JOINT SERVICES – SOFTWARE SAFETY AUTHORITIES (JS-SSA)

Software System Safety

Implementation Process and Tasks Supporting MIL-STD-882E

With Joint Software System Safety Engineering Handbook References

JS-SSA-IG Rev. B

3/14/2018

This report is provided in fulfillment of the JS-SSA and provides implementation guidance for Software System Safety program requirements that comply with the requirements specified in MIL-STD-882E and guidance detailed in the Joint Software Systems Safety Engineering Handbook (JSSSEH).

Enclosure (2)

Table of Contents

List of Figures

Appendices

Revision History

Revision	Date	Description
-	29 June 2016	Initial Release
A	17 October 2017	Updated to incorporate annual JS-SSA member comments. Text and figures updated to create tighter coupling between IG and MIL-STD-882E Tasks
B	14 March 2018	Out of cycle revision to incorporate JS-SSA approved comments and recommendations resulting from U.S. Navy initiated Carnegie Mellon (CM) Software Engineering Institute (SEI) review of Rev. A. Comments resulted in revising Section 3.9 Perform Code-Level Safety Analysis, as well as CM SEI recommendations to clarify importance of application within the software development life-cycle and processes. Technical content and data structure is unchanged. Corrects identified administrative and clerical errors in text and figures.

1.0 Software System Safety Abstract

Software System Safety Engineering (SSSE) focuses on two primary objectives; first to design, code, test, and support software with the appropriate Level-of-Rigor (LOR) to instill a confidence, or the assurance of safe software; and to define the necessary safety requirements for the design, code, test, verification, and validation of software that specifically target and mitigate the software "causes" of the defined hazards and mishaps of the system. Each of these two objectives is covered in detail within the Joint Software Systems Safety Engineering Handbook (JSSSEH). Requirements to meet the SSSE objectives are specified in MIL-STD-882E. The challenge is getting Acquirers (Customer) and Developers (software developers) to specify how they will turn the objectives of MIL-STD-882E and the JSSSEH "guidance" into actual SSSE requirements. The objective of this document is to provide Department of Defense (DoD) Acquirers and their Developers with the implementation details necessary to take the requirements of MIL-STD-882E and the "guidance" of the JSSSEH and define the process and tasks required for a compliant SSSE program. MIL-STD-882E and guidance of the JSSSEH will continue to be the parent source for guidance on how to meet identified software safety engineering requirements. This document is also appropriate for use by non-DoD entities developing software for safety-significant systems.

This revision of the originally released version of this Implementation Guide focuses on correcting known errors and inconsistencies in the document, create greater alignment between the document and the parent MIL-STD-882E requirements and tasks, and incorporate results and lessons learned from the successful initial release of the document. This revision maintains the core purpose and structure of the original version, which is the process and tasks to implement and execute a MIL-STD-882E compliant SSSE program as part of System Safety Engineering (SSE) and integrated within the overall software development processes.

2.0 Specialty Task Outline and Process

This distillation of MIL-STD-882E and the JSSSEH into implementable process task requirements is presented as a decomposition of parent and children activities in a process task numbering format. The parent tasks are graphically represented depicting inputs to the tasks and the products that the task would likely produce. Tasks identified as MIL-STD-882 requirements are coded in the graphics using an extreme bold border of the task box. Task decomposition is to the level necessary for a basic understanding of the process, the tasks that implement the process, and the products the tasks would likely produce. The requirements derived that apply to each task will be specified and cross referenced to both the applicable MIL-STD-882E requirements and JSSSEH sections and paragraphs that provide guidance on meeting the requirements. As such, any DoD Acquirer or Developer should be able to develop SSSE tasks and requirements that comply with MIL-STD-882E and the guidance of the JSSSEH. Appendix A of this document is a LOR task table that should be used to develop the defined process tasks necessary to meet MIL-STD-882E Table V LOR requirements. The LOR task table also supports accomplishment of the MIL-STD-882E Tasks for the SSSE contribution to SSE. The LOR table should be assessed for tailored implementation for any given program, and tailoring is permitted as long as the tailored LOR tasks are approved by both the Acquirer and Developer.

3.0 Process and Process Tasks for Software System Safety (SSS)

The process for accomplishing a successful SSS program begins with the contract between the Acquirer (typically a DoD Agency, Program, Project or Product Office) and the Developer (generally referred to as the Developer or Software Developer). It is essential for the DoD Acquirer to adequately specify the SSE and SSSE tasks and artifacts necessary to meet the requirements of MIL-STD-882E on contract. If the Statement of Work (SOW) does not define the required Developer safety tasks and artifacts, then the overall safety program is likely to not meet either the service-specific or the Joint Services safety requirements. The vast majority of SOW tasks are reflected in Appendix A under SSE Tasks required for adequately supporting the software system safety effort. Additionally, the SOW must specify the frequency of meetings; Contract Deliverable Requirements List (CDRLs) items; and necessary reviews in order for the developer to adequately bid their efforts.

The Acquirer must adequately plan for the tasks that will be required and implemented by the Developer. This planning is accomplished prior to the Request for Proposal (RFP) (or contract change for existing programs) and documented in the System Safety Management Plan (SSMP) as referenced in JSSSEH Para 4.2.1 and detailed in Section 3.1 Process Task 1.0. Specific Acquirer tasks that must be accomplished prior to contract award include (but are not limited to);

- Develop SSMP.
- Utilize the Mishap Risk Matrix, Software Criticality Matrix (SCM) and associated input definitions of MIL-STD-882E, unless a tailored version has been approved in accordance with (IAW) DoD Component Policy.
- Charter the System Safety Working Group (SSWG) to include all managerial, organizational, and technical relationships.

- Develop Safety input to the RFP, SOW and other contractual documentation (this is where Tasks, CDRLs, and required analyses, etc. should be specified, as well as when/where delivered/documented). Required analyses should include: MIL-STD-882E Task 102 System Safety Program Plan (SSPP), Task 106 Hazard Tracking System (HTS), Task 201/202 Preliminary Hazard List (PHL)/ Preliminary Hazard Analysis (PHA), Task 203 System Requirements Hazard Analysis (SRHA), Task 204 Subsystem Hazard Analysis (SSHA), Task 205 System Hazard Analysis (SHA), Task 206 Operating and Support Hazard Analysis (O&SHA), Task 208 Functional Hazard Analysis (FHA), and Task 301 Safety Assessment Report (SAR). As applicable, Task 209 System-of-Systems (SoS) Hazard Analysis may be required.
- Define Acquirer specification safety requirements.
- Provide safety requirements input to other relevant documentation (e.g., Software Development Plan (SDP), Test and Evaluation Master Plan (TEMP), System Engineering Master Plan (SEMP), and Configuration Management Plan (CMP)).
- Work with the Program/ Project/ Product Manager (PM) to ensure the system safety and software system safety program is adequately resourced and staffed.
- Ensure Acquirer Safety is part of the configuration control process (voting member of Acquirer chaired boards, participant/reviewer of safety impacted items at the level (MIL-STD-882E Task 304). Evidence can be incorporated into the CMP and/or Board Charter(s).
- Perform analyses required to define the System Level Mishaps and interfaces/contributions provided by supporting system elements (e.g., multiple Developers may be developing different critical subsystems and each must account for their respective contributions to system mishaps).

Figure 1.0 includes the initial process tasks required by the Acquirer and then transitioning to the tasks required by the Developer after contract award. This document will step through the parent process tasks beginning with Task 1.0: Prepare System Safety Management Plan. Subtasks (children to the parent task) will also be presented and discussed to ensure that the reader fully comprehends the scope and details of each major task described. Where applicable, references to MIL-STD-882E and the JSSSEH are provided for further detail.

The SSSE process and tasks described in this document can be applied within an integrated system and software development process regardless of software development technique. Current software development best practices have shifted from the classical, aligned with the overall system development schedule, waterfall process towards more rapid (e.g., Agile), model-based software development processes. Current processes use varying terminology to describe their tasks and life-cycle phases. However, at their core, all software development processes follow the “Establish the Program, Requirements, Design and Architecture, Implementation, and Test and Verification” process steps. The primary “difference” is that within the rapid, model-based software development processes, the development steps are applied to smaller subsets of overall software functionality and capabilities during a given development cycle and the process occurs within a much smaller timeframe than the overall acquisition schedule. For example, the “Requirements Phase” may focus on taking the system engineering domain architecture artifacts and developing Use Cases to implement the “Requirements and Capabilities” specified by the architecture artifacts as opposed to a natural language Software

Requirements Specification (SRS). The tailoring of the LOR Tasks specified in Appendix A becomes even more important to accomplish at program outset, to include updating of terminology to match the software development, and subsequent integration within the software development process activities. SSSE must be integrated within the software development team, as well, to both be able to provide SSSE analysis in support of the software development effort and to ensure the development team is meeting their SSSE LOR task requirements.

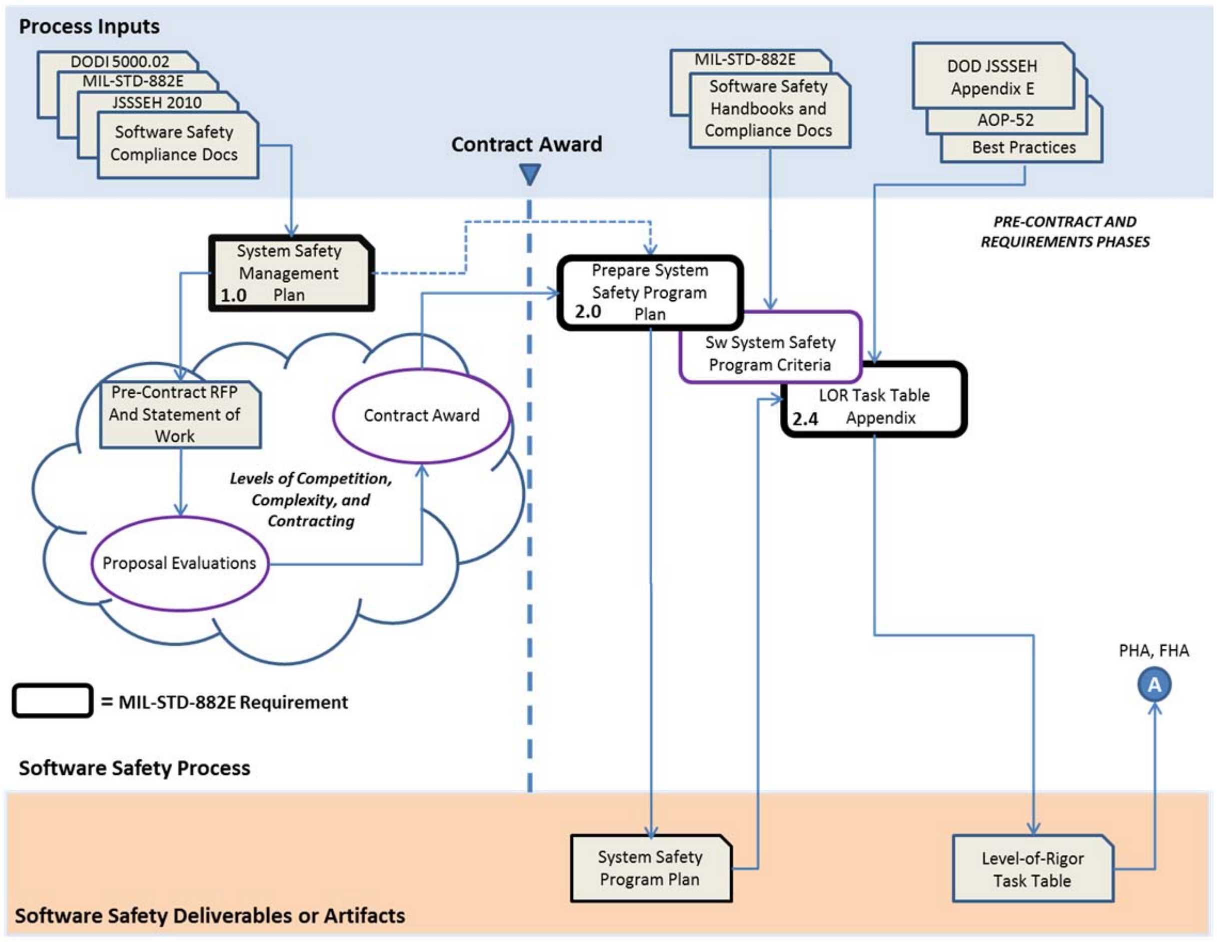

Process Inputs
DODI 5000.02
MIL-STD-882E
JSSSEH 2010
Software Safety Compliance Docs
Contract Award
MIL-STD-882E
Software Safety Handbooks and Compliance Docs
DOD JSSSEH Appendix E
AOP-52
Best Practices
PRE-CONTRACT AND REQUIREMENTS PHASES
System Safety Management 1.0 Plan
Prepare System Safety Program 2.0 Plan
Sw System Safety Program Criteria
LOR Task Table 2.4 Appendix
Pre-Contract RFP And Statement of Work
Contract Award
Levels of Competition, Complexity, and Contracting
Proposal Evaluations
= MIL-STD-882E Requirement
PHA, FHA
A
Software Safety Process
System Safety Program Plan
Level-of-Rigor Task Table
Software Safety Deliverables or Artifacts

Figure 1.0: Initial SSS Process Chart for Pre-Contract and Requirements Phases

3.1. Process Task 1.0: Prepare the SSMP

It is standard practice within an Acquirer's program office to develop a SSMP for a program (or family of programs). This document defines the Acquirer's requirements for the establishment, structure, resourcing, and implementation of a system safety and software system safety activity for the program, or family of programs, to be managed by an individual Acquirer program office. The SSMP must document the system safety program requirements as established by applicable Federal and Civil law, and DoD acquisition authorities. Subtasks to this process task are depicted in Figure 1.1.

3.1.1. Reference Documents

The following documents provide the basis for the format and criterion for the SSMP and this implementation guide:

- DoD Instruction 5000.02, Change 3 – Operation of the Defense Acquisition System, August 10, 2017.
- MIL-STD-882E – Department of Defense Standard Practice, System Safety, May 11, 2012.
- DoD Joint Software System Safety Engineering Handbook, Version 1.0, August 27, 2010.
- RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification, 2011.
- SAE Aerospace Recommended Practice (ARP) 4754, Certification Considerations for Highly-Integrated or Complex Aircraft Systems, November 1, 1996.
- SAE Aerospace Recommended Practice (ARP) 4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, 1996.

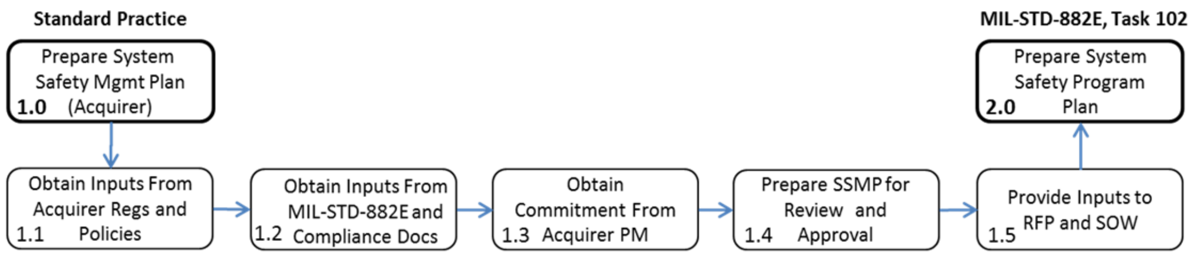

Standard Practice
Prepare System Safety Mgmt Plan 1.0 (Acquirer)
MIL-STD-882E, Task 102
Prepare System Safety Program Plan 2.0
Obtain Inputs From Acquirer Regs and Policies 1.1
Obtain Inputs From MIL-STD-882E and Compliance Docs 1.2
Obtain Commitment From Acquirer PM 1.3
Prepare SSMP for Review and Approval 1.4
Provide Inputs to RFP and SOW 1.5

Figure 1.1: Process Task 1.0 Prepare the SSMP Subtasks

3.1.2. Process Subtask 1.1: Obtain Inputs from Acquirer Regulations and Policies

The requirements for a system safety program are explicitly documented in MIL-STD-882E. The Acquirer's specific regulations and policies provide the requirements for generation of the SSMP. The SSMP must reflect the criterion established in these regulations and policies as well as in MIL-STD-882E. Examples may include such things as Air Worthiness Certification criteria for air vehicles to the requirements established by individual safety boards (i.e., regulatory, Service and Joint Safety Review Boards). The SSMP must include these additional safety program requirements that will aid the Acquirer in meeting all certification or acceptance authority criteria. The SSMP should ensure that all of the Acquirer's system safety and software safety requirements are adequately transitioned to the RFP and the SOW, and ultimately flowed down to the Developer. This flow down of the SSMP ensures the Developer's System Safety Program Plan (SSPP) supports the Acquirer meeting their safety requirements. Developers bidding on the contract must have a clear understanding of the Acquirer's expectations for the system and software safety engineering efforts. This allows the Developer to bid and propose a system safety program based upon Acquirer requirements and expectations.

3.1.3. Process Subtask 1.2: Obtain Inputs from MIL-STD-882E and Compliance Documents

MIL-STD-882E, Task 101 provides direction on content for system safety program management. Individual Acquirer programs may possess system safety requirements that are not explicitly covered in

MIL-STD-882E, regulations or policies, but deemed important by the Program Office. The development of a SSMP must take into consideration the requirements that are defined by other compliance documents.

3.1.4. Process Subtask 1.3: Obtain Commitment from Program Management

Without Program Management "buy-in" regarding the necessity and Return-On-Investment (ROI) of a comprehensive system safety engineering effort, the probability of successfully influencing the design from a safety perspective is greatly reduced. The system safety program defined by the SSMP must possess concurrence and acceptance from the PM. This is the best opportunity for the System Safety Manager to adequately communicate the system safety ROI to Program Management in terms of the total life-cycle costs associated with mishaps as it affects human injury or death, programmatic costs, schedule, operational readiness, operation effectiveness, and organizational reputation.

3.1.5. Process Subtask 1.4: Prepare SSMP for Review and Approval

Ultimately, it is the Acquirer who must meet their respective program requirements. The SSMP defines the path forward for all system safety efforts to be performed by both the Acquirer and the Developer(s). This plan establishes the overall system safety requirements whereas the Developer's SSPP defines the processes, methods, tasks, and tools to be implemented to meet the SSMP and contracted safety requirements. The SSMP must also include all applicable requirements for the establishment and implementation of a software system safety program.

3.1.6. Process Subtask 1.5: Provide Inputs to the RFP and SOW

Once the SSMP is produced and approved by Acquirer Management, the system safety requirements language defined in the plan must be captured in each RFP and SOW that are published by the program office to support the design, development, and test, of each program asset being developed or updated. If the system safety requirements are not captured in the RFP and the supporting SOW, the Developers will possess no contractual basis to perform the necessary tasks to complete a successful system safety or software system safety program.

3.2. Process Task 2.0: Prepare SSPP

[Ref: JSSSEH Paragraph 4.2.1, Fig. 4-6, and MIL-STD-882E Task 102]

The SSPP is the document of compliance for the contract as it applies to system safety and software system safety engineering. Figure 1.2 depicts the process subtasks as they apply to the task of preparing the SSPP for approval. The Developer's SSPP, including the Software System Safety Program Plan (SwSSPP) requirements, must define how the Developer's contractual safety requirements are flowed down, implemented and verified by their development team, sub-developers, subcontractors, or vendors.

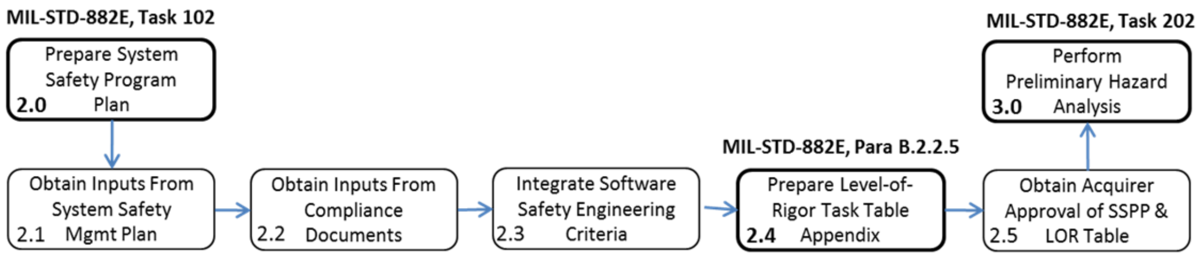

MIL-STD-882E, Task 102
Prepare System Safety Program Plan
2.0
MIL-STD-882E, Task 202
Perform Preliminary Hazard Analysis
3.0
MIL-STD-882E, Para B.2.2.5
Obtain Inputs From System Safety Mgmt Plan
2.1
Obtain Inputs From Compliance Documents
2.2
Integrate Software Safety Engineering Criteria
2.3
Prepare Level-of-Rigor Task Table Appendix
2.4
Obtain Acquirer Approval of SSPP & LOR Table
2.5

Figure 1.2: Process Task 2.0 SSPP Task and Subtasks

3.2.1. Process Subtask 2.1: Obtain Inputs from the SSMP

The SSMP (or equivalent Acquirer document) defines the relevant compliance criteria from standards, regulations, and handbooks, and defines the terms and term definitions to be used on the program and to charter the SSWG. From the SSMP and Contract tasks, the Developer prepares the SSPP that defines and documents the processes, tasks, and deliverables to be accomplished on the program to comply with the contractual safety requirements. The SSPP should contain, as a minimum, the information defined by MIL-STD-882E, Task 102 and the corresponding Data Item Description (DID), DI-SAFT-80100A, System Safety Program Plan. If the JSSSEH is cited in the SOW, SSMP, and/or SSPP, then this JSSSEH Implementation document should be used to develop Software System Safety Program (SwSSP) requirements.

3.2.2. Process Subtask 2.2: Obtain Inputs from Compliance Documents

The primary compliance document is MIL-STD-882. Depending upon whether this is a new acquisition program or a fielded system in the Sustainment phase (such as a legacy program), the contractual version of MIL-STD-882 may be Revision E or an earlier version. In addition, each DoD Service may have separate compliance documents for software system safety. On aviation related contracts, aviation specific compliance documents or standards may be required. This is important to understand because each individual standard can use terms that are common to the safety community but possess totally different meanings. As an example, the FHA as defined by SAE ARP 4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, has a different purpose, content, and format as an FHA as defined by a MIL-STD-882E.

3.2.3. Process Subtask 2.3: Integrate Software Safety Engineering Criteria

The specific processes, tasks, and deliverables to support SSS engineering should be completely integrated into the main text of the SSPP. IAW with MIL-STD-882E requirements, the SSPP must detail how the SSS requirements are going to be addressed for all safety-significant software (as software is defined in MIL-STD-882E) within the system. The SSS input to the SSPP must address the requirements for each development phase and also address validation and approval of tools, models, and simulations that will be used in the development, support, and verification/validation of safety-significant software. The SSS engineering criteria to be specified in the SSPP, and implemented, should be extracted from both MIL-STD-882E and the JSSSEH.

3.2.4. Process Subtask 2.4: Prepare LOR Appendix

[Ref: JSSSEH Paragraph 4.2.1.5, and 4.3.2, JSSSEH Figure 4-13, Table 4-3, and Table 4-4; and Appendix A-Level of Rigor Task Table]

The SSPP must contain the comprehensive LOR Task Table that establishes both the Developer's process and design requirements for SSS. The Acquirer, either via reference to the SSPP, or inclusion in the SSMP must specify an initial LOR Task Table for the Developer to tailor and implement as part of their SSS and software development programs. The LOR Table contains the specific required process tasks and design requirements to obtain the necessary body of evidence that software introduces an acceptable contribution to mishap risks for the program, as well as to implement and meet the intent of the general requirements of MIL-STD-882. This confidence can only be obtained if the defined tasks for each software development life cycle phase are successfully defined and accomplished. The subtasks of Process 2.0 are presented in Figure 1.3.

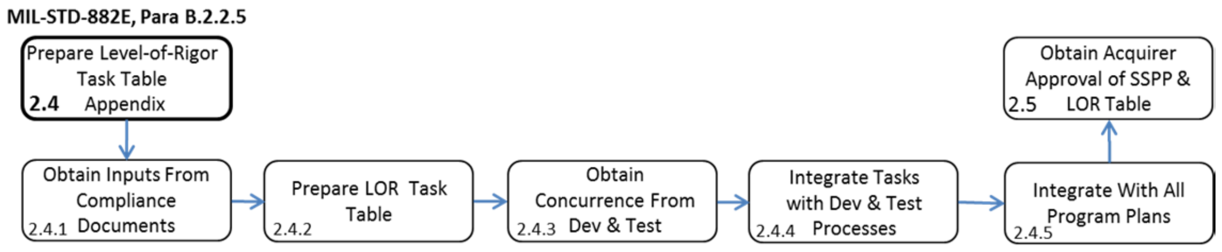

MIL-STD-882E, Para B.2.2.5
Prepare Level-of-Rigor Task Table 2.4 Appendix
Obtain Inputs From Compliance 2.4.1 Documents
Prepare LOR Task Table 2.4.2
Obtain Concurrence From 2.4.3 Dev & Test
Integrate Tasks with Dev & Test 2.4.4 Processes
Integrate With All Program Plans 2.4.5
Obtain Acquirer Approval of SSPP & 2.5 LOR Table

Figure 1.3: Sub-Process Task 2.4 Prepare LOR Appendix to SSPP

3.2.4.1. Process Subtask 2.4.1: Obtain LOR Task Inputs from Compliance Documents

Preparation of a LOR Table that will be accepted by the Developers' development team and approved by the Acquirer and their independent safety reviews, necessitates that compliance documents become the foundational input to the table. For example, in MIL-STD-882E, Table V defines the general LOR requirements for each level of Software Criticality. The purpose of Process Subtask 2.4 is to take those general requirements and then specify the specific implementation requirements the program will execute to fulfill the MIL-STD-882E criteria. DoD Service requirements must also be assessed for inclusion into the table, as well as the requirements of the various Service and Joint Safety Reviews.

For aviation-related programs, SAE ARP 4754/4761 and RTCA DO-178C, Software Considerations in Airborne Systems and Equipment Certification, are called out for the purposes of ensuring that airworthiness requirements are established and fulfilled to obtain an Air Worthiness Release (AWR). In addition, each of the Services, either jointly or independently, may have unique LOR requirements for specific subsystems that require separate approvals, such as fuze, ignition and laser subsystems.

3.2.4.2. Process Subtask 2.4.2: Prepare the LOR Task Table Appendix

An initial LOR task table is provided in Appendix A. An initial LOR task table can also be included in the Acquirer's SSMP and provided separately by the Acquirer to the Developer. This table can be tailored for each individual program's capabilities and requirements based upon the criticality, complexity, and constraints of the program. All tailoring must be approved by the SSWG and the Acquirer. It is highly recommended that the Acquirer obtain concurrence of tailored tasks from the various applicable review authorities as well. Where tailoring is implemented, it must be adequately explained, justified and approved.

Within the LOR Table, the criticality is ranked in accordance with both the mishap severity definitions and the software control category definitions of MIL-STD-882E. LOR ranking is from lowest criticality Software Criticality Index (SwCI 5) to highest criticality (SwCI 1) as referenced in MIL-STD-882E Table V.

System Functions that are deemed Safety-Significant Function (SSF) IAW MIL-STD-882E are assessed for their criticality (see Sections 3.5.2, 3.5.3, and 3.5.4) and are architected, designed, coded, and verified in accordance with the LOR tasks documented in the program's approved LOR Table. LOR tasks must include the software development process tasks, software test and verification tasks, and design assurance features that are required for each SSF.

In reality, budgetary and schedule constraints may play a role in tailoring requirements. However, if the budget and schedule do influence the tailoring process, it will likely produce more safety risk for the system, or at minimum impact acceptability by review authorities. Table VI of MIL-STD-882E provides requirements for notifying management and risk acceptance authorities of the consequences of lack of LOR application. When potential programmatic or safety risks resulting from budget and schedule impacts are identified, they may also require inclusion in the overall Program Risk tracking system to maintain management visibility. Risk acceptance performed in one contractual activity should be reassessed for the next contractual activity.

3.2.4.3. Process Subtask 2.4.3: Obtain LOR Concurrence from Development and Test

The software developers, architects, and testers must be integrated into the software SSE activities and be involved with the definition and implementation of LOR tasks. This is an important step in that the safety team must understand the system and software architecture to define the software criticality and the derived LOR tasks. In addition, the software developers and testers must fully comprehend their role in the execution of a successful software system safety (SSS) program. They must understand what they are required to accomplish by LOR definition and safety assurance rationale of the tasks to be accomplished. The software developers and testers should have an input to the definition and tailoring of LOR tasks. Any task that is perceived to possess little ROI for the resources expended should be flagged as candidates for tailoring with appropriate justification to include the potential safety risk rationale.

3.2.4.4. Process Subtask 2.4.4: Integrate LOR Tasks with Development and Test Processes

The approved LOR tasks must be fully integrated into the software development, coding, and test activities of the Developer. Some of the LOR tasks are accomplished by the system safety team, whereas many are actually accomplished by the software design, code, and test teams. Those tasks assigned to software development and test must be part of their defined processes as documented in their planning documents.

3.2.4.5. Process Subtask 2.4.5: Integrate LOR Tasks into Pertinent Program Plans

The approved LOR Table tasks must be adequately documented in the applicable Developer specifications, plans, and process documentation as requirements. Documents typically include: Software Requirements Specification (SRS), SDP, Software Test Plan (STP), Software Configuration Management Plan (SCMP), and Software Quality Assurance Plan (SQAP). The Developer must ultimately be able to answer the questions "How did you meet LOR requirements?" and, "Where is the evidence?"

3.2.5. Process Subtask 2.5: Obtain Acquirer Approval of the Developer's SSPP

The Developer's SSPP must be submitted for review and approval by the Acquirer to ensure it adequately addresses SSE and SSS requirements and tasks, to include references to other Developer documentation that may implement SSS requirements. This review and approval includes the approval of the LOR Appendix. Acquirer approval of the LOR Table, including concurrence from their safety review authorities, represents Acquirer concurrence that the tasks defined in the table are sufficient (if implemented and evidenced) to provide the necessary assurance that safety-significant software is being designed, coded, and tested in accordance with defined best practices. All LOR tailoring must be explained and justified. As the program matures, changes to the LOR table should be coordinated with and approved by the Acquirer (often via the SSWG) and Developer, as well as the applicable review authorities.

3.3. Process Task 3.0: Preliminary Hazard Analysis

[Ref: JSSSEH Paragraph 4.3.4 and MIL-STD-882E, Tasks 201 and 202]

The software system safety process as initiated in Figure 1.0 continues as depicted in Figure 2.0 below. Within the software development life cycle, this is considered the requirements and preliminary design phase of development.

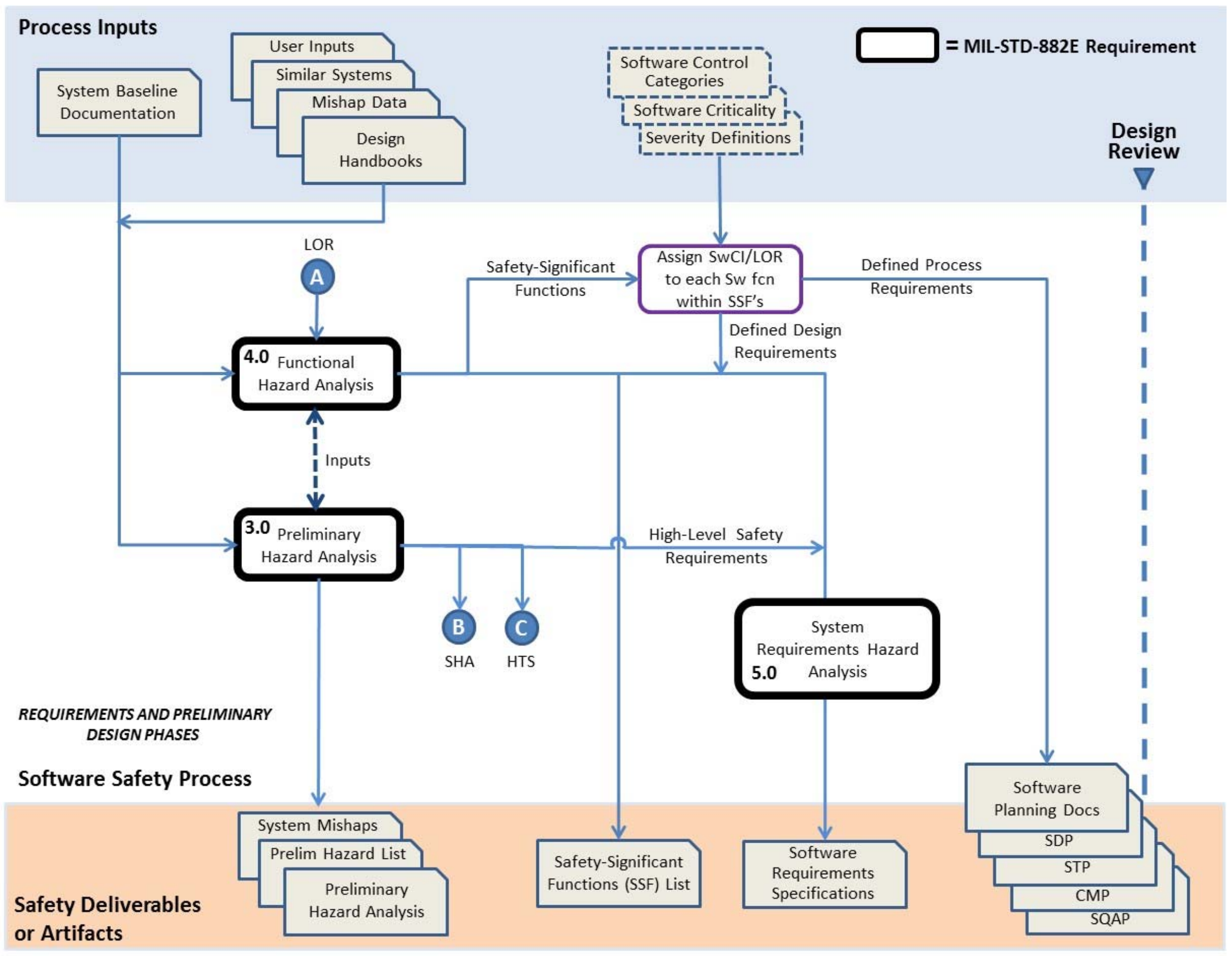

Process Inputs
System Baseline Documentation
User Inputs
Similar Systems
Mishap Data
Design Handbooks
Software Control Categories
Software Criticality
Severity Definitions
= MIL-STD-882E Requirement
Design Review
LOR
A
Safety-Significant Functions
Assign SwCI/LOR to each Sw fcn within SSF's
Defined Process Requirements
Defined Design Requirements
4.0 Functional Hazard Analysis
Inputs
3.0 Preliminary Hazard Analysis
High-Level Safety Requirements
B
SHA
C
HTS
System Requirements Hazard 5.0 Analysis
REQUIREMENTS AND PRELIMINARY DESIGN PHASES
Software Safety Process
Software Planning Docs
SDP
STP
CMP
SQAP
System Mishaps
Prelim Hazard List
Preliminary Hazard Analysis
Safety-Significant Functions (SSF) List
Software Requirements Specifications
Safety Deliverables or Artifacts

Figure 2.0: SSS Process Chart for Requirements and Preliminary Design Phases

Figure 2.1 depicts the initial Developer PHA effort in Process Task 3.0. The PHA commences almost immediately after contract award. Notice that the Functional Hazard Analysis (Section 3.4 Process Task 4.0) is likely to be performed concurrently with the PHA. This is considered acceptable because these two analyses basically provide specific and essential information that brings accuracy and fidelity to each individual analysis. The safety DID pertaining to the PHA is DI-SAFT-80101A.

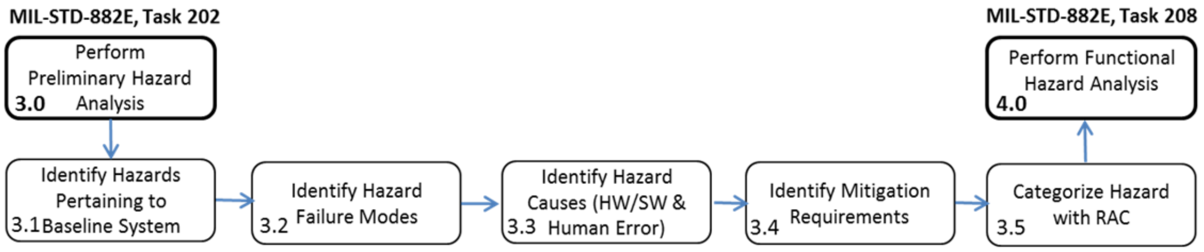

MIL-STD-882E, Task 202
Perform Preliminary Hazard 3.0 Analysis
MIL-STD-882E, Task 208
Perform Functional Hazard Analysis 4.0
Identify Hazards Pertaining to 3.1 Baseline System
Identify Hazard 3.2 Failure Modes
Identify Hazard Causes (HW/SW & 3.3 Human Error)
Identify Mitigation 3.4 Requirements
Categorize Hazard 3.5 with RAC

Figure 2.1: Process Task 3.0 PHA Task and Subtasks

The PHA is performed under the responsibility of SSE and its scope is dictated by the SOW and contract. The PHA is the initial analysis performed on the system for the purpose of the identification of potential hazards and mishaps which are documented within the PHA. The PHA begins with development of the PHL. The PHL, detailed in MIL-STD-882E Task 201, provides a summary list of potential hazards and mishaps for the system, including those with software contributions. SSSE must support the development of the PHA by providing assessment of the system's software within the context of the system. Another important purpose of the PHA is to identify potential failure modes and causes of the hazards in order to define (as early as possible) mitigation requirements for the system and software specifications. Mitigation requirements should be defined as early in the analysis process as possible and documented in the specifications, resulting in fewer derived safety requirements after the design matures within the life cycle process. Section 3.6.3 provides discussion of Generic System Safety Requirements (GSSR). During the PHA, mitigating requirements will likely consist of high level specification requirements and GSSRs. The PHA is one of the earliest opportunities to influence the design and design decisions regarding the use of software in performing SSFs. PHL and PHA results are used to populate the initial Hazard Tracking System (HTS) records as defined in MIL-STD-882E, Task 106. As the results of the PHL and PHA mature and evolve in subsequent safety analyses, the HTS records will also evolve and mature.

3.3.1. Process Subtask 3.1: Identify Hazards Pertaining to the Baseline System

[Ref: MIL-STD Task 202 and JSSSEH paragraph 4.3.4]

The preliminary mishaps and hazards are identified based upon the capabilities the system is to provide, the preliminary design baseline, and the safety risk potential that the system could possess. Preliminary hazards may be either formalized or eliminated as they are peer reviewed by system designers and the SSE team within the SSWG. Regardless of the final disposition of the identified hazard, all hazards are captured and documented in the HTS (MIL-STD-882E, Task 106).

3.3.2. Process Subtask 3.2: Identify Hazard Failure Modes

Hazard failure modes are the primary failure paths leading to a hazard as represented in the example logic diagram of Figure 2.2. In the depicted example, "Loss of Engine" is the system-level hazard with four primary failure modes (there are likely others); Bird Strike; Loss of Fuel to Engine; Failure of Engine Control; and Failure of the Compressor. It should be noted that these failure modes will likely be tracked as separate sub-system-level hazards in a subsequent SSHA. Continuing the logic diagram lower, it is evident that each sub-system-level hazard possesses individual failure modes. It is important to accomplish the analysis for the PHA in a "top-down" manner in order to keep track of the context between mishaps, system-level hazards, sub-system-level hazards, failure modes, and failure mode lower-level causes. Once the hazard failure modes are identified, each failure mode can then be further analyzed for specific causes from a hardware, software, and human error perspective.

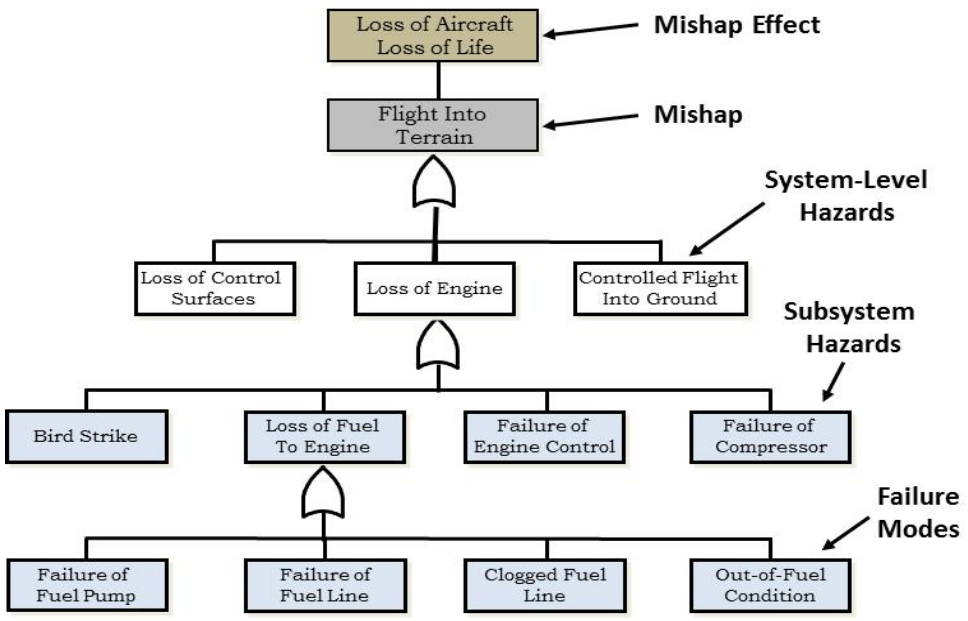


Figure 2.2: Example of Hazard Failure Modes Represented in Simple Logic Diagram

Both "loss of fuel to the engine" and "failure of engine control" events are likely to contain one or more software causes (contributions to fault or failure). The PHA should continue as far down the causal pathway as the design will allow in order for the definition of as many safety-significant software mitigation requirements as possible early in the system's design cycle.

3.3.3. Process Subtask 3.3: Identify Hazard Causes (Hardware/Software & Human Error)

The PHA must be performed in such a way as to be able to see the context of how software reacts to hardware and human operators, and how the hardware and the human reacts to how the software functions. Hardware, software and human operator hazard causes must be addressed within the PHA to ensure that functional and physical interfaces are included in the analysis. Note: It is essential that the PHA and follow-on analyses be performed to the depth necessary for the identification of specific hazard mitigation requirements that provide the evidence of sufficient AND-Gate protection (e.g., design redundancy) against the probability of failure propagation to the top event.

3.3.4. Process Subtask 3.4: Identify Mitigation Requirements

[Ref: JSSSEH paragraph 4.3.5.1.3]

Because the PHA is performed early in development, the hazard mitigation requirements that are identified by the PHA and the FHA analysis might be more general (high level requirement) in nature. Multiple derived lower-level requirements may be necessary to fulfill a high-level requirement as the design matures. This derivation of lower level requirements must occur as the system design matures. From the SSSE perspective, it is important to understand context for potential mitigation requirements that may be assigned to software and to ensure their integration into the PHA and software specifications.

3.3.5. Process Subtask 3.5: Categorize Hazards with Risk Assessment Code (RAC)

Each identified and documented hazard and mishap is initially categorized in terms of mishap severity and likelihood of occurrence. This initial RAC is assigned prior to any mitigation action unless there is evidence of specific provisions, alternatives, and mitigation measures that have been verified to have been implemented within the design to reduce risk. MIL-STD-882E defines severity and likelihood of occurrence (probability) and must be used unless tailoring has been approved by the Program's DoD Component Executive. Each hazard and mishap is assigned an initial risk assessment, commonly known as a RAC, and documented in the HTS hazard record. RACs can be obtained from the Risk Assessment Matrix as defined in Table III of MIL-STD-882E. Software does not have a probability of occurrence component, so it is not necessary to assign a probability at the software causal level (of the hazard), but the PHA should consider how a software cause(s) affects the overall hazard and mishap probability of occurrence.

RAC is the allocation of severity and probability of the mishap when all hazard mitigations are considered in the design and requirements are implemented for the procedures and training of personnel operating and maintaining the system. In most instances, the severity of the hazard or mishap will not change, unless the associated system capabilities and design changes. Changes in RAC, from initial assignment to risk acceptance, will likely be a reduction in probability only. The PHA task concludes with the capture of all PHA data in the hazard analysis worksheets that are imported into the Acquirer-approved HTS.

3.4. Process Task 4.0: FHA

[Ref: JSSSEH paragraph 4.3.3, and MIL-STD-882E Task 208]

The FHA is another foundational SSE analysis performed under the responsibility of system safety engineering and its scope is dictated by the SOW and contract. Additionally, virtually all safety review authorities expect a FHA as part of the program's objective evidence for identifying and classifying the system functions and the safety consequences of functional failure or malfunction prior to obtaining review acceptance and concurrence. The FHA is one of the most important analyses that the system safety analyst will perform. As the software implements functions within the context of system, it is essential to understand which functions are safety-significant and which of these will be implemented by the software. It is also important to ensure (by LOR analysis and test tasks) that the SSFs implemented by the software perform exactly as intended and that they do not perform any unintended functions. Further still, and given the fact that software will possess control over safety-significant functions and that undesired events are likely to occur, it is important that fault/failure detection, isolation, annunciation, and tolerance is built into the system and software design architectures. The FHA is the first step in reaching these objectives. The Process Subtasks of the FHA are presented in Figure 2.3 below.

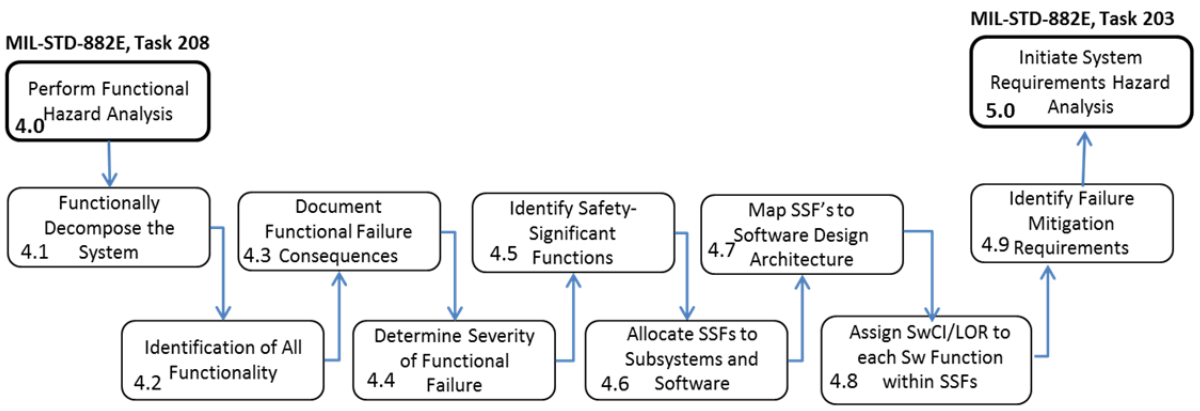

MIL-STD-882E, Task 208
4.0 Perform Functional Hazard Analysis
4.1 Functionally Decompose the System
4.2 Identification of All Functionality
4.3 Document Functional Failure Consequences
4.4 Determine Severity of Functional Failure
4.5 Identify Safety-Significant Functions
4.6 Allocate SSFs to Subsystems and Software
4.7 Map SSF's to Software Design Architecture
4.8 Assign SwCI/LOR to each Sw Function within SSFs
4.9 Identify Failure Mitigation Requirements
MIL-STD-882E, Task 203
5.0 Initiate System Requirements Hazard Analysis

Figure 2.3: Process Task 4.0 Task and Subtasks

The FHA described here is not the same as the FHA described in SAE ARP 4761 that is required for Airworthiness Release. There are different purposes for the two analyses. The primary purpose of the FHA described in SAE ARP 4761 is the identification of mishaps and hazards by analyzing the system functionally. Conversely, a primary purpose of the FHA described here and in MIL-STD-882E is to identify all system functionality, determine which are safety-significant and implemented by the software, and then map these SSFs to the software design architecture. Once mapped to the architecture, mitigating requirements can be identified. By performing the FHA described here and in MIL-STD-882E, the analyst will be afforded insight to the mishaps and hazards of the system. It should also be noted that there is no reason why the FHA format cannot be formatted in such a way to meet the intent and purpose of both SAE ARP-4761 and the safety FHA described here.

3.4.1. Process Subtask 4.1: Functionally Decompose the System

The information contained in the FHA reflects the same level of maturity as the design architecture. This is expected, and reinforces that the FHA must be kept current through all phases of the development lifecycle, to include functional, physical, and contractual changes made under configuration control. Frequency of updates to the FHA should be specified within the SOW and contract. However, SSSE should update the software inputs to the FHA IAW the software development process and schedule. The format of the FHA should reflect that which will provide the analysis "answers" required by the analyst and criteria of the contract.

The first step of the analysis is to decompose the system. If the system is mature enough, this first step may be a physical decomposition of the system. If the system has not yet been allocated to specific pieces of hardware, this decomposition will be functional. The system must be analyzed functionally from the perspective of both "what the system is documented to do functionally," and "what you think the system can do functionally." The former is an assessment of documented functionality from the functional specifications and the latter is assessed by analyzing the functionality of the physical components of the system. The analysis of the physical attributes of the system is likely to provide insight to "hidden" or undocumented functionality. This is especially true for systems heavily using Commercial-off-the-Shelf (COTS) components.

FUNCTIONAL HAZARD ANALYSIS								
System Decomposition	**Individual Functional Descriptions**	**Functional Failure Modes**	**Consequence of Each Failure Mode**	**Severity of Consequence of Failure Modes**	**Safety-Significant Functions**	**Assignment of SCC and SwCI**	**Map to Software Design**	**Failure Mitigation Requirements**
Process Subtask 4.1	Process Subtask 4.2	Process Subtask 4.3	Process Subtask 4.3	Process Subtask 4.4	Process Subtask 4.5	Process Subtask 4.8	Process Subtask 4.7	Process Subtask 4.9

Figure 2.4: Example FHA Format

Figure 2.4 provides an example of a FHA format that will provide the analyst with the most basic of information required by the analysis. If the analyst (or the Acquirer) requires more than this simple example format can provide, the format can be tailored to add the appropriate columns to the format to identify and track the information required. The decomposition of the system is documented in Column one. System decomposition can be done in a Work Breakdown Structure -like structure which may aid in structure, flow, traceability, and assignment of responsibilities. For instance, on large, complex programs such as an Aircraft (see Figure 2.2) the hazard "Loss of Engine" may be completely under the control of the Engine Integrated Product Team (IPT). The Engine IPT is more likely to support safety if the FHA can readily show the IPT which parts it is responsible for.

3.4.2. Process Subtask 4.2: Identification of All Functionality

Column two of the example FHA format in Figure 2.4 depicts where the system functionality is documented. For the initial FHA, the functionality may be "higher level" functions that have not yet been decomposed to lower level functionality. For an initial FHA this is sufficient for this level of analysis maturity as lower-level functionality will likely take on the same criticality as their parent higher-level functions. Ensure that all functionality is identified. First, identify what you think it can or should do functionally. Second, compare the functionality identified with the documented functionality of the functional specification and reconcile the two lists. Lastly, identify any functionality that is identified in hardware literature or the performance specification to determine whether there are "hidden" or unintended functions residing in the system. During this activity, it is also good to keep a list of undesired functions from a safety perspective. It will be imperative to ensure that the system either does not have the capability to perform undesired functions or that the system possesses the necessary inhibits and/or interlocks in place to ensure these functions do not occur when they pose a safety risk.

3.4.3. Process Subtask 4.3: Document Functional Failure Consequences

Once all known functionality of the system is identified and documented, each function must be assessed against the following scenarios:

- The function is unavailable (does not occur when expected to occur).
- The function malfunctions (degraded, partial, or unexpected results of the function).
- The function performs its intended activity but is out of sequence.
- The function performs its intended activity, but at the incorrect time (too early, too late, outside defined window).

When documenting the consequences of functional failure it is important to understand that the consequences can (and will likely) be different for each of the failure scenarios described above. Each functional failure consequence must be documented in the FHA table.

3.4.4. Process Subtask 4.4: Determine Severity of Functional Failure Consequences

After all functional failure consequences are adequately identified and documented, each must be assessed against the hazard and mishap severity definitions as defined by the SSPP. This is only an assessment against severity of the consequence and not the likelihood that it will occur. Once a severity allocation is determined for each functional failure scenario consequence, these allocations are documented in column five of the example FHA depicted in Figure 2.4. MIL-STD-882E Task 208 specifies inclusion of a probability component to the analyses. If specified for the program, a column should be added to the analysis worksheet to document the probability assigned for the function to be allocated to the functional design. Assignment of both severity and probability components of the RAC to each SSF supports management risk decisions. The probability of a function failing or malfunctioning as a causal factor to a hazard must be accounted for within the context of the hazard (record) that it applies.

3.4.5. Process Subtask 4.5: Identify SSFs

Each functional failure consequence is assessed against the severity definitions and formally documented in the FHA. For an individual function, there may be multiple severity consequences, and severity of consequences for that function. However, the function is assigned the worst-case severity consequence as determined by the analysis. The functions will be identified as having a safety consequence or no safety consequence based upon their linkage to a mishap/hazard. Those with a safety consequence will be referred to as SSFs and be assigned to the following two subcategories:

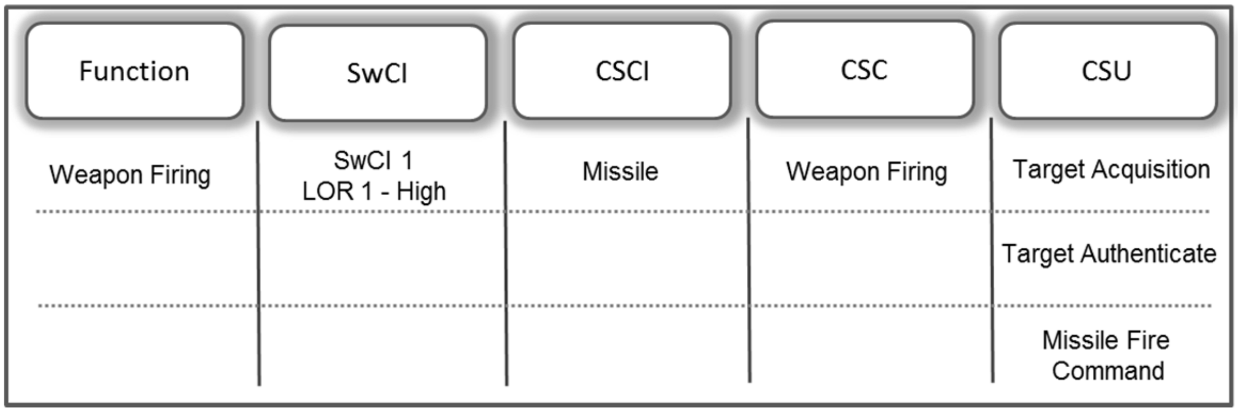
- Safety Critical Function (SCF): Functions that possess either a Catastrophic or Critical severity consequence
- Safety-Related Functions (SRF): Functions that possess either a Marginal or Negligible severity consequence

3.4.6. Process Subtask 4.6: Allocate SSFs to Subsystems and Software

As SSFs are identified and classified as SCF or SRF, each must be allocated to the respective subsystem and software that implement them. The initial version of the FHA, performed early in the program, may only be able to allocate SSFs to major subsystem and the software levels only. However, as the FHA evolves and is updated as the design matures, the allocation to specific subsystems and software implementation must be specified. The greater the fidelity of the allocation of SSFs to subsystems and software, the more effective SSE will be in working with the design and architecture, software development and test teams in focusing required hazard mitigations and LOR resources.

3.4.7. Process Subtask 4.7: Map SSFs to the Software Design Architecture

Once all SSFs have been identified, the analyst must map each function to its software design architecture (to either software modules of code or “use cases” in Object Oriented Design (OOD)). This is required both for end-to-end traceability of requirements and to support subsequent detailed analyses. This will be important when specific hazard analysis is accomplished and software causes to hazards are identified and analyzed. This mapping will provide the analyst with a defined point of entry to the software in order to analyze the software's contribution to hazard initiation or propagation. This mapping will also allow the analyst to determine the simplicity or complexity of the design of the SSFs and the effectiveness of functional and physical partitioning of the software design architecture. An example of the SSF mapping that is required is presented in Figure 2.5. The SwCI is determined in the next section (3.4.8) although it is represented in Figure 2.5. Mapping the SSFs to the software design architecture supports the derivation of the SwCI since it may be necessary to define the software control category, a key element of the SwCI.



Function	SwCI	CSCI	CSC	CSU
Weapon Firing	SwCI 1 LOR 1 - High	Missile	Weapon Firing	Target Acquisition
				Target Authenticate
				Missile Fire Command

Figure 2.5: SSF Mapping to Software Design Architecture Example

3.4.8. Process Subtask 4.8: Assign SwCI to each Software Function within SSFs

[Ref: MIL-STD-882E paragraph 4.4.1 and JSSSEH paragraph 4.2.1.4]

Software's functional criticality as described in MIL-STD-882E, Table V, is determined by the unmitigated severity of the consequence of functional failure (or malfunction) in conjunction with the software control category assignment as defined in MIL-STD-882E, Table IV. The result is the SwCI assignment from SwCI 1 to SwCI 5. The SwCI will be assigned for each function assessed. The SwCI assignment becomes the LOR by definition and provides the software developers and testers with the safety assurance requirements for the design of each.

Process subtask 4.8, depicted earlier in Figure 2.3, identifies the assignment of SwCI to each software function with identified SSFs. Allocation of SSFs to specific LOR categories is essential, both to ensure the provision of rigor to the functions of highest safety criticality and to ensure the management of the critical resources necessary to implement that rigor. This Process task can be integrated into the accomplishment and documentation of the FHA. Regardless of whether it is included in the FHA or accomplished separately, the accomplishment of the specific subtasks comprising subtask 4.8, identified in Figure 2.6, must be performed for each SSF and be thoroughly documented within the artifacts of the safety analysis.

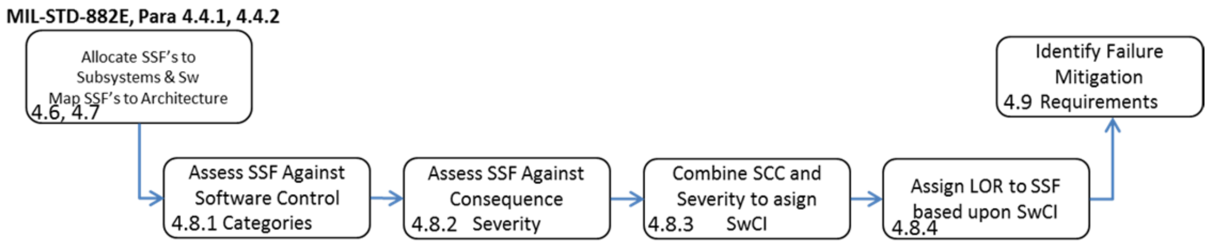

MIL-STD-882E, Para 4.4.1, 4.4.2
Allocate SSF's to Subsystems & Sw Map SSF's to Architecture
4.6, 4.7
Assess SSF Against Software Control
4.8.1 Categories
Assess SSF Against Consequence
4.8.2 Severity
Combine SCC and Severity to asign
4.8.3 SwCI
Assign LOR to SSF based upon SwCI
4.8.4
Identify Failure Mitigation
4.9 Requirements

Figure 2.6: Process Subtasks for 4.8 Assign SwCI/LOR

3.4.8.1. Process Subtask 4.8.1: Assess SSF against Software Control Categories (SCC)

MIL-STD-882E, Table IV defines the SCCs. These definitions may be tailored if change is warranted. However, all tailoring must be thoroughly justified and approved by the appropriate Acquirer authority in accordance with MIL-STD-882E. This subtask focuses on the assessment of SSFs against the defined definitions of the SCCs documented in the SSMP and SSPP. Accurate assessment of the SCC based upon the complexity of the system, autonomy of the system's functionality, and/or its command and control authority is imperative.

3.4.8.2. Process Subtask 4.8.2: Assess the SSF for the Consequence Severity

This task should have been accomplished and the information documented in the FHA. This information is also required at this point to assess the criticality of the SSF against SCC and the worst-case Severity criteria for the purpose of assigning the LOR to the function.

3.4.8.3. Process Subtask 4.8.3: Combine SCC and Severity to Assign SwCI

The Software Safety Criticality Matrix (SSCM) provides the LOR allocation for SSFs. MIL-STD-882E Table V defines the SSCM. Once the SCC and the severity of consequence of the hazard/mishap are determined for a SSF, the LOR can be determined by the predefined and approved SSCM. The SwCI specifies the LOR value for SSFs and is determined by combining the SCC and the severity of consequence (from the FHA), which were previously assigned to the SSF.

3.4.8.4. Process Subtask 4.8.4: Assign the LOR to the SSF

Combining the activities of Process Subtasks 5.2 and 5.3, the LOR can now be assigned to the assessed SSF. An LOR assignment of 1-5 is allocated to each SSF and the SSF must be designed, coded, tested, and verified against the approved LOR criteria.

3.4.9. Process Subtask 4.9: Identify Failure Mitigation Requirements

Specific software contributions to hazard and mishap failure conditions must be adequately mitigated as a design priority. As insight and design maturity is obtained, insight as to how the system is to function, its physical characteristics, and the potential failure pathways to hazards, must be used to determine whether adequate mitigation is either present or absent in the design. If adequate hazard mitigation is already present or accounted for in the form of preliminary (either generic best practice, or from the PHA) requirements, those requirements should be tagged for follow-on safety verification and validation. However, if the FHA identifies a shortfall in hazard control, mitigation requirements must be identified and documented and communicated to the system engineering and software development teams for inclusion in the applicable system and software requirements specifications. The initial FHA task concludes with the capture of all FHA analysis data in the hazard analysis worksheets that are imported into the Acquirer-approved HTS.

3.5. Process Task 5.0: Initiate System Requirements Hazard Analysis (SRHA)

[Refer: JSSSEH 4.3.5 and MIL-STD-882E Task 203]

The primary mechanism to "influence the design" in order to reduce the safety risk is to define specific safety-significant requirements and include them in the system and software specifications. Safety requirements are the primary mechanisms to fulfill the first step in the system safety order of precedence; design for minimum risk.

MIL-STD-882E, Task 203, if specified by the contract is to perform and document a SRHA to determine the design requirements to eliminate hazards or reduce the associated risks for a system, to incorporate these requirements into the appropriated system documentation, and to assess compliance of the system with these requirements. Within the JSSSEH, this process task is Section 4.3.5 Safety Requirements Analysis (SRA). Regardless of whether MIL-STD-882E, Task 203 is specified on the contract or not, the requirements analysis tasks must be performed as part of the SSE process. The JSSSEH SRA task executes a process to ensure that the safety constraints and criteria of the system are aligned with the safety requirements of the system to minimize the safety risk potential of the hazards within predefined Concept of Operations (CONOPS). Safety requirements to minimize the safety risk potential that are present in the specifications, are tagged as safety-significant. Tagging of requirements usually takes place in the Developer's requirements management and traceability toolset. Where requirements are absent, they must be defined, documented, tagged, and included into the specifications. If the new system safety requirements (SSRs) are not added to the specification, the contribution to the applicable system risk(s) must be included in the hazard analysis risk assessment. The subtasks for initiating the JSSSEH SRA process are defined in Figure 2.7. The results of the SRA must establish and provide evidence of bi-directional traceability (top down and bottom up) from safety criteria and specifications to design, implementation, Verification and Validation (V&V) and mishaps and hazards. The JSSSEH SRA task accomplishes many of the requirements specified within Task 203 SRHA. However, there are additional items prescribed by Task 203 SRHA that must be accounted for if that task is specified on the contract. Figure 2.7.1 specifies the additional Task 203 SRHA subtasks that must be accomplished to successfully address that contractual analysis.

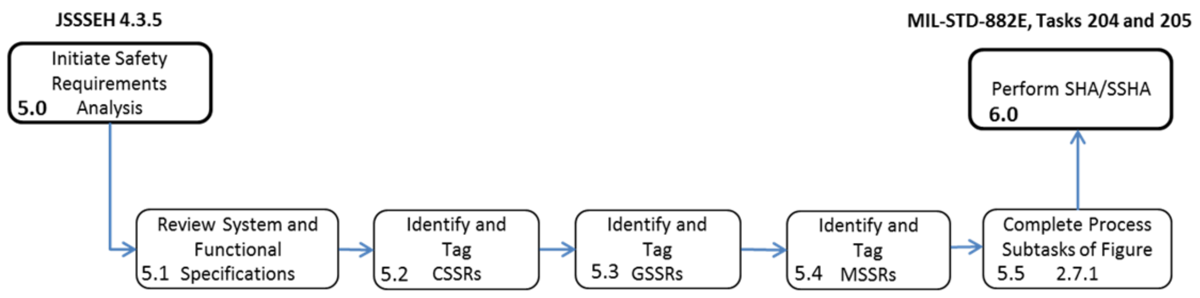

JSSSEH 4.3.5
MIL-STD-882E, Tasks 204 and 205
Initiate Safety Requirements Analysis 5.0
Perform SHA/SSHA 6.0
Review System and Functional Specifications 5.1
Identify and Tag CSSRs 5.2
Identify and Tag GSSRs 5.3
Identify and Tag MSSRs 5.4
Complete Process Subtasks of Figure 2.7.1 5.5

Figure 2.7: Process Task 5.0 Initiate Safety Requirements Analysis Task and Subtasks

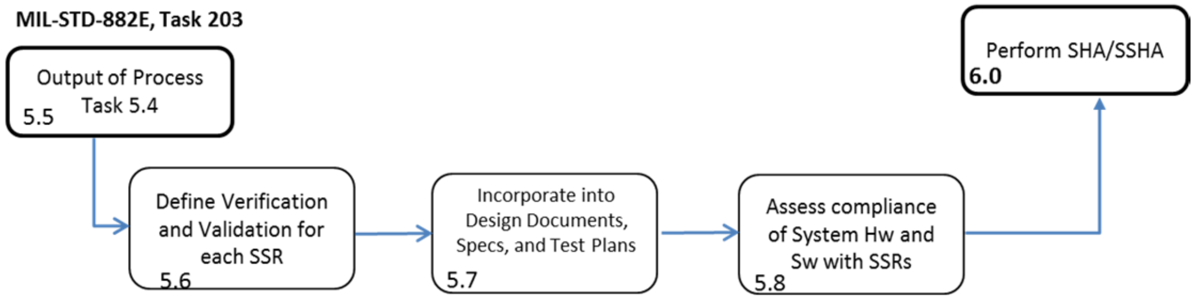

MIL-STD-882E, Task 203
Output of Process Task 5.4 5.5
Perform SHA/SSHA 6.0
Define Verification and Validation for each SSR 5.6
Incorporate into Design Documents, Specs, and Test Plans 5.7
Assess compliance of System Hw and Sw with SSRs 5.8

Figure 2.7.1: Initiate System Requirements Hazard Analysis Task and Subtasks

3.5.1. Process Subtask 5.1: Review System and Functional Specifications

This subtask involves a review of the system and functional specifications for the system in development. The primary purpose of this task is to identify and include the missing requirements from a safety perspective. SSS should be a part of the configuration control process. SSS must provide their inputs to the requirements identification process in a timely manner. SSS must thoroughly review the preliminary specifications, as well as proposed requirements change actions, and determine where safety-significant requirements are necessary for incorporation.

3.5.2. Process Subtask 5.2: Identify and Tag Contributing Safety-Significant Requirements (CSSR)

Requirements that are safety-significant must be identified and tagged. For example, a requirement to "Issue Fire Command" is a safety-significant requirement because it "contributes" to the safety risk potential of the system, it does not mitigate it. Subsequently identified and tagged derived lower level requirements provide the actual mitigations to mishaps associated with "Issue Fire Command."

3.5.3. Process Subtask 5.3: Identify and Tag GSSR

This task focuses on the identification of initial (generic) safety requirements for the system and getting them included into both the hardware and software specifications. These GSSRs are based on:

- Lessons learned from other programs.
- Similar systems hazard analysis.

- Generic lists of “best practices” (e.g., JSSSEH Appendix E; STANAG 4404).
- The safety implications of the choice(s) of particular programming languages and development processes.
- Historical mishap data on similar system.
- User (operator, maintainer, supporter, etc.) inputs.
- Information gleaned by accomplishing the PHA.
- Information gleaned by accomplishing the FHA.

3.5.4. Process Subtask 5.4: Identify and Tag Mitigating Safety-Significant Requirements (MSSR)

MSSRs are safety significant requirements that specifically provide mitigation of identified hazard and mishap causes. MSSRs can only be identified if hazard analysis is accomplished to the detail level necessary to specifically derive new requirements that mitigate a cause of the hazard by reducing the likelihood of causal initiation and/or causal propagation.

MSSRs can also be derived by decomposing higher-level requirements such as GSSRs into lower-level requirements for the design. Higher-level GSSRs, such as “The system shall initialize into a known and predefined safe state,” must be decomposed to lower-level requirements that mitigate the possibility of initializing into an unsafe state or define specific safe states of the system. These lower level requirements can only be identified, in this specific example, after the specific steps of initialization are defined and the unsafe states or conditions of the system identified. As with all the safety-significant requirements that are identified and tagged within the requirements management and traceability application, they must be traced both to design, and back to the hazard(s) that it helps to mitigate. In addition, all safety-significant requirements must take on the LOR criteria of the function that it is implementing within the design architecture. The initial SRHA task concludes with the analysis of all requirements documented, an associated assessment of all requirements gaps and an assessment of compliance of the development of the system hardware and software with identified SSRs. Results from the analysis are imported into the Acquirer-approved HTS.

As the design of the system matures, the SRHA must also mature. The maturation of the SRHA will be covered in Process Task 7.0, Finalize SRHA.

3.5.5. Process Subtask 5.5: Document Output of SRA Steps

The purpose of this task is to document the results of the SRA task into the Acquirer specified format of the SRHA, if the SRHA Task 203 has been specified for the contract. The results of the SRA task will completely satisfy the SRHA requirement or provide core input to the SRHA.

3.5.6. Process Subtask 5.6: Define Verification and Validation for Each SSR

The purpose of this subtask is to define the verification and validation approaches for each SSR identified to eliminate hazards or reduce associated risk. The verification and validations methods are typically classified as:

- Inspection - An examination of the item against applicable documentation to confirm compliance with requirements.
- Analysis - Use of analytical data or simulations under defined conditions to show theoretical compliance. Used where testing to realistic conditions cannot be achieved or is not cost-effective.
- Demonstration - A qualitative exhibition of functional performance, usually accomplished with no or minimal instrumentation.
- Test - An action by which the operability, supportability, or performance capability of an item is verified when subjected to controlled conditions that are real or simulated. These verifications often use special test equipment or instrumentation to obtain very accurate quantitative data for analysis.

3.5.7. Process Subtask 5.7: Incorporate Each SSR into the Applicable Documents

SSRs identified and approved must be incorporated into the applicable engineering design documents, and hardware, software, and system test plans.

3.5.8. Process Subtask 5.8: Assess Compliance of the System Hardware and Software with SSRs

As the program proceeds and the design evolves and matures, the SSE must continue to analyze the system and software design, architecture, hardware, and software for compliance with the SSRs. SSRs must be addressed at all contractually required technical reviews, and SSE must address the hazards, mitigation measures, and means of verification and validation for SSRs at technical reviews. As test plans are developed and executed, both the plans and results must be reviewed for compliance with SSRs. Ultimately, at the end of the SRHA effort, applicable mitigation information must be incorporated into manuals and plans.

3.6. Process Task 6.0: Perform System and Subsystem Hazard Analyses

[Refer: JSSSEH paragraph 4.3.6 and MIL-STD-882E Tasks 204-206, and 209 for SoS]

As the system design matures, the hazards identified in the PHA and the failure modes identified in the FHA must be transitioned from the PHA and FHA to the SSHA and the SHA as defined in Figure 3.0. The SSHA and SHA are SSE analyses performed under the responsibility of SSE and their scope is dictated by the SOW and contract. Whether the SSS analyst is working at the system level or the subsystem level analysis, this is the phase of the program where in-depth causal analysis typically takes place due to the availability of documentation for a maturing design.

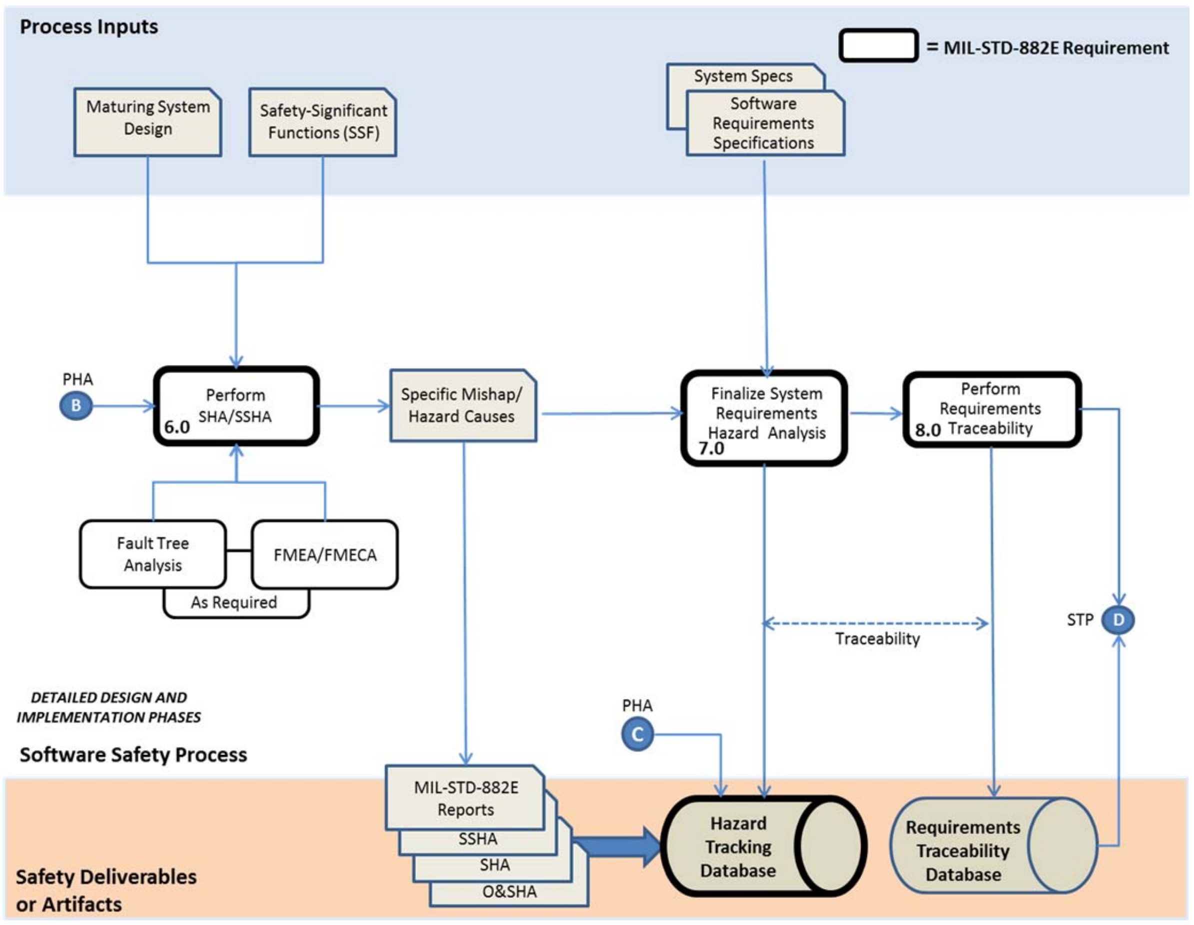

Process Inputs
= MIL-STD-882E Requirement
Maturing System Design
Safety-Significant Functions (SSF)
System Specs
Software Requirements Specifications
PHA
B
Perform SHA/SSHA
6.0
Specific Mishap/ Hazard Causes
Finalize System Requirements Hazard Analysis
7.0
Perform Requirements Traceability
8.0
Fault Tree Analysis
FMEA/FMECA
As Required
Traceability
STP
D
DETAILED DESIGN AND IMPLEMENTATION PHASES
Software Safety Process
PHA
C
MIL-STD-882E Reports
SSHA
SHA
O&SHA
Hazard Tracking Database
Requirements Traceability Database
Safety Deliverables or Artifacts

Figure 3.0: SSS Process Chart for Detail Design and Implementation Phases

Regardless of SSS analysis techniques used, it must have the ability to allow the analyst to:

- Map or track SSFs to specific modules (or use cases) of code.
- Possess insight into the software's functional and physical interfaces with hardware, other modules of software, or the human interface with the system.
- Provide insight to both the system and software design architecture.
- Comprehend what could functionally take place within the software design or code based upon loss of function, degraded function (or malfunction) or functioning outside the bounds of the predetermined parameters of timing and sequencing events.
- Determine where fault management should reside within the software design architecture (fault detection, isolation, annunciation, logging, tolerance, and/or recovery).

The process subtasks of Process Task 6.0, Perform SHA/SSHA are depicted in Figure 3.1 below.

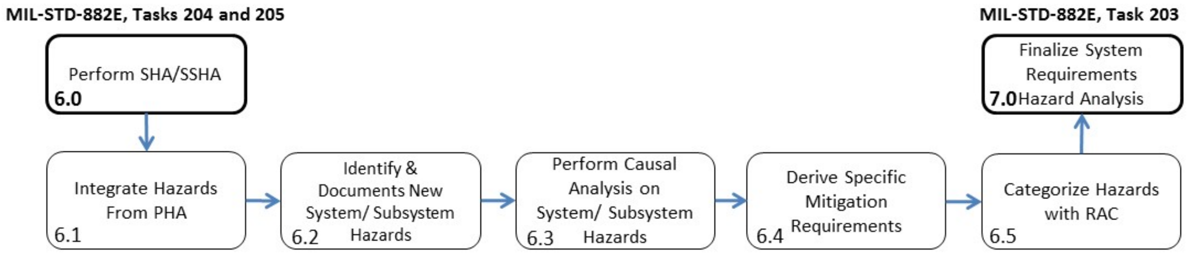

MIL-STD-882E, Tasks 204 and 205
Perform SHA/SSHA
6.0
MIL-STD-882E, Task 203
Finalize System Requirements Hazard Analysis
7.0
Integrate Hazards From PHA
6.1
Identify & Documents New System/ Subsystem Hazards
6.2
Perform Causal Analysis on System/ Subsystem Hazards
6.3
Derive Specific Mitigation Requirements
6.4
Categorize Hazards with RAC
6.5

Figure 3.1: Process Task 6.0 Perform In-depth Hazard Analysis and Subtasks

3.6.1. Process Subtask 6.1: Integrate Hazards from the PHA

This subtask was introduced in the previous paragraphs. From the total set of hazards considered in the prior phases of the safety program, only those that are determined to be credible for the system and its intended test and operational environments are carried forward to the SSHA or the SHA. In addition, some of the hazards may be transitioned to either the O&SHA or the Health Hazard Analysis (HHA).

3.6.2. Process Subtask 6.2: Identify and Document New System/Subsystem Hazards

The purpose of the SHA and SSHA is to take the safety information generated in the predecessor PHA and FHA, combined with safety assessment of the system interfaces and maturing subsystems to identify system and subsystem hazards. The more architecture and design specific SHA and SSHA will likely yield new hazards that were not discovered during the PHA and FHA. Identified subsystem and system hazards are formally entered into the SSHA and SHA worksheets and each must be analyzed to ensure the hazard risk assessment and associated RAC is reflective of the matured design, implementation and mitigations.

3.6.3. Process Subtask 6.3: Perform Causal Analysis on System/Subsystem Hazards

This task requires access to up-to-date and accurate system design documentation. Regardless of the methods or tools used to perform the causal analysis, the results must be at a level:

- Necessary to either account for mitigation already in the design architecture (probably as the result of the GSSRs included in the early versions of the specifications), or to derive MSSRs where mitigation is either absent or insufficient.
- Sufficient to account for software causal factors (either as causal initiators, or causal propagations).
- Sufficient to comprehend the interdependencies and interfaces between hardware, software, and human error causes.
- Necessary to account for physical, functional, or contractual interfaces between the system integrator and other sub-developers or vendors.
- That validates the rationale to discontinue analysis at a lower level (further down the causal pathway to its root source).

One of the best ways to determine the adequacy of the design architecture in context with the systems' functional and operational environments is to accomplish a simple logic diagram (event or fault tree) of the hazard and its causal pathways. This provides a graphical representation of the hazard causes in conjunction with the Boolean "AND" and "OR" logic required to accomplish an estimation of the adequacy of the probability of occurrence. If a quantitative Fault Tree Analysis (FTA) tool is utilized as a method to understand the design logic, the failure probability of software functionality should be set to "one (1)" to understand the control of the software within the system context and impact to top-level event failure probability and cut sets. This will help to demonstrate the dependency of the software functionality on the design architecture.

3.6.4. Process Subtask 6.4: Derive Specific Mitigation Requirements

[Ref: JSSSEH paragraph 4.3.5.1.3 and 4.3.5.2]

As stated above in process subtask 6.3, the hazard causal analysis will either confirm the existence and adequacy of hazard mitigation, or it will determine that it is either nonexistent or inadequate. In the latter case, the remaining safety risk potential must be adequately dealt with from a hazard mitigation perspective. This task requires that specific mitigation requirements be derived and successfully included in the design maturation process, such as the system safety program documenting a Change Request to explicitly identify an existing design requirement that implements the identified mitigation as part of the program's CCB process .

3.6.5. Process Subtask 6.5: Categorize Hazards with a RAC

[Refer: MIL-STD-882E paragraph 4.3 and subsections; and JSSSEH paragraph 4.4.4]

Upon completion of the causal analysis and assessing the adequacy of the mitigation or control of each hazard failure mode (propagation pathway), each hazard and mishap is then reassessed against the risk assessment criteria of the SSPP. Each hazard record should be assessed for its RAC and that RAC should be annotated in the record. Each hazard analysis task concludes with the capture of all safety analysis data in the Acquirer-approved HTS in the form of individual hazard records.

3.7. Process Task 7.0: Finalize SRHA

[Refer: JSSSEH paragraph 4.3.5 and MIL-STD-882E Task 203]

Process task 5.0 was the initial SRHA, performed early in system development. Task 7.0 represents the culmination of the SRHA during the SHA and SSHA efforts of process task 6.0. This task will ensure that safety requirements analysis is formally completed and adequately documented. This formal documented safety artifact will be an essential document to be revisited for future updates or changes made to the system. The subtasks of this process are presented in Figure 3.2 below.

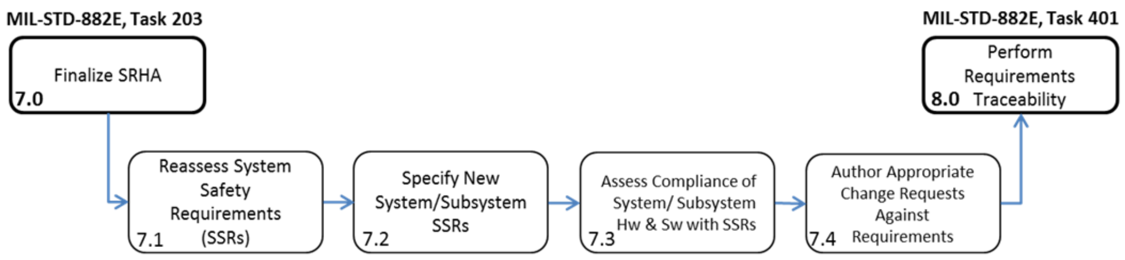

MIL-STD-882E, Task 203
Finalize SRHA
7.0
Reassess System Safety Requirements (SSRs)
7.1
Specify New System/Subsystem SSRs
7.2
Assess Compliance of System/ Subsystem Hw & Sw with SSRs
7.3
Author Appropriate Change Requests Against Requirements
7.4
MIL-STD-882E, Task 401
Perform Requirements Traceability
8.0

Figure 3.2: Process Task 7.0 Finalize System Requirements Hazard Analysis

3.7.1. Process Subtask 7.1: Reassess SSRs

SSS must first reassess the GSSRs and MSSRs defined for the program as part of Process Task 5.0 Initial SRHA. There must be evidence within the design architecture or design processes that these requirements have been adequately addressed. Traceability from GSSRs to these artifacts must be evident. MSSRs must be reassessed to determine those initial MSSRs that must be further decomposed and allocated based upon the results of the SHA/SRHA.

3.7.2. Process Subtask 7.2: Specify New System/Subsystem SSRs

This task formally wraps up and documents the specific GSSRs and MSSRs to mitigate and control the SSHA and SHA hazards. Much of this effort was accomplished in process subtask 6.4, but it is important to finalize the SRHA documentation in this area. Traceability from SSRs to the design must be evidenced and documented in this phase of development.

3.7.3. Process Subtask 7.3: Assess Compliance of System/Subsystem Hardware and Software with SSRs

As with process subtask 7.2, this process subtask is also a summation of the efforts that were accomplished to finalize each hazard record in Process Task 6.0 and assess the current RAC. The software test cases and procedures must be reviewed to ensure that the testing actually verifies the safety-significant requirements in context to their intended or expected functionality. Further discussions regarding test is provided in Process Task 10.0. Traceability is from safety-significant requirements (or functions) to design, and then to software test cases and results. Traceability must be documented and provided as evidence.

3.7.4. Process Subtask 7.4: Author Appropriate Change Requests against Requirements

This process subtask is also a summation of efforts accomplished in process task 6.0. As the software design is being reviewed and design reviews are accomplished, incorrect interpretation of the safety-significant requirements must be identified and adjudicated. The actual documentation associated with a defect or deficiency will be governed by the configuration management tools used and the Configuration Management Plan. The primary cause of software defects is poorly defined, ambiguous, unclear, incorrect, or missing requirements. Therefore, the SSE should work with the requirements team and engineering teams to author any necessary requirements inclusions or clarifications that remain for action to reduce safety risks. The SRHA concludes with the analysis of all requirements documented and the results from the analysis included in the Acquirer-approved HTS.

3.8. Process Task 8.0: Perform Final Safety Requirements Traceability

[Refer: JSSSEH paragraphs 4.3.5.3, 4.3.6.3.3]

An important task of SSS is the preparation of the safety-significant engineering artifacts that provide the evidence or audit trail of the SSS work accomplished. As depicted in Figure 3.3, the SSRs of the system must be sufficiently traced to the design and also back to their corresponding hazards and mishaps to complete the evidence audit trail. Much of the work required for this task has already been performed during the previous analysis tasks. As such, this Requirements Traceability task is intended to be a final assessment of end-to-end SSRs traceability prior to conducting software testing and Code-level Analysis for LOR-1 software. The importance of performing this final assessment of traceability is to reduce the risk and impacts of either failing to adequately test SSRs or having to do separate, late testing of SSRs.

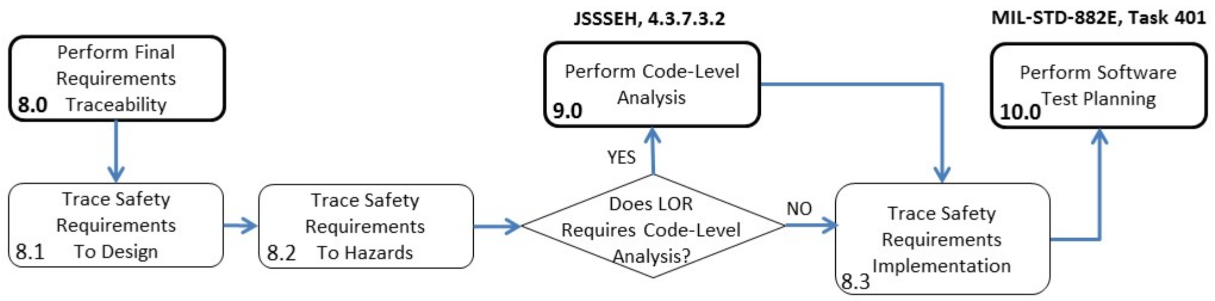

JSSSEH, 4.3.7.3.2
MIL-STD-882E, Task 401
8.0 Perform Final Requirements Traceability
9.0 Perform Code-Level Analysis
10.0 Perform Software Test Planning
YES
NO
8.1 Trace Safety Requirements To Design
8.2 Trace Safety Requirements To Hazards
Does LOR Requires Code-Level Analysis?
8.3 Trace Safety Requirements Implementation

Figure 3.3: Process Task 8.0 Perform Final Requirements Traceability and Subtasks

3.8.1. Process Subtask 8.1: Trace Safety Requirements to Design Architecture

As depicted conceptually in Figure 3.4, there is traceability from the hazard/mishap record, where causes are determined and mitigations identified via mitigation requirements, to the design implementation of the requirements within the system design architecture. This traceability from hazards to the design is essential to ensure mitigation requirements are complete, correct, consistent, implementable and verifiable.

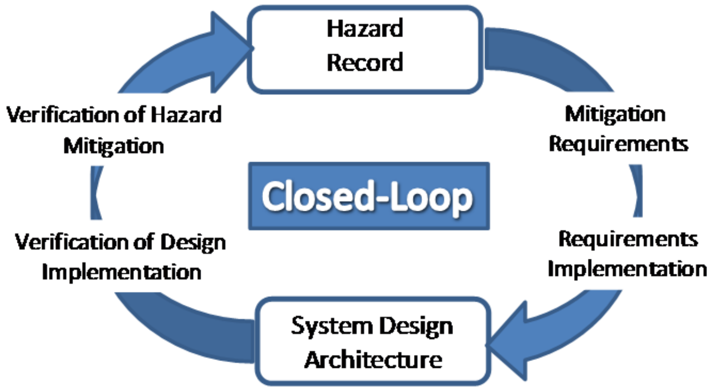

Hazard Record
Mitigation Requirements
Requirements Implementation
System Design Architecture
Verification of Design Implementation
Verification of Hazard Mitigation
Closed-Loop

Figure 3.4: Hazard Closed-Loop Requirements Traceability

3.8.2. Process Subtask 8.2: Trace Safety Requirements to Hazards

To complete the closed-loop hazard mitigation process, safety requirements are traced from the design and their verification back to the hazard/mishap record to formally provide the evidence of hazard mitigation and control. This traceability flows from the system design architecture and the requirements verification back to the hazard/mishap to confirm the mitigation and control of hazard causal factors (refer to Figure 3.4).

3.8.3. Process Subtask 8.3: Trace Safety Requirements to Implementation

The traceability of safety requirements to the design architecture must include the implementation within the software code. This implementation will later be verified in accordance with the approved Software Requirements Verification Matrix (SRVM).

3.9. Process Task 9.0: Perform Code-Level Safety Analysis

[Ref: JSSSEH paragraph 4.3.7.3.2]

As depicted in Figures 3.3 and 4.0 and depending on the LOR assessed, Code-Level Safety Analysis is the next step of the software system safety process. Code-level safety analysis is required and called out in the LOR table for LOR-1 software.

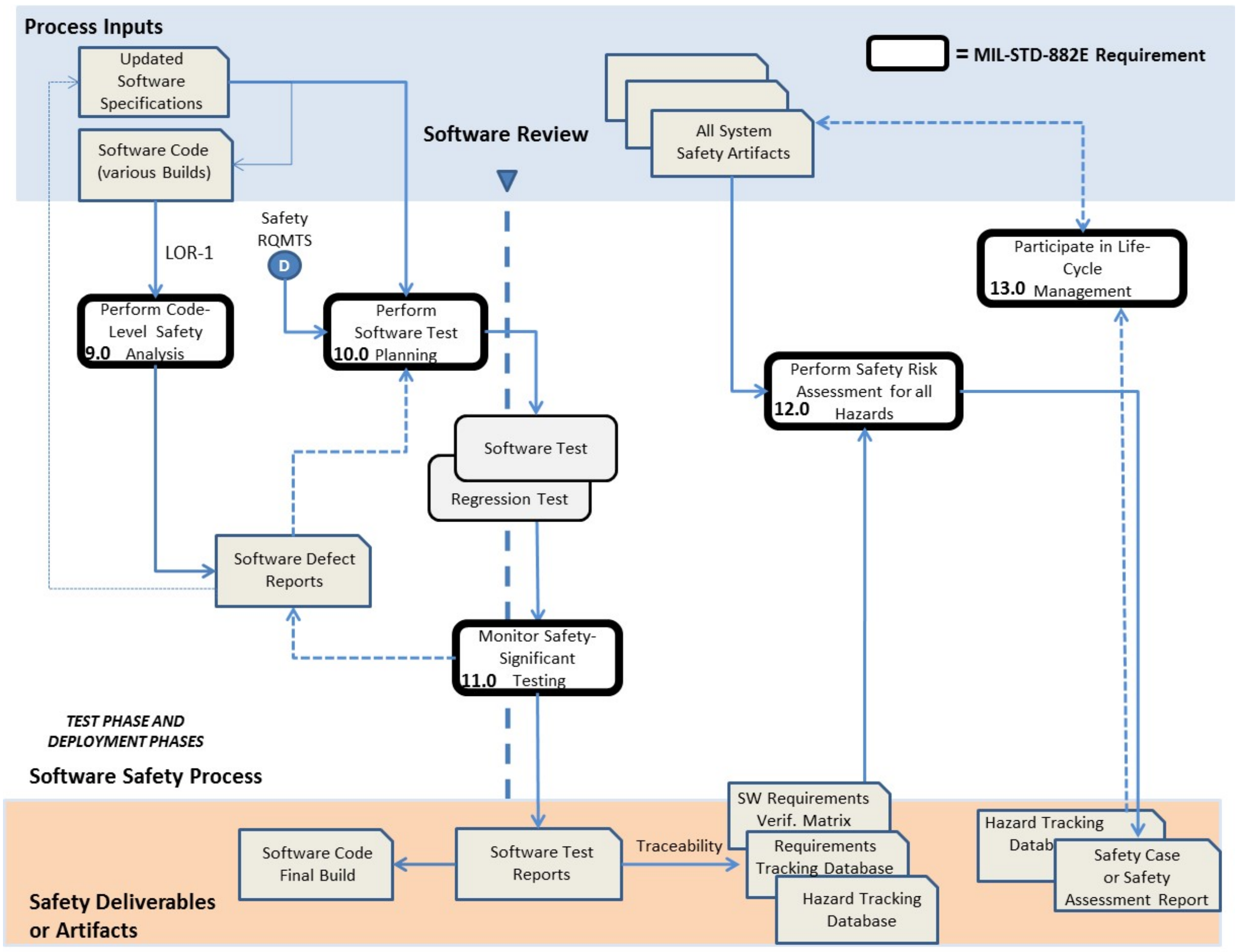

Process Inputs
= MIL-STD-882E Requirement
Updated Software Specifications
Software Code (various Builds)
Software Review
All System Safety Artifacts
Safety RQMTS
D
LOR-1
Participate in Life-Cycle 13.0 Management
Perform Code-Level Safety 9.0 Analysis
Perform Software Test 10.0 Planning
Perform Safety Risk Assessment for all 12.0 Hazards
Software Test
Regression Test
Software Defect Reports
Monitor Safety-Significant 11.0 Testing
TEST PHASE AND DEPLOYMENT PHASES
Software Safety Process
SW Requirements Verif. Matrix
Hazard Tracking Database
Traceability
Requirements Tracking Database
Safety Case or Safety Assessment Report
Software Code Final Build
Software Test Reports
Hazard Tracking Database
Safety Deliverables or Artifacts

Figure 4.0: SSS Process Chart for Test and Deployment Phases

3.9.1. Process Subtask 9.1: Determine the Software Functionality to Analyze

Figure 4.1 depicts the process subtasks for the selection and implementation of the code-level safety analysis. Software assessed as LOR-1 should be evaluated and assessed using the code-level safety analysis technique.

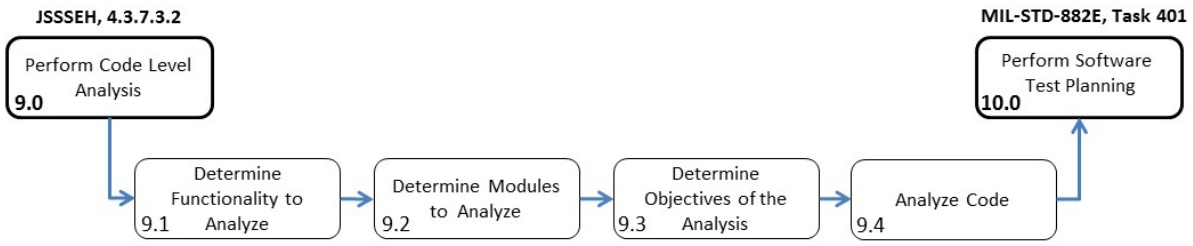

JSSSEH, 4.3.7.3.2
MIL-STD-882E, Task 401
Perform Code Level Analysis 9.0
Perform Software Test Planning 10.0
Determine Functionality to Analyze 9.1
Determine Modules to Analyze 9.2
Determine Objectives of the Analysis 9.3
Analyze Code 9.4

Figure 4.1: Process Task 9.0 Perform Code Level Analysis and Subtasks

3.9.2. Process Subtask 9.2: Determine the Software Modules to Analyze

Software modules (or use cases) that implement the functionality identified in Process Subtask 9.1 must be identified and tagged for analysis.

3.9.3. Process Subtask 9.3: Determine the Objectives of the Analysis

Before the safety-code level analysis begins, the SSS analyst and the assisting Subject Matter Experts (SMEs) determine the specific objectives that are required to be fulfilled by the code-level analysis. Objectives of the analysis should consider the operational context of the system/software to determine values, data ranges, etc. to use in the analysis. Examples of specific objectives that may be fulfilled by the accomplishment of a safety-code-level analysis include, but are not limited to:

- Specification to code tracing.
- Complex logic accuracy.
- Equation and algorithm accuracy.
- Fault and exemption handling.
- Forward or backward logic tracing.
- Safety-significant requirements implementation (compatible with architecture, models, is verifiable, conforms to standards, complies with requirements).
- Safety-significant data handling.
- Effects of concurrent processing.
- Accuracy and integrity of external file structures.
- Integrity of lower-level functional interfaces.
- Off-nominal inputs from functional or physical interfaces.

3.9.4. Process Subtask 9.4: Analyze LOR-1 Software

Upon identification and documentation of the objectives to be accomplished, the code level analysis review must be scheduled and conducted. Any errors or software deficiencies discovered in the safety code-level analysis review, to include errors identified in safety-significant code that is not specifically part of the analysis, must be formally documented and submitted to the software development team for defect resolution. Specific questions to be answered include, but are not limited to:

- Is the code uniquely identified as such in the module header?
- Is the intended functionality of the software coded correctly?
- Have the requirements been correctly interpreted and coded?
- Is the timing and sequencing of the functionality correct?
- Is the logic for functionality accurate and as simple as necessary?
- Have all nominal and off nominal inputs been accounted for?
- Are variables or file structures adequately protected?
- Have fault and exception handling been adequately considered and implemented?
- Does the code contain any dead or unused code, or unintended functionality?
- Is this code influenced by concurrent processing?

- Are safety significant software programs and functions implemented and executed on multi-core processors?
- Are data races or shared data issues detected to prevent the corruption of safety-critical data or variables?
- Can corrupted data lead to incorrect decisions by safety-critical software?
- Can mutual exclusion deadlocks freeze autonomous software control over safety-critical functionality or processing?
- Are the code's interfaces with other code and modules compatible?
- Are the functions and code isolated and/or partitioned from non-safety code where required?

3.10. Process Task 10.0: Perform Software Test Planning

[Ref: MIL-STD-882E Tasks 303 and 401, JSSSEH paragraph 4.4.1 and Appendix A – LOR Table]

Software test planning from a safety perspective actually begins during Process Task 2.0 when the LOR task table is defined, documented, and agreed upon by the SSS, software development and software test teams. It is best practice of software test teams to test each software requirement that is documented in the software requirements specification. At a minimum, all SRS safety requirements must be reviewed to ensure the implementation complies with safety design requirements and mapped to test cases.

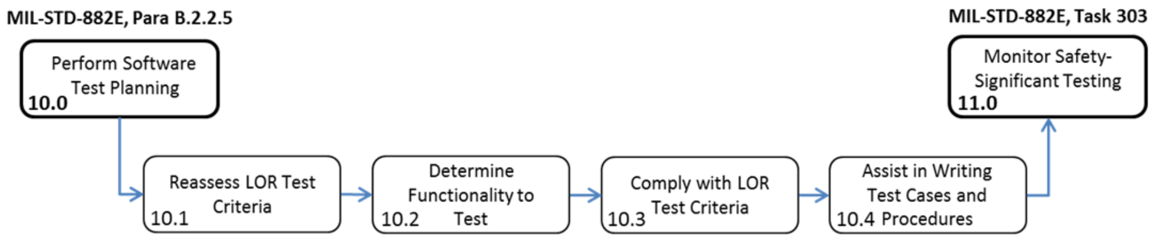

MIL-STD-882E, Para B.2.2.5
Perform Software Test Planning 10.0
Reassess LOR Test Criteria 10.1
Determine Functionality to Test 10.2
Comply with LOR Test Criteria 10.3
Assist in Writing Test Cases and Procedures 10.4
MIL-STD-882E, Task 303
Monitor Safety-Significant Testing 11.0

Figure 4.2: Process Task 10.0 Perform Software Test Planning and Subtasks

3.10.1. Process Subtask 10.1: Ensure Correctness and Application of the LOR Test Criteria

A portion of the software test planning for the program has been accomplished with the fulfillment of Process Task 2.0 when defining the LOR tasks for each phase of the software development and test life cycle. The LOR Table provides specific software test tasks for each LOR allocation. At this specific point in time, the SSS reassesses the LOR software test tasks to ensure that they are still relevant and to verify that the tasks have been accurately accounted for in the STP.

3.10.2. Process Subtask 10.2: Ensure Safety Functionality is Tested

All safety-significant SRS requirements should be tested. Safety-significant requirements should be formally documented within the SRS as accounted for in Process Task 7.0, Finalize SRHA. SSS can assist the software test team in developing specific test cases and procedures to ensure that each SSF is exercised and tested IAW its LOR. The testing should demonstrate that the software functions as it is expected to function in both nominal and off-nominal operations and environments.

3.10.3. Process Subtask 10.3: Comply with the LOR Test Criteria

LOR test requirements, as defined in the program's LOR table, must be specifically adhered to for the purpose of increasing the confidence that the software does not possess unnecessary or undocumented safety risk potential. At this point in time, it is the responsibility of SSS to verify that the software testing is conducted in accordance with the criteria as documented.

3.10.4. Process Subtask 10.4: Assist in Writing Test Cases and Test Procedures

SSS should assist in the test case and procedure development for safety-significant Computer Software Configuration Item (CSCIs), Computer Software Component (CSCs) and Computer Software Unit (CSUs). SSS should possess insight as to how the software should perform functionally and what the software should be prohibited from doing. Specific safety test criteria for consideration when writing test cases and test procedures can be found in the program's LOR Table.

3.11. Process Task 11.0: Monitor Safety-Significant Software Testing

It is SSS's responsibility to monitor the software testing of SSFs. Monitoring should also include notifying and inviting the Customer to witness testing. "Monitoring" can come in the form of participation or witnessing test events, or from the review of test results and Software Quality Assurance (SQA) sign-offs. The test objectives and test criteria must be fulfilled by the test activity in accordance with the STP and the LOR Table's design assurance criteria as depicted in the Process Subtasks of Figure 4.3.

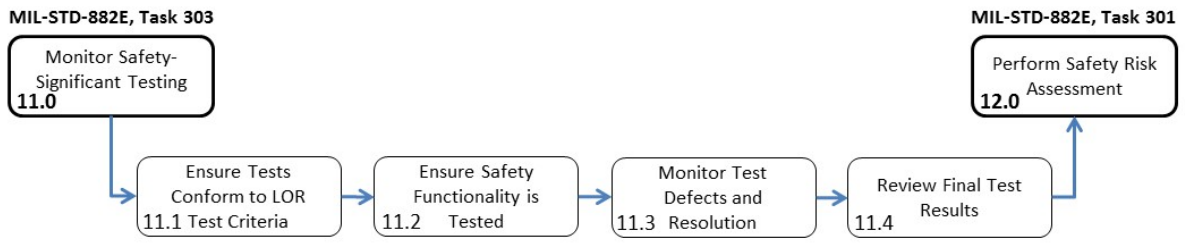

MIL-STD-882E, Task 303
Monitor Safety-Significant Testing
11.0
MIL-STD-882E, Task 301
Perform Safety Risk Assessment
12.0
Ensure Tests Conform to LOR Test Criteria
11.1
Ensure Safety Functionality is Tested
11.2
Monitor Test Defects and Resolution
11.3
Review Final Test Results
11.4

Figure 4.3: Process Task 11.0 Monitor Safety-Related Testing and Subtasks

3.11.1. Process Subtask 11.1: Ensure Software Testing Conforms to LOR Test Criteria

Specific software test criteria have been established in the LOR task table. The software test activities must be accomplished in accordance with this established LOR criteria. LOR test criteria that are not fulfilled must be formally documented and accounted for in any safety risk assessment that is accomplished for the program, IAW MIL-STD-882E criteria. MIL-STD-882E, Table VI, provides requirements for documenting potential contributions to system level risk associated with LOR shortfalls (i.e., if LOR tasks are unspecified or incomplete).

3.11.2. Process Subtask 11.2: Ensure Safety Functionality is Tested

As defined in Process Subtask 11.1, SSS ensures that all safety-significant functionality is adequately tested in accordance with LOR criterion. Specific test objectives for safety-significant functionality should include such criteria as:

- Software performs the function as intended and produces the expected outcome.
- Software performs the function in its intended time allocation and within its defined sequence.
- Software does not perform undocumented, undefined, and unintended functions.
- Software performs as expected in normal or nominal environments and conditions.
- Software performs as expected in off-nominal environments and conditions.
- Software can detect faults/failures of safety-significance.
- Software can isolate faults/failures to minimize the propagation of faults/failures to the system.
- Software can annunciate fault/failures to appropriate control entity responsible for recovery action.
- Software can take appropriate autonomous recovery action (if there is a requirement) to defined faults/failures.
- Functional, physical and human interfaces to ensure they are under positive control.

3.11.3. Process Subtask 11.3: Monitor Test Defects and Corrective Actions

As the software testing commences, SSS must monitor the testing accomplished (unit testing, integration testing, and Formal Qualification Testing (FQT)) on safety-significant elements of the software design architecture. "Monitoring" can be in the form of reviewing test cases, procedures, and results or witnessing software test events themselves. Any failures, anomalous conditions, causal factors, or hazards identified in software test must be documented, and tracked to a suitable solution or corrective action. Defect resolution, or changes made to correct software deficiencies must be accomplished in accordance with the Software Configuration Management Plan.

3.11.4. Process Subtask 11.4: Review Final Software Test Results

Upon the completion of defined software test cases and procedures, the SSS must review the software test reports. The review of the software test report should confirm:

- The software test case is accomplished in accordance with the test procedure.
- The software test results verify the successful implementation of safety requirements.
- The software test results verify the adequate mitigation of hazards and hazard causal factors.
- The software test anomalies and defects are adequately identified, documented, and rectified.
- The software defect resolutions are adequately regression tested.
- Traceability from test reports/results to hazards and mitigations is performed and evidenced.

3.12. Process Task 12.0: Perform Safety Risk Assessment

[Ref: MIL-STD-882E, Task 301, and JSSSEH paragraph 4.4.4]

As the SSS process is implemented, the conclusion of software testing will usually bring the program to a point in time where "influencing the design" is also concluded. The remaining options for hazard mitigation or control in our "order of precedence" are procedures and training for operators and maintainers.

SSS must support the system safety requirements to document safety risk as depicted in Figure 4.4.

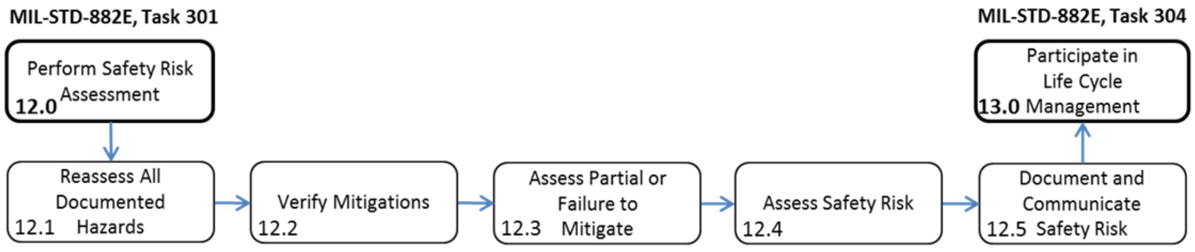

MIL-STD-882E, Task 301
Perform Safety Risk Assessment
12.0
MIL-STD-882E, Task 304
Participate in Life Cycle Management
13.0
Reassess All Documented Hazards
12.1
Verify Mitigations
12.2
Assess Partial or Failure to Mitigate
12.3
Assess Safety Risk
12.4
Document and Communicate Safety Risk
12.5

Figure 4.4: Process Task 12.0 Perform Safety Risk Assessment and Subtasks

3.12.1. Process Subtask 12.1: Reassess all Documented Hazards

The safety risk assessment update after the completion of verification activities begins with a comprehensive assessment of each documented mishap and hazard in the HTS. This assessment is a confirmation that the hazard records contain complete and accurate information.

3.12.2. Process Subtask 12.2: Verify Hazard Mitigation

As the HTS records are being assessed for hazard mitigation verification, the SSS must verify that documented hazard mitigations have been adequately and accurately documented within the HTS records. The evidence pertaining to the successful implementation of these safety requirements becomes the necessary evidence for mishap and hazard mitigation.

3.12.3. Process Subtask 12.3: Assess Partial Mitigation or Failure to Mitigate

In the process of verifying the successful implementation of safety requirements, SSS may discover that some safety requirements were only partially implemented, deferred to later software builds, or completely rejected. Partial or no implementation of safety requirements (including hardware, software, and human action requirements) for a given mishap or hazard equates to safety risk. The amount of safety risk will be dependent on other factors that must be considered, such as:

- The severity of the mishap or hazard occurrence.
- The number of other hazard mitigations implemented.
- The Boolean relationship of other mitigations with the mitigation that was not implemented.
- The ability (or inability) of the system to detect, isolate, and recover from a failure should the failure occur.
- Body of Evidence (i.e., the results of LOR task implementation) to meet the LOR specified in MIL-STD-882E, Table V.

Software safety requirements that are not implemented must remain in in the software requirements and CM documentation and be prioritized for the next software build or Engineering Change Proposal (ECP) that occurs.

3.12.4. Process Subtask 12.4: Assess Safety Risk

Safety risk assessment is a comprehensive evaluation of the mishap risk that must be accepted prior to test or operation of the system and exposing people, equipment, or the environment to known hazards. This process subtask determines the safety risk that must be accepted IAW DoDI 5000.02.

3.12.5. Process Subtask 12.5: Document and Communicate Safety Risk

The results and conclusions of the safety risk evaluation are formally documented within the SAR. System Safety Risk Assessments (SSRAs), and corresponding risk acceptance must also be performed, as applicable.

3.13. Process Task 13.0: Participate in Life-Cycle Management and Support

[Ref: MIL-STD-882E, Task 304, and JSSSEH paragraph C.11)

Modifications or changes to the system are likely to occur multiple times before the system is decommissioned and taken out of service. Changes are either the correction of defects and deficiencies identified by the system user or maintainer, or the functional or physical upgrade of the system to enhance operational effectiveness and suitability. The latter can even be the result of the redefinition of the mission that the system is to accomplish. Regardless of the reason for change, the SSS program must be prepared for change and to accomplish the process tasks regarding change as depicted in Figure 4.5. Detailed Life-cycle Management tasks are also found in Appendix A, Life-cycle Support Phase Tasks.

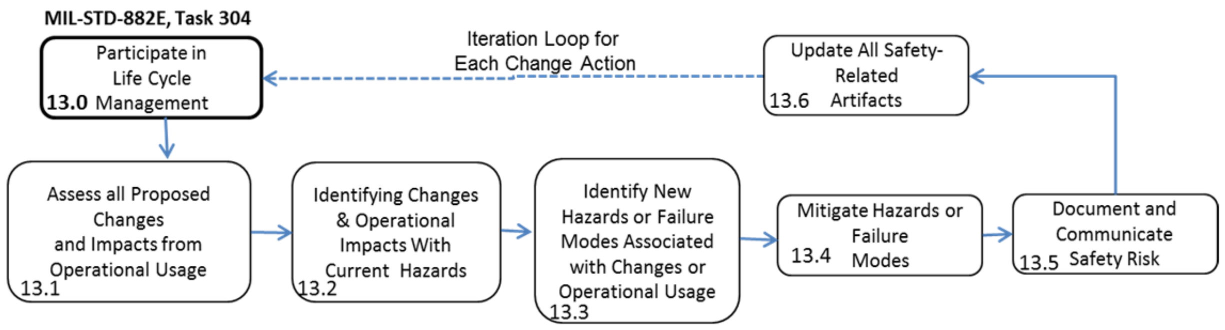

MIL-STD-882E, Task 304
13.0 Participate in Life Cycle Management
Iteration Loop for Each Change Action
13.6 Update All Safety-Related Artifacts
13.1 Assess all Proposed Changes and Impacts from Operational Usage
13.2 Identifying Changes & Operational Impacts With Current Hazards
13.3 Identify New Hazards or Failure Modes Associated with Changes or Operational Usage
13.4 Mitigate Hazards or Failure Modes
13.5 Document and Communicate Safety Risk

Figure 4.5: Process Task 13.0 Participate in Life Cycle Management and Subtasks

To actively participate in a product's life cycle management, SSS must be familiar with, and an active participant, in the configuration management process.

3.13.1. Process Subtask 13.1: Assess all Proposed Changes to and Operational Impacts on the System

At a minimum, System Safety should be a member of the Configuration Control Board (CCB) with signature authority on ECP actions or upgrades. SSS must review every change request pertaining to the system software and provide input to the System Safety representative on the CCB. Additionally, as results from Operations of the system are obtained, any impacts of Operational usage must be assessed for safety impact.

3.13.2. Process Subtask 13.2: Identify Changes and Operational Impacts Associated with Current Hazards

Functional or physical changes to a legacy system will likely affect the status quo of the existing hazard analysis and must be assessed against documented hazards and accepted risks, or for the potential to introduce new mishaps and hazards. Additionally, Operations of the system may reveal previously unidentified safety hazards and/or causes and mitigations of existing hazards.

3.13.3. Process Subtask 13.3: Identify New Hazards, Failure Modes, or Causes Associated with Changes and Operational Usage

The system safety analysis of a change to the system or Operations of the system must determine whether the change or Operations creates a mishap/hazard that did not exist in the legacy system, or has an impact on an existing mishap/hazard. If this is the case, the mishap/hazard must be analyzed to determine how it will be mitigated or controlled to an acceptable level of risk. If new mishaps/hazards are not created by the proposed design change, there is a potential that new failure modes or causes are created for existing hazards of the systems. These new failure modes and causes to existing hazards must also be addressed, to include re-visiting accepted safety risks.

3.13.4. Process Subtask 13.4: Mitigate Hazards, Failure Modes, or Causes

Mishaps and hazards, failure modes, and causal factors identified by the safety analysis for the proposed system change(s) and Operations must be adjudicated just as any hazard identified during system development. Mitigation is not complete until the modified software functionality has been analyzed and tested (including regression testing) IAW its LOR.

3.13.5. Process Subtask 13.5: Document and Communicate Safety Risk

As change requests are processed, approved, analyzed, and implemented, all safety analyses must be accomplished for the purpose of reducing safety risk potential to the greatest extent possible (or practical). Upon the completion of the system safety and SSS engineering tasks, a safety risk assessment is performed and documented. New or updated safety risks must be accepted in accordance with DoDI 5000.02.

3.13.6. Process Subtask 13.6: Update all Safety-Related Artifacts

Upon the completion of system safety engineering and management tasks associated with a change, all system and system safety related artifacts must be updated to account for the change and its ultimate safety risk potential. For any given change action, the following engineering artifacts should be considered candidates for update:

- SSPP (if there were any changes to management or engineering processes, tasks, budgets, or schedules).
- Hazard Analysis (analyses accomplished to date, e.g., PHA, FHA, SSHA, SHA, O&SHA).
- In Depth Causal Analysis (e.g., FTA, Failure Mode, Effects and Criticality Analysis (FMECA)).
- SRHA (all safety requirements artifacts to include updates to the SRS, SDP, or STP as required).

- SAR (or possibly Safety Case to account for the safety risk assessment).
- HTS (to account for all hazard analysis record keeping to include hazard mitigation and/or control).

Updating the artifacts related to safety produces the necessary evidence of hazard identification, documentation, categorization, and mitigation for those organizations and personnel operating, maintaining, and supporting the legacy system.

4.0 Acronym List

ARP	Aerospace Recommended Practice
AWR	Air Worthiness Release
CCB	Configuration Control Board
CDRL	Contract Data Requirements List
CM SEI	Carnegie Mellon Software Engineering Institute
CMP	Configuration Management Plan
CONOPS	Concept of Operations
COTS	Commercial-off-the-Shelf
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSSR	Contributing Safety-Significant Requirement
CSU	Computer Software Unit
DID	Data Item Description
DoD	Department of Defense
ECP	Engineering Change Proposal
FAA	Federal Aviation Administration
FHA	Functional Hazard Analysis
FMECA	Failure Mode, Effects and Criticality Analysis
FQT	Formal Qualification Testing
FTA	Fault Tree Analysis
GSSR	Generic Safety-Significant Requirement
HHA	Health Hazard Analysis
HTS	Hazard Tracking System (database)
IAW	In Accordance With
IPT	Integrated Product Team
JSSSEH	DoD Joint Software Systems Safety Engineering Handbook
LOR	Level-of-Rigor
MISRA	Motor Industry Software Reliability Association
MSSR	Mitigating Safety-Significant Requirement
O&SHA	Operating and Support Hazard Analysis
OOD	Object Oriented Design
PHA	Preliminary Hazard Analysis
PHL	Preliminary Hazard List
PM	Program Manager

RAC	Risk Assessment Code
RFP	Request for Proposal
ROI	Return-On-Investment
SAR	Safety Assessment Report
SCC	Software Control Category
SCF	Safety Critical Function
SCM	Software Criticality Matrix
SCMP	Software Configuration Management Plan
SDP	Software Development Plan
SEMP	System Engineering Master Plan
SHA	System Hazard Analysis
SME	Subject Matter Expert
SoS	System-of-Systems
SOW	Statement of Work
SQAP	Software Quality Assurance Plan
SRA	Safety Requirements Analysis
SRF	Safety-Related Function
SRHA	System Requirements Hazard Analysis
SRS	Software Requirements Specification
SRVM	Software Requirements Verification Matrix
SSCM	Software Safety Criticality Matrix
SSE	System Safety Engineering
SSF	Safety-Significant Function
SSHA	Sub-System Hazard Analysis
SSMP	System Safety Management Plan
SSPP	System Safety Program Plan
SSR	System Safety Requirement
SSRA	System Safety Risk Assessment
SSS	Software System Safety
SSSE	Software System Safety Engineering
SSWG	System Safety Working Group
STP	Software Test Plan
STR	Software Trouble Report
SwCI	Software Criticality Index
SwSSP	Software System Safety Program
SwSSPP	Software System Safety Program Plan
TEMP	Test and Evaluation Master Plan
TRR	Test Readiness Review
V&V	Verification and Validation

5.0 Glossary

Acceptance Criteria – Criteria that a system, software build, or component must satisfy in order to be accepted by an Acquirer, acceptance authority, or a certification authority.

Acquirer – Stakeholder that acquires or procures a product or service from a supplier. The Acquirer may be one of the following: buyer, customer, owner, or purchaser.

Baseline – Specification or product that has been formally reviewed and agreed upon that thereafter serves as the basis for further development and that can be changed only through formal change management procedures.

Causal Factors – (1) The particular and unique set of circumstances that can contribute to a hazard. (2) The combined hazard sources and initiating mechanisms that may be the direct result of a combination of failures, malfunctions, external events, environmental effects, errors, inadequate design, or poor judgment.

Contributing Safety-Significant Requirements – A subcategory of the defined safety requirements of a system. CSSRs are requirements contained within the specifications that contribute to the safety risk potential of a system by the functionality that they will perform. CSSRs do not mitigate risk.

Control Entity – The specific entity that provides autonomous, semi-autonomous, or responsive situational awareness command or control authority over unmanned system functionality. The entity may be human, software logic, or the logic programmed into firmware or programmable logic devices.

Developer – A private or government enterprise or organizational element engaged to provide services or products within agreed limits specified by the Acquirer.

Failure – The inability of an item to perform its intended function.

Failure Mode – A term used to describe one (of possibly many) mechanisms that could contribute to failure. In context to a hazard, the failure modes are descriptors of the overall mechanisms that could lead to a hazards existence. Individual failure modes consist of causal factors, causal pathways, and pathway initiation events.

Firmware – The combination of a hardware device and computer instructions and/or computer data that resides as read-only software on the hardware device.

Function – A task, action, or activity that must be performed to achieve a desired outcome.

Generic Safety-Significant Requirements – A subcategory of the defined safety requirements of a system. GSSRs are a product of documented system development, safety best practices, and lessons learned from legacy programs.

Level-of-Rigor – A specification of the depth and breadth of software analysis, test, and verification activities necessary to provide a sufficient level of confidence that a safety significant software function will perform as required.

Mishap – An unplanned event or series of events resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment.

Mishap Probability – The aggregate probability of occurrence of the individual events or hazards that might create a specific mishap.

Mishap Risk – An expression of the impact and probability of a mishap in terms of potential mishap severity and probability of occurrence.

Mishap Severity – An assessment of the consequences of the most reasonable credible mishap that could be caused a specific hazard or combination of hazards.

Mitigating Safety-Significant Requirements – A subcategory of the defined safety requirements of a system. MSSRs are normally identified during in-depth mishap and hazard causal analysis and are derived for the purpose of mitigating or controlling failure pathways to the mishap or hazard.

Qualification Testing – Testing conducted to determine whether a system or component is suitable for operational testing.

Regression Testing – The testing of software to confirm that functions that were previously performed correctly continue to perform correctly after a change has been made.

Requirement – (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2).

Safety Critical – A term applied to a condition, event, function, operation, process, or item of whose mishap severity consequence is determined to be either Catastrophic or Critical by definition.

Safety Requirements Analysis – An analysis which identifies, categorizes, prioritizes, and justifies the safety requirements to be implemented on a system to influence the design of that system from a safety perspective.

Safety Related – A term applied to a condition, event, function, operation, process, or item of whose mishap severity consequence is determined to be either Marginal or Negligible (less than critical) by definition.

Safety Significant - A term applied to a condition, event, function, operation, process, or item that possesses a mishap or hazard severity consequence by definition. That which is defined as safety-significant can either be safety-critical or safety-related.

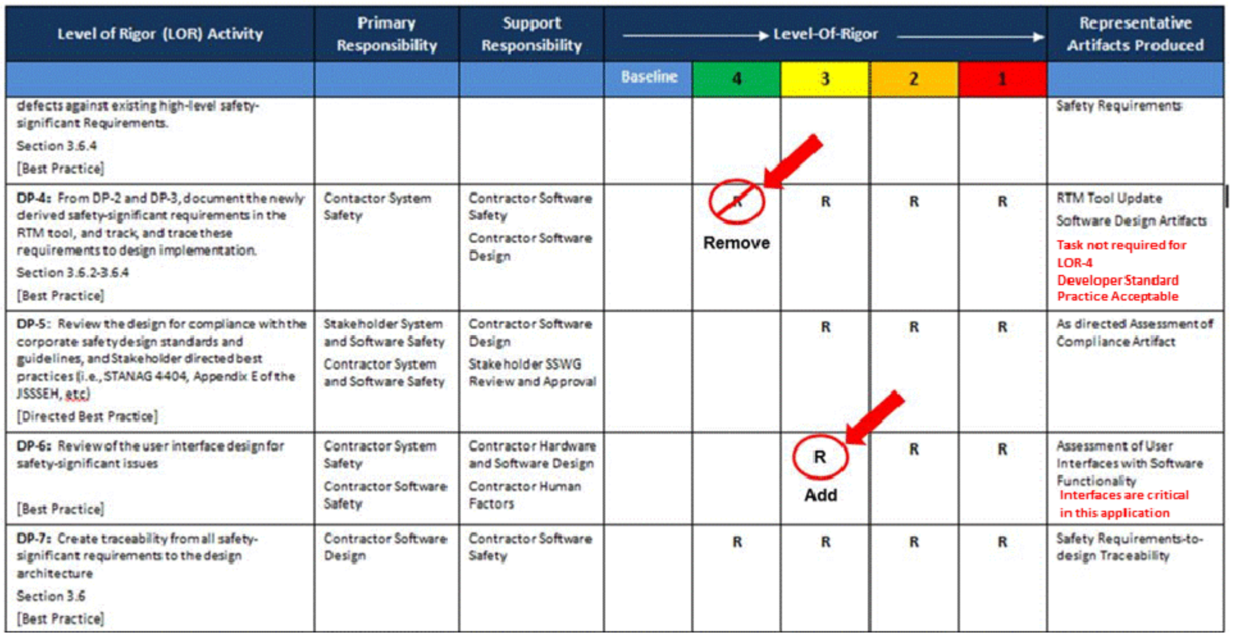
Validation – The determination that the requirements for a product are sufficiently correct and complete.

Verification – The evaluation of an implementation of requirements to determine that they have been met.

Appendix A

Preferred Level of Rigor Activities Table

The Level-of-Rigor (LOR) task table formally defines the software safety process tasks, software development and test tasks, and special design criteria required to fulfill the requirements of MIL-STD-882E, Table V. It is essential that the LOR tasks defined and contractually required on each program make logical and economic sense from a both a safety risk and return-on-investment perspective. In addition, it is important that the tasks defined are fully integrated into the standard practice processes of both system safety engineering and software development and test processes. With this in mind, the Acquirer and the Developer may tailor the preferred tasks provided in this implementation guide. Figure A.1 provides a graphical representation of the recommended method of tailoring the task table.



Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	→ Level-Of-Rigor →					Representative Artifacts Produced
			Baseline	4	3	2	1	
defects against existing high-level safety-significant Requirements. Section 3.6.4 [Best Practice]								Safety Requirements
DP-4: From DP-2 and DP-3, document the newly derived safety-significant requirements in the RTM tool, and track, and trace these requirements to design implementation. Section 3.6.2-3.6.4 [Best Practice]	Contactor System Safety	Contractor Software Safety Contractor Software Design		~~R~~ Remove	R	R	R	RTM Tool Update Software Design Artifacts Task not required for LOR-4 Developer Standard Practice Acceptable
DP-5: Review the design for compliance with the corporate safety design standards and guidelines, and Stakeholder directed best practices (i.e., STANAG 4404, Appendix E of the JSSSEH, etc) [Directed Best Practice]	Stakeholder System and Software Safety Contractor System and Software Safety	Contractor Software Design Stakeholder SSWG Review and Approval			R	R	R	As directed Assessment of Compliance Artifact
DP-6: Review of the user interface design for safety-significant issues [Best Practice]	Contractor System Safety Contractor Software Safety	Contractor Hardware and Software Design Contractor Human Factors			R Add	R	R	Assessment of User Interfaces with Software Functionality Interfaces are critical in this application
DP-7: Create traceability from all safety-significant requirements to the design architecture Section 3.6 [Best Practice]	Contractor Software Design	Contractor Software Safety		R	R	R	R	Safety Requirements-to-design Traceability

Figure A.1: Preferred LOR Tailoring Method Example

It is recommended that tailoring consists of making changes to the columns of the table to determine whether a specific task will be required for a given LOR level as depicted in the figure. It is also recommended making word changes to the actual tasks themselves not be accomplished. The tasks themselves have been formally documented and peer reviewed as "best practices" and should only be tailored as a last resort where special circumstances warrant the change. As an example, LOR Task RP-8 states: "Coordinated Safety-significant Requirements Review for correctness and completeness." This could be tailored to state that only Safety-Critical requirements be reviewed for correctness and completeness. The documented rationale for this tailoring may be that the system possesses an extraordinary number of safety-related requirements and the rapid acquisition budget and schedule does not warrant the accomplishment of the task for lower severity level requirements.

As a reminder, all LOR tailoring of tasks and the rationale for LOR tailoring must be reviewed and approved by the Acquirer to ensure that the intent of the LOR activity meets the intent of MIL-STD-882E and any acceptance authority.

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR | **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation | **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Required System Safety Tasks to Support Software System Safety Per MIL-STD-882E								
ACQ-1: Document the Acquirer's plans and processes to meet the requirements of the System Safety and Software System Safety programs. Section 3.0 Process and Tasks for Software System Safety	Acquirer PM Acquirer Systems Engineering (SE) Acquirer System Safety Engineering (SSE)	Acquirer Software System Safety (SSS) Service Safety Review Authorities	**PR**					Acquirer System Safety Management Plan (SSMP)
ACQ-1.1: Specify Mishap Risk Matrix, SwCM, Hazard Tracking System (HTS) and charter SSWG. Obtain approval from DoD Component for any tailoring of MIL-STD-882E matrices.	Acquirer PM Acquirer SE Acquirer SSE	Acquirer SSS Service Safety Review Authorities	**PR**					Acquirer SSMP
ACQ-1.2: Specify the baseline LOR Task Table to be used and/or tailored for the program	Acquirer SSS	Acquirer Team	**PR**					Acquire SSMP
ACQ-2: Develop safety input to Request for Proposal (RFP), Statement of Work (SOW) and other contractual documentation (tasks, analyses, CDRLs, etc.) Section 3.1.6 Provide Inputs to the RFP and SOW	Acquirer SSE Acquirer SSS	Acquirer SE	**PR**					RFP, SOW, contract
ACQ-3: Define Acquirer specification safety requirements	Acquirer SSE Acquirer SSS	Acquirer SE	**PR**					Govt. System Specification, CDD
ACQ-4: Ensure safety is a member of the configuration control process (voting member of Acquirer chaired boards)	Acquirer SSE Acquirer SSS	Acquirer CM	**PR**					CM Plan and Charters
ACQ-5: Initiate hazard analyses	Acquirer SSE Acquirer SSS	Acquirer Team	**AD**					Acquirer safety artifacts
SSE-1: Document the Developer plans and processes to meet the requirements of the System Safety and Software System Safety programs.	Developer System Safety Manager Developer Software Safety	Developer Program Manager Developer Hardware and Software Design Engineering Developer Software	**PR**					System Safety Program Plan (SSPP) and Software System Safety Program Plan (SwSSPP). SOW, CDRL Acquirer Approved SSPP/SwSSPP

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR — **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation — **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Section 3.0 Process and Process Tasks for Software System Safety MIL-STD-882E, Task 102		Design Architect Developer Configuration Management						
SSE-1.1: Define the safety terms (and the definitions) to be used on the program, to include deviations from MIL-STD-882E. Section 3.1, Prepare the SSPP; Subsections 3.1.1 - 3.1.5 [Best Practice]	Acquirer System Safety Manager Developer System Safety Manager Developer Software Design Architect	Developer System Safety Acquirer SSWG Review and Approval	**PR**					Documented Program-Specific Terms and Definitions. MIL-STD-882E definitions and terms are required unless approved by appropriate authorities Acquirer Approved SSPP
SSE-1.2: Detail within the SSPP/ SwSSPP, how the SwSS tasks will be accomplished within the specific software development life-cycle for the project. Section 3.2.3 Integrate Software Safety Engineering Criteria MIL-STD-882E, Task 102	Developer Software Safety Developer Software Development	Acquirer SSWG Review and Approval	**PR**					SOW, CDRL. SSPP/SwSSPP Acquirer Approved SSPP/SwSSPP
SSE-1.3: Develop safety entry/exit criteria for each program phase of the software development life cycle to include concept refinement, requirements, preliminary and detailed design, coding, Test Verification and Validation (V&V), software release and support). [Best Practice]	Developer Software Safety Developer Software Development and Test Configuration Mgmt	Acquirer SSWG Review and Approval	**PR**					Input to SwSSPP Input to Software Development Plan (SDP) Input to Software Test Plan (STP) Input to CMP Input to SwQAP SSWG Minutes
SSE-1.4: Document the Software Control Category (SCC) Definitions to be used on the program Section 3.2 Prepare the SSPP [MIL-STD-882E, Table IV]	Developer Software Safety Developer Software Development and Test Developer Software Design Architect	Acquirer SSWG Review and Approval	**PR**					Defined Software Control Category Definitions SwSSPP
SSE-1.5: Document the Software Criticality Matrix (SCM) for the program	Acquirer System Safety Manager	Acquirer SSWG Review and Approval	**PR**					SSMP Program Software Criticality

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR — **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation — **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Section 3.2 Prepare the SSPP [MIL-STD-882E, Table V]	Developer Software Safety Developer Software Development and Test Developer Software Design Architect							Matrix SwSSPP SSWG Minutes
SSE-1.6: Develop (or update) Level of Rigor (LOR) task table for the program to include tasks and work products for each LOR software development phase Section 3.2.4 Prepare Level-of-Rigor Table [MIL-STD-882E, Table V]	Developer Software Safety Developer Software Development and Test Developer Software Design Architect	Acquirer SSWG Review and Approval	**PR**					LOR Table SwSSPP SSWG Minutes
SSE-2: Support the System Safety Working Group (SSWG) Section 3.0 Process and Tasks for Software System Safety	Acquirer System Safety Manager Developer System Safety, Software Design Architect, Software Safety, Software Development & Test	Acquirer SSWG	**PR**					SSMP, SSPP. SSWG Charter and Proceedings
SSE-3: Set-up a Hazard Tracking System (HTS) for the program Section 3.2.1 Obtain Inputs from the SSMP Also, Statement of Work tasks [MIL-STD-882E, Paragraph 4.3.2]	Developer System Safety Acquirer System Safety	Acquirer SSWG Review and Approval	**PR, AD**					Hazard Tracking Database SSWG Minutes
SSE-4: Perform a Preliminary Hazard Analysis (PHA) to identify the safety mishaps and hazards and safety mitigation requirements Section 3.3 Preliminary Hazard Analysis (PHA) [MIL-STD-882E, Task 202]	Developer System Safety	Acquirer SSWG Review and Approval	**PR**					List of System Level Mishaps List of Hazards and Hazard Failure Modes Preliminary Hazard Analysis (PHA) SSWG Minutes

Legend:

PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR

ACQ: Task Requirement Performed by the Acquirer

R: Required for assigned LOR | **AD**: As directed by Customer/Contract

IV&V: Independent Verification and Validation | **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
SSE-4.1: Enter each hazard identified into the HTS Section 3.3.1 Identify Hazards Pertaining to the Baseline System to Include the Preliminary System/Software Architecture [MIL-STD-882E, Paragraph 4.3.2]	Developer System Safety	Acquirer SSWG Review and Approval	**PR**					Individual Hazard Records of the HTS SSWG Minutes
SSE-4.2: Assign the severity and probability of occurrence to each hazard identified and calculate the initial Risk Assessment Code (RAC) based on the best available data documented, including provisions, alternatives, and mitigation measures to eliminate hazards or reduce associated risk Section 3.3.5 Categorize Hazards with [MIL-STD-882E, Para 4.3.3]	Developer System Safety	Acquirer SSWG Review and Approval	**PR**					RAC Assignment for each Hazard Record SSWG Minutes
SSE-5: Perform a Functional Hazard Analysis (FHA) to identify the safety-significant functions Section 3.4 FHA [MIL-STD-882E, Task 208]	Developer System Safety	Acquirer SSWG Review and Approval	**PR**					FHA List of Safety-Significant Functions Functional Flow to Subsystems and Software Items. LOR Assignment to Safety-Significant Software Functions SSWG Minutes
SSE-6: Perform a System Requirements Hazard Analysis (SRHA) Section 3.5 Initiate Safety Requirements Hazard Analysis (SRHA), Section 3.7 Finalize Safety Requirements Hazard Analysis [MIL-STD-882E, Task 203]	Developer System Safety	Acquirer SSWG Review and Approval	**PR, AD**					SOW, CDRL, Safety Requirements Analysis (SRA) SSