stressing within-boundary, on-boundary, and out-of-boundary conditions of key internal parameters and external interfaces
    e. All unit test cases documented and traceable to unit ran against
    f. FOR FLIGHT CRITICAL SSSEs: unit testing conducted with independence from coder/designer
103. CSC/integration testing performed and results documented
104. The software (e.g., configuration item) is fully integrated together
105. Qualification environment utilizes target hardware
106. Qualification environment designed to provide a representative, high fidelity dynamic response
107. Qualification environment is capable of fully representing full envelope operation

108. Software design requirements are fully verified
109. Mode inputs and defined states are sufficient for safe SCF operation under all dynamic conditions anticipated
110. Operational flight modes are sufficient for safe SCF operation under all dynamic conditions anticipated
111. Software requirements verification (qualification) performed and results documented by an independent test team
    a. Software requirements are proven to be met
        i. A release with any unmet requirements is evaluated for safety risk
112. Full qualification of a CSCI is performed for every flight release
    a. 100% of requirement coverage achieved during qualification activity
    b. All SCF functional requirements verified
    c. All SCF performance requirements verified
    d. All safety requirements verified
    e. Qualification performed on target hardware
    f. Results documented
    g. All test failures captured in problem report database and properly dispositioned
113. Structural coverage analysis of software testing is appropriate and complete
    a. Any deficiencies in coverage appropriately justified
    b. Structural coverage analysis is performed against the source code and meets Condition/Decision coverage and Statement coverage at a minimum.
    c. Results documented
    d. FOR FLIGHT CRITICAL SSSEs: Structural coverage analysis is performed against the object code and meets Condition/Decision coverage and Statement coverage at a minimum; the analysis should include identification and verification of all object code statements that do not directly trace to source code statements.
114. Functional operation of SCFs fully validated
115. All qualification and integration test anomalies adequately dispositioned for airworthiness safety impact

System V&V

116. Qualification environment utilizes target hardware
117. Qualification environment designed to provide a representative, high fidelity dynamic response
118. Qualification environment capable of fully representing the full envelope of operation
119. System/subsystem performance requirements supported by software are verified
120. System/subsystem safety requirements supported by software are verified
121. SSSEs fully integrated together
    a. Integration effort address full scope of integrated software functionality

122. SSSEs fully validated to support SCFs
123. Subsystem level testing performed by independent test team
    a. Subsystem level regression testing performed on modifications utilizing, at a minimum, a core test set that provide sufficient functional coverage of all supported SCFs
    b. Results documented
124. System level testing performed by independent test team
    a. System level regression testing performed on modifications utilizing, at a minimum, a core test set that provides sufficient functional coverage of all supported SCFs
    b. Results documented
125. All system/subsystem level anomalies adequately dispositioned for safety impact

Building & Loading

126. Build process is robust and fully documented
127. Build process sufficiently safe to support SCF operation
128. Build process is consistently repeatable
129. Build process identifies all equipment needed
130. Build process does not corrupt loadable products or introduce errors into the software
131. Build process addresses security requirements
132. Build process identifies expected time to produce loadable products
133. Build process ensures loadable products are complete
134. Build process ensures loadable products are compatible
135. Build process addresses the production of all loadable executable code files and data files
136. Build process is version aware
137. Build process accounts for unique load image requirements
138. Air system software load process is fully documented and addresses loading all system software executable code and data files
    a. All equipment needed for software loading and load verification is identified
    b. Load process provides clear instructions how all box/depot loadable software is installed on the system
139. Air system software load process is suitable to support safe SCF operation
140. Air system software load process is consistently repeatable
141. Air system software load process verifies that installed software was loaded correctly and completely

142. Air system software load process verifies that installed software was not corrupted during load
143. Air system software load mechanization detects errors in installation
144. Air system software load mechanization notifies installer of error conditions
145. Air system software load mechanization prevents unauthorized use
146. Air system software load process verifies all box/depot loaded software

Additionally the following system/SPA level product and processes attributes are expected to be associated with an airworthy system:

SPA Level Development Foundation

147. System's SCFs are identified
148. System integration and system V&V methodologies and processes are fully documented
    a. Tiered build-up and test approach utilized to verify SPA requirements, operation, and performance
    b. System integration methodology and approach is consistent with Attachment 3 of this document
149. CSIL processes are adequate for the criticality of support provided to SCFs
    a. For software the CSIL processes are adequate when they support the process expectations identified in this attachment; applicable to firmware as appropriate.
    b. For CS&S hardware the CSIL processes are adequate when they ensure the environmental and functional performance of the hardware support the requirements associated with executing the supported SCFs over the entire envelope of expected operation; applicable to firmware as appropriate.
150. SPA safety requirements documented
151. SPA functional & performance requirements supporting SCFs documented
152. Power safety requirements defined and verified for SPA SSEs
    a. These requirements should include safety oriented requirements such as uninterruptable power, emergency power requirements, and power performance requirements needed in support of SCFs

SPA Level Basic Development

153. CSIL processes are applied to all SPA elements and verified to be followed
154. SCF threads identified and traced to all supporting equipment (SSEs, components, and interfaces), see Attachment 2 in this document
155. Flight Critical SCFs identified and documented along with associated processing systems
156. As applicable, PLOC, PLOA, and probability of loss of critical function requirements are specified and verified

157. SPA design mitigates the occurrence of SEUs for systems operating above 35,000 feet.
158. Requirements and/or design guidelines are established to identify and avoid/limit data and functional coupling
159. Software partitioning mechanizations are developed to the highest level CSIL of executable applications managed
160. All safety interlocks are identified and documented
161. All safety interlocks have their data coupling and functional coupling documented
162. Development team utilizes a problem reporting process associate will all SSEs, SSHEs, and SSSEs, which assesses and documents any safety impacts with identified problems
163. Problem reporting process utilizes a review board to review and close out problem reports
164. SPA technical & safety risks identified and mitigation management performed
    a. Includes hazard analysis and mitigation management activities
165. SPA reliability analysis performed

SPA Level Interface Control

166. Interfaces supporting SCFs have been fully documented
167. Data and functional coupling between SSEs supporting an SCF thread is documented
168. SPA interface control flow documented
    a. All synchronous and asynchronous control flow that influences the execution of an SCF is identified and analysis of design integrity and robustness is performed
169. SPA interface data flow documented
    a. All data generated within the SPA or transferred between elements within the SPA, which is utilized in support of executing an SCF, is identified and analyzed for the functional utilization and transformation of the data throughout the entire SPA
170. SPA interfaces have had data/calculation/timing dependencies verified to support safe SCF operation
171. SPA interface safety, functional, and performance requirements documented and verified
172. SPA support for the switching of UAS command & control (C2) data links does not result in loss of control and is sufficient for safe SCF operation under all dynamic conditions anticipated
    a. C2 data links in this context refer to UAS data links that provide communications that control (directly or indirectly) the flight of the UAS. Switching of C2 data links refers to the action of switching from one type of C2 data link (e.g., UHF LOS, Ku LOS, Ka BLOS) that is being used to control the UAS air vehicle, to a different type of C2 data link that will then begin to be used to control the UAS air vehicle after the switch.

SPA Level Failure Detection & Accommodation

173. FMEA/FMECA performed on SPA elements
    a. Analysis verifies no SPFs in SPA architecture
    b. Functional impacts of potential software faults are evaluated
174. FMET performed on SPA
    a. Testing verifies that all single failure conditions can be detected, isolated, and accommodated safely
    b. Comprehensive FMET performed on SCF threads
    c. SPA design can detect 100% of all critical failures
        i. Expectation is for continuous automated BIT checks to provide the vast majority (i.e., > 90%) of detection coverage for critical failures; remaining coverage obtained by initiated BIT that checks for latent failures and failure conditions requiring intrusive BIT.

SPA Level System V&V

175. All SPA requirements verified
176. Complete V&V coverage is achieved for SCFs
    a. End-to-end V&V coverage is achieved for SCF threads
    b. May require multi-level V&V activity to fully achieve end-to-end coverage
177. Conditions and limitations for utilizing delta testing (less than full set) are clearly defined
    a. Full testing should be required at periodic intervals or times of major modification
178. Adequate regression testing performed on unmodified SPA elements
    a. If a full regression (e.g., all system level tests run) is not performed, core tests are run that provide adequate regression test coverage of all supported SCFs
179. All development and V&V tools are qualified
    a. Tools are verified to ensure they are capable of providing accurate and reliable airworthiness verification evidence
180. V&V tools requiring calibration are calibrated prior to being used to produce airworthiness verification evidence
181. Development status of tools are categorized and documented as off-the-shelf, modified-off-the-shelf, or in-house developed
182. End-to-end V&V coverage of every safety interlock is performed
183. Implemented security techniques are verified to provide safe SCF operation and do not introduce hazards into the system

System Flight Clearance

184. Air Vehicle software release process ensures proper levels of regression testing and delta testing are performed

185. No unresolved/open safety critical problem reports exist against flight released hardware or software
    a. In the rare case that a flight release is made with an open problem report that has a safety critical impact, operational limitations and restrictions are in place to prevent the safety critical impact from occurring
186. Problem reports mitigated through aircraft limitations are documented and reviewed in the flight clearance process
187. Flight clearance process ensures released configuration has accomplished all release process requirements, software has been fully tested, hardware has been SOF tested at a minimum, and the cleared configuration is fully compatible.

When assigning a specific CSIL to elements that support a non-Flight Critical SCF in which a redundant, fully independent path of elements also supports the same SCF, the USAF Airworthiness Office's expectation is that the primary functional path (the one primarily depended upon for every day, nominal operation) will be developed and verified at the highest appropriate CSIL and will be designed with an appropriate level of redundancy. If the other fully independent functional path of the SCF is purely secondary (or providing backup functionality) the path's elements should be developed and verified to a CSIL appropriate for SSEs, but can be lower than the primary path's CSIL if a lower level CSIL sufficient for SSEs has been defined. If the primary and secondary functional paths are not truly independent then both should be developed at the same CSIL level appropriate for the SCF implemented. Independent, redundant paths of Flight Critical SCFs should have all paths developed to the highest CSIL. If a monitoring function is used in conjunction with a primary functional path, then both should be developed at the same CSIL level appropriate for the SCF implemented.

Formalized or standardized software development processes that are utilized to manage the software development must follow sound system engineering principles. The development process utilized needs to address the standard set of system engineering development phases as follows:

1. Requirements
2. Design
3. Coding
4. Unit Testing
5. Software Integration
6. Software Requirements Verification
7. Subsystem and System V&V

Variations in the specifics of how and when these phases are implemented may be acceptable, but clear justification for deviation from standard, recognized system engineering principles should be documented along with how the deviations do not compromise the safety integrity of the software. A number of defined software development processes are not suitable for the development of

SSSEs. Agile Development as originally conceived and typically implemented in the commercial sector is one that falls into this category. Modification of the Agile Development process to account for the process integrity needed for SSSEs may be possible, but at the time of publication of this document the USAF is not aware of a defined and proven process that successfully modifies Agile Development to provide the needed process integrity for SSSEs.

All software processes, Agile or otherwise, must be evaluated on their own merits and weaknesses to meet program OSS&E, staffing and schedule objectives.

Activity Phasing Guidance

Software process planning should begin before (or at) contract award. Although some definition of the planned development processes are typically in place at contract award, it is very typical that the processes will be reviewed and updated post contract award. SPA and software development processes should be updated and finalized by SRR. SPA and software V&V related processes should be updated and finalized by PDR.

Attachment 7
ABBREVIATIONS, ACRONYMS, AND GLOSSARY OF TERMS

Abbreviations and Acronyms

AC – Airworthiness Circular
AW – Airworthiness
APU – Auxiliary Power Unit
BIT – Built-In-Test
BLOS – Beyond Line Of Sight
C2 – Command and Control
CCB – Configuration Control Board
CCDL – Cross Channel Data Link
CDR – Critical Design Review
CG – Center of Gravity
CM – Configuration Management
CS&S – Computer Systems & Software
CSC – Computer Software Component
CSCI – Computer Software Configuration Item
CSIL – Computer System Integrity Level
EN-EZ – Air Force Materiel Command, Life Cycle Management Center, Engineering and Technical Management/Services Directorate
FF – First Flight
FD & Rep – Fault Detection and Reporting
FHA – Functional Hazard Analysis
FMEA – Failure Modes Effects Analysis
FMEA/T – Failure Modes Effects Analysis/Testing
FMECA – Failure Modes, Effects and Criticality Analysis
FMET – Failure Modes Effects Testing
FRR – Flight Readiness Review
G – Gravity
HQ – Handling Qualities
HW – Hardware
HWCI – Hardware Configuration Item
I/O – Input/Output
Ka – 26.4-40 GHz electromagnetic energy band
Ku – 12-18 GHz electromagnetic energy band
Lab – Laboratory
LOR – Level of Rigor

LOS – Line Of Sight
LRU – Line Replaceable Unit
LRM – Line Replaceable Module
lvl – level
MACC – Modification Airworthiness Certification Criteria
MBD – Model Based Design
O2 – Oxygen
OPR – Office of Primary Responsibility
OSS&E – Operational Safety, Suitability, and Effectiveness
OTSS – Off-The-Shelf Software
PDR – Preliminary Design Review
PLOA – Probability of Loss Of Aircraft
PLOC – Probability of Loss Of Control
Qual – Qualification
Rqmts – Requirements
SCF – Safety Critical Function
SCFTA – Safety Critical Function Thread Analysis
SDF – Software Development File
SDP – Software Development Plan
SES – Senior Executive Service
SI – Safety Interlock
SME – Subject Matter Expert
SPA – System Processing Architecture
SPF – Single Point Failure
SQA – Software Quality Assurance
SRB – Software Review Board
SRR – System Requirements Review
SSE – Safety Supporting Element
SSHE – Safety Supporting Hardware Element
SSSE – Safety Supporting Software Element
SW – Software
TAA – Technical Airworthiness Authority
TACC – Tailored Airworthiness Certification Criteria
TRR – Test Readiness Review
UAS – Unmanned Aircraft System
UHF – Ultra High Frequency
USAF – United States Air Force
V&V – Verification and Validation

End of Abbreviations and Acronyms

GLOSSARY OF TERMS

Note: Definitions with an * are found in the definitions section of MIL-HDBK-516C; Definitions with a document reference in brackets at the end indicate where the definition was taken from.

Air System*
An air vehicle plus the training and support systems for the air vehicle (e.g., communications, control, ground/surface/control station, launch and recovery, and support elements), and any weapons to be employed on the air vehicle. (Reference: JSSG-2000). For example, an Unmanned Aircraft System (UAS) is an air system. An air vehicle, manned or unmanned, is a subset of its associated air system.

Air Vehicle*
An air vehicle includes the installed equipment (hardware and software) for airframe, propulsion, on-board vehicle and applications software, communications/identification, navigation/guidance, central computer, fire control, data display and controls, survivability, reconnaissance, automatic flight control, central integrated checkout, antisubmarine warfare, armament, weapons delivery, auxiliary equipment, and all other installed equipment. [JSSG-2001]

Airworthiness*
The property of a particular air system configuration to safely attain, sustain, and terminate flight in accordance with the approved usage and limits.

Auto-Coding
The process of a machine generating source or object code using very high level language or model inputs into the machine.

Baseline
A formally controlled and maintained set of data that serves as the basis for defining change. When used as a verb, baseline is the act of initially establishing and approving a set of data.
[SAE EIA-649_1]

Black Box Testing
Testing that is performed on an item by applying certain inputs and looking for expected outputs without attempting to evaluate the internal design/functioning of the item to produce the outputs.

Build
A version of software that meets a specified subset of the requirements that the completed software will meet. [MIL-STD-498]

Change Impact Analysis
An analysis performed on a SPA when a modification is made to it. The analysis identifies the areas of the SPA that will be impacted by a modification, the areas of physical and functional coupling that could be impacted by the modification, and identifies any specific V&V regression activities that should be employed to provide confidence that unintended impacts have not occurred within unmodified portions of the system.

Code*
As applied to Computer Systems and Software: The implementation of particular data and/or computer instructions in the form of source code, object code, or machine code.

Coding Standard
A set of guidelines that are to be adhered to when writing code for a software development activity.

Component*
As applied to Computer Systems and Software: Any item that is used to construct an element.

Computer Software Component (CSC)
A functional or logically distinct part of a CSCI that implements a particular sub-function within the software. A CSC may consist of other CSCs or CSUs. [DOD-STD-2167A]

Computer Software Configuration Item (CSCI)
An aggregate of software that satisfies an end use function and is designated for purposes of specification, interfacing, qualification testing, CM or other purposes. A CSCI is composed of one or more software units which may consist of: (1) source code, object code, control code, control data or a collection of these items (2) an aggregation of software, such as a computer program or database, that satisfies an end use function and is designated for specification, qualification testing, interfacing, CM or other purposes, or (3) an identifiable part of a software product. A CSCI may also be interchangeably termed as a Software Configuration Item (SWCI). [SAE EIA-649_1]

Computer Software Unit (CSU)
See "Unit".

Computer System Integrity Level (CSIL)*
A designation applied to an element/component which determines the set of development and verification processes that will be applied to the element/component in order to achieve a defined level of integrity in the design.

Condition/Decision Coverage
See definition for "Structural Coverage Analysis"

Configuration Control Board (CCB)
A chartered board composed of technical and administrative representatives who recommend approval or disapproval of proposed engineering changes and variances to a CI's current approved and baselined configuration documentation. [SAE EIA-649_1]

Configuration Item (CI)
Any hardware, software, firmware or aggregation that satisfies an end use function and is designated by the Acquirer for separate configuration control. [SAE EIA-649_1]

Configuration Management (CM)
A technical and program management process applying appropriate resources, processes and tools to establish and maintain consistency between the product requirements, the product and associated product configuration information. [SAE EIA-649_1]

Configuration Status Accounting (CSA)
The configuration management function that formalizes the recording and reporting of the established product configuration information (including historical information), the status of proposed changes, and the implementation of approved changes and changes occurring to the product units due to operation and maintenance. CSA implementation includes assurance that the information is current, accurate, and retrievable. [SAE EIA-649_1]

Control Flow
The order that logic instructions, routines, and functions are processed within a computer program (or within a SPA). When analyzing control flow the control mechanisms that direct the order that things are executed need to be identified. These control mechanisms can be grouped into two different categories: Synchronous, which are control mechanisms encountered during the normal execution of the program logic (e.g., a call to a routine, an if-then-else statement) or Asynchronous, which are control mechanisms that are designed to abruptly divert the control from the normal execution of program logic (e.g., a priority interrupt).

Core Tests*
As applied to Computer Systems and Software: A subset of test cases of the entire test suite that is required to verify the Safety Critical Functions of a system for every flight release. Also known as Core Regression Tests.

Coupling*
As applied to Computer Systems and Software: The degree of interdependency between components/elements.

Critical Failure
A failure that impacts the operation of an SCF.

Criticality
A designation that indicates the degree that an item is considered to influence safe operation.

Data Channel*
As applied to Computer Systems and Software: The combination of transition medium(ia), protocols, and logical connections that is required to transfer data from a source to a destination.

Data Coupling
The degree of interdependency created between two or more software items as a result of the data transferred and utilized between the software items.

Data File
Machine readable data loaded into a computer that is accessed by executable code.

Data Flow
(1) For software: The logical path of movement, interfaces, and functional conversion that a given data item experiences as it is processed within a software program.

(2) For SPAs: The logical path of movement, interfaces, functional conversion, and hardware media that a given data item experiences as it is processed within a SPA.

Deactivated Code*
As applied to Computer Systems and Software: Any code that exists in an application that is designed to not be executed during intended operational conditions.

Dead Code*
As applied to Computer Systems and Software: Any code that exists in an application that is inaccessible during operation or whose outputs when executed are not utilized.

Depot Loadable Software
Software that can only be loaded onto the processing equipment at a depot maintenance environment, therefore the installation of the processing equipment on the air system is how the software is loaded on the air system.

Design Assurance Level (DAL)
See definition for "CSIL"

Dynamic Memory Allocation
The activity of a digital computer system allocating memory for a process or data structure during run time operations, which is initiated by the executing software logic.

Element*
As applied to Computer Systems and Software: A logically grouped collection of items having a definitive interface and functional purpose.

Endorsed AW SME-Level II
A subject matter expert that has been recognized by the USAF TAA as having the technical skills, education, and experience needed to oversee airworthiness assessments made by Endorsed AW SMEs-Level III and to render hazard risk assessments for MIL-HDBK-516 paragraphs found noncompliant. [AWB-1011A]

Endorsed AW SME-Level III
A subject matter expert that has been recognized by the USAF TAA as having the technical skills, education, and experience needed to provide a quality assessment of compliance to specified airworthiness criteria paragraphs within MIL-HDBK-516. [AWB-1011A]

Essential Functionality
Functionality necessary to ensure airworthiness safety.

Executable Code
System or application software that can be executed by a computer system. Executable files (also known as load images) are formed by linking object code and data files together with addressing for how the file should be loaded onto and accessed by the computer system.

Failure*
The inability of a system or system component to perform a required function within specified limits.

Failure Modes, Effects Analysis (FMEA)
A type of safety analysis that identifies failure modes and potential fault conditions in a system and the associated effects on the system.

Failure Modes, Effects, and Criticality Analysis (FMECA)*
A procedure for identifying potential failure modes in a system and classifying them according to their severity. A FMECA is usually carried out progressively in two parts. The first part identifies failure modes and their effects (also known as failure modes and effects analysis). The second part ranks the failure modes according to the combination of their severity and the probability of occurrence (criticality analysis).

Failure Modes, Effects Testing (FMET)
Testing derived from requirements, failure analyses, and lessons learned that are designed to verify and validate the robustness of the fault tolerance, failure management (FM), and redundancy management (RM) design.

Fault*
As applied to Computer Systems and Software: A manifestation of an error in software.

Fault Tolerance*
The ability of a system to provide an acceptable level of operational performance and safety in the event of one or more failures.

Firmware*
The combination of a hardware device and one or more of the following: computer instructions, computer data, and programmable hardware logic. The programming/data is not readily capable of being changed. NOTE: For the purposes of this document, the functionality of firmware defined by the programmable hardware logic is considered to be hardware; and functionality of firmware with computer instructions/data is treated as software.

Flight Critical*
A term applied to any condition, event, operation, process, or item whose proper recognition, control, performance, or tolerance is essential to achieving or maintaining controlled flight of an aircraft.

Flight Release
The act of releasing a specific configuration of software with authorization to use for flight operations.

Formal Qualification Test (FQT)
A verification activity that allows the contracting agency to determine whether a configuration item complies with the allocated requirements for that item. Requirements that pass during an FQT are considered to have been successfully met with regards to developmental and contractual obligations. The term “formal” refers to the type of witnessing that is associated with the official verification. Truly “formal” FQTs have the customer witness the testing and review the results, however some development activities do not contractually require a customer witness to be in attendance when the FQT is performed.

Frame
The amount of time designated within a real time digital processing system for processing a specified process or task.

Frame Rate
The number of frames executed by a real time digital processing system over a fixed period of time.

Full FQT
An FQT is considered to be full when all the requirements associated with the software configuration under test are completely tested.

Functional Baseline*
The approved configuration documentation describing a system's or top-level configuration item's performance (functional, interoperability, and interface characteristics) and the verification required to demonstrate the achievement of those specified characteristics.

Functional Coupling
The degree of interdependency that two or more items have with each other regarding the operation of a given function.

Functional Separation*
An attribute of a component/element that is achieved when its functional performance has no dependency on the functional performance of another component/element.

Functional Thread
The collection of elements/components within a system and the required interfacing and interaction of those elements/components that are necessary to achieve operation of a given function.

Hardware Configuration Item (HWCI)
An aggregation of hardware that satisfies an end use function and is designated for separate configuration management by the acquirer. [MIL-STD-498]

Hardware-Software Integration
Informal testing that ensures that the software architecture and design are compatible with the target hardware and that software/system performance requirements can be supported by that hardware.

Hazard*
A real or potential condition that could lead to an unplanned event or series of events (i.e. mishap(s)) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment. [MIL-STD-882]

Independent (also Independence)
Separation of responsibilities which ensures the accomplishment of objective evaluation.

(1) For software verification process activities, independence is achieved when the verification activity is performed by a person(s) other than the developer of the item being verified, and a tool(s) may be used to achieve equivalence to the human verification activity.

(2) For the software quality assurance process, independence also includes the authority to ensure corrective action. [DO-178C]

Integrity
An attribute of the system or an item indicating that it can be relied upon to work correctly on demand [DO-178C]

Interface
(1) In software development, a relationship among two or more entities (e.g., CSCI to CSCI, CSCI to HWCI, CSCI to a user, software unit to software unit) in which the entities share, provide, or exchange data. An interface is not a CSCI, software unit, or other system component; it is a relationship among them. [MIL-STD-498]

(2) The functional and physical characteristics at a common boundary. [SAE EIA-649_1]

Interlock*
System design mechanization to enable or disable systems, functions, subsystems, or modes at given times and conditions.

Interrupt
A suspension of a task, such as the execution of a computer program, caused by an event external to the task, and performed in such a way that the task can be resumed. [DO-178C]

Item
A term used to denote any product, including systems, materials, parts, subassemblies, sets, accessories, software items, etc. [SAE EIA-649_1]

Level of Rigor (LOR)
A specification of the depth and breadth of software analysis and verification activities necessary to provide a sufficient level of confidence that a safety-critical or safety-related software function will perform as required. [MIL-STD-882E]

Load Image
A machine readable file containing the programming logic and data that is loadable to a specific processing unit.

Memory Leak
Memory locations that are unavailable for allocation and use by a digital computer system because the memory management functionality of the system did not make the memory locations available for allocation and reuse after a deallocation event occurred for the memory locations.

Model Based Design (MBD)
A design method in which system functionality and operation is represented by models. Some MBD tools have the capability to generate code directly from the generated models.

Object Code
Machine readable code that is generated by a compiler after processing source code. To become executable the software typically must be linked with other object code files to form the executable code file that can be loaded into and used by the computer system hardware.

Partitioning*
As applied to Computer Systems and Software: A technique for providing system resource (e.g., memory, throughput) isolation to a given piece of software.

Peer Review
The activity within a software development activity of taking a specific product produced by a developer and have the product reviewed by independent peers to ensure quality and correctness of the product. Products typically peer reviewed in a high quality software development activity include requirements, software design, hand-coded source code, and test plans & procedures.

Performance*
A quantitative or qualitative measure characterizing a physical or functional attribute relating to the execution of an operation or function. Performance attributes include quantity (how many or how much), quality (how well), coverage (how much area, how far), timeliness (how responsive, how frequent), and readiness (availability, mission/operational readiness). Performance is an attribute for all systems, people, products, and processes including those for development, production, verification, deployment, operations, support, training, and disposal. Supportability parameters, manufacturing process variability, reliability, and so forth are all performance measures.

Physical Separation*
As applied to Computer Systems and Software: A hardware item is considered to be physically separated from another hardware item when the two items are electrically isolated from each other. A software application (e.g., CSCI, load image) is considered to be physically separated from another software application when the two applications do not share throughput or memory resources (resource isolation can be achieved through a partitioning mechanism).

Primary Functional Path
The end-to-end collection of equipment that is primarily relied upon to provide a given function of operation.

Probability of Loss Of Control (PLOC)*
An expression of the likelihood of the loss of control, based upon the failure probabilities of those elements contributing to control of the air vehicle system.

Product Baseline*

The approved technical documentation which describes the configuration of a CI during the production, fielding/deployment and operational support phases of its life cycle. The product baseline prescribes all necessary physical or form, fit, and function characteristics of a CI, the selected functional characteristics designated for production acceptance testing, and the production acceptance test requirements (MIL-HDBK-61). When used for re-procurement of a CI, the product baseline documentation also includes the allocated configuration documentation to ensure that performance requirements are not compromised.

Process

(1) Related to general development activity: An organized set of activities performed for a given purpose. [MIL-STD-498]

(2) Related to operating software: A set of logic instructions that are interrelated and interact on inputs to produce controlled outputs.

Process Attribute

A characteristic quality or feature associated with a specific process.

Product Attribute

A characteristic quality or feature associated with a specific product.

Qualified*

The resulting state or condition from a formal verification process being applied to hardware and/or software systems to verify that requirements have successfully been met.

Redundancy*

Utilization of two or more components, subsystems, or channels so that the functions that they support are capable of being sustained in the event of a failure.

Redundancy Management*

The process of managing redundant elements in order to identify a failure, and then reconfiguring the system to remove or mitigate the effects of the failed element and continue operation with operating elements.

Regression Test

The testing to confirm that unmodified functions of a system, that were previously performing correctly, continue to perform correctly after a change has been made to the system.

Release (software)

The act of releasing a specific configuration of software with authorization to use it in specifically approved ways

Reportable

Any permanent or temporary configuration change or alteration to an item, change in capability, change to the service life, or change in mission usage that has a potentially significant impact on airworthiness. Per AWB-007, airworthiness oversight and certification approval comes from the independent Technical Airworthiness Authority.

Requirement

(1) A characteristic that a system, element or configuration item must possess in order to be acceptable to the acquirer.

(2) A mandatory statement in a standard or contract.

Robustness Testing*

Testing that stresses software by running it under abnormal or off-nominal conditions (e.g., input values, timing) or very near the performance limits of the system under nominal conditions.

Safety Critical*

A safety classification given to any condition, event, operation, process, or item whose proper recognition, control, performance, or tolerance is essential to safe system operation and support (e.g., safety critical function, safety critical path, safety critical software, or safety critical component). Safety critical is a broader definition of the categorizations of safety and includes flight critical, but may not be limited to controlling flight. The term safety critical, as defined in MIL-STD-882, is "a condition, event, operation, process, or item whose mishap severity consequence is either Catastrophic or Critical".

Safety Critical Function (SCF)*

A function whose failure to operate or incorrect operation will directly result in a mishap of either Catastrophic or Critical severity. [MIL-STD-882]

Safety Critical Function (SCF) Thread*

The combination of elements/components within a system and the required interfacing and interaction of those elements/components whose overall contribution is necessary for the operation of a given Safety Critical Function.

Safety Critical Function Thread Analysis (SCFTA)

An analysis of the thread of equipment supporting an SCF and the associated V&V activities applied to the thread. An SCFTA is intended to ensure that all equipment supporting the SCF is identified, the interfacing and integration of that equipment is understood, and that the end-to-end coverage of V&V is sufficient to provide the safety assurance desired.

Safety Interlocks*

An interlock that is necessary for the operation of one or more Safety Critical Functions.

Safety Supporting Element (SSE)*

A single instance, logical grouping, or combination of SSSEs and SSHEs that is necessary for the operation of an SCF. An SSE can be a combination of SSEs.

Safety Supporting Hardware Element (SSHE)*

An element, comprised only of computer hardware, which is necessary for the operation of an SCF.

Safety Supporting Software Element (SSSE)*
An element, comprised only of software, which is necessary for the operation of an SCF.

SCF Data Receiver
An item identified in an SCFTA that is an end point for receiving data, commands, or signals output by the processing functionality associated with an SCF.

SCF Data Source
An item identified in an SCFTA that is the origination point for data, commands, or signals input into the processing functionality associated with an SCF.

Secondary Functional Path
The end-to-end collection of equipment that is relied upon to provide a given function of operation when there is a failure in the primary functional path.

Single Event Upset (SEU)*
The resulting unintentional change in the state of a binary logic storage cell (i.e., changing from 0 to 1 or vice versa) that occurs when an ionizing particle or electro-magnetic radiation strikes the storage circuitry. The erroneous state change is not permanent and is considered a soft failure because resetting or rewriting to memory will clear the condition.

Software
A combination of associated computer instructions and computer data definitions required to enable the computer hardware to perform computational or control functions. [DOD-STD-2167A]

Software Architecture*
The organizational structure and interrelationship of software that identifies its components, interfaces, and the control/flow of execution.

Software Development File (SDF)
A repository for material pertinent to the development of a particular body of software. Contents typically include (either directly or by reference) considerations, rationale, and constraints related to requirements analysis, design, and implementation; developer-internal test information; and schedule and status information. Also known as Software Development Folders.

Software Development Plan (SDP)
The collection of documentation that establishes the development plans and overarching processes that are to be applied to a given software project. The SDP defines how the software project will be managed and identifies the safety oriented development processes that apply to SSSE development and V&V.

Software Levels
A term utilized in the DO-178, *Software Considerations in Airborne Systems and Equipment Certification,* which is defined as: The designation that is assigned to a software item as determined by the system safety assessment process. The software level establishes the rigor necessary to demonstrate compliance with DO-178. DO-178 establishes 5 Software Levels from level A to E with level A being the most stringent of the levels. [DO-178C]

Software Review Board
A CCB for CSCIs.

Source Code
Code written in source languages, such as assembly language or high level languages, that becomes an input into an assembler or a compiler or interpreter so that the code and be executed.

Stakeholder

An individual or organization having a right, share, claim or interest in a system or in its possession of characteristics that meet their needs and expectations. [IEEE 12207]

Statement Coverage
See definition for "Structural Coverage Analysis" below

Structural Coverage Analysis
An analysis of code structure and interfaces, which are evaluated during testing of the code. There are numerous types of structural coverage analysis that can be performed on software code. Those referenced in this document are listed below:

> **Condition/Decision Coverage**
> Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken on all possible outcomes at least once, and every decision in the program has taken on all possible outcomes at least once. [DO-178B]
>
> **Statement Coverage**
> Every statement in the program has been invoked at least once [DO-178C]

System
An aggregation of elements that achieve a given purpose or provide a capability.

System Processing Architecture (SPA)*
A collection of processing elements, and the structure and interconnections of those elements, which form systems or subsystems to meet processing requirements.

System Safety*
The application of engineering and management principles, criteria, and techniques to achieve acceptable risk within the constraints of operational effectiveness and suitability, time, and cost throughout all phases of the system life cycle. [MIL-STD-882]

Target Hardware
The processing hardware that software is intended to run on in the operational or fielded environment.

Throughput

In general, throughput refers to the amount of actual time utilized to execute a set of tasks over a given period of time. For computer processors, it is the amount of time spent executing a set of instructions over a given period of time. For data channels, it is the time utilized to transfer a set of data products (e.g., bits, words) over the data channel in a given period of time. For both computer processors and data channels there is a maximum amount of throughput that can be achieved by a given design. Throughput measures are typically expressed in a percent utilization of the maximum throughput that a device is designed to provide.

Traceability*

The ability to take a particular piece of development or design information and trace it to related information (examples: System level requirements to subsystem level requirements, software requirements to software design components, hardware component requirements to hardware test cases, function to sub-function, sub-function to hardware and software components that support the sub-function). Bi-directional traceability is the ability to make the trace in both directions.

Training System Hooks

Mechanizations implemented within air vehicle loadable software that permit the system to operate as, and/or integrate with, training system equipment.

Unit (software)

The smallest logical entity specified in the detailed design that completely describes a single function in sufficient detail to allow implementing code to be produced and tested independently of other units. Units are the lowest level decomposition of software design and the actual physical entities implemented in code. Also called Computer Software Unit (CSU). [DOD-STD-2167A]

Unmanned Aircraft System (UAS)*

A UAS is comprised of individual elements consisting of the unmanned air vehicle (UAV), the control station, and any other support elements necessary to enable operation including, but not limited to data links, communications systems/links, and UAV-unique launch and recovery equipment. There may be multiple unmanned aircraft, control stations, and support elements within a UAS. The control station may be located on the ground (stationary or mobile), on a ship, submarine, aircraft, etc.

Validation

Confirmation that an implemented function or design product is suitable for its intended use.

Verification

Confirmation established through objective, empirical evidence that specified requirements have been fulfilled [IEEE 12207]

V&V Coverage

The scope of integrity assurance that is provided by a given set of verification and validation activities.

White Box Testing

Testing that is performed on an item by applying certain inputs and looking for expected outputs while also evaluating the internal design or function of the item to produce the outputs. Also known as clear box testing.

Worst Case Loading

The functional conditions that stress a processing system into its heaviest resource utilization (e.g., processor throughput, I/O throughput, memory utilization)

End of Glossary of Terms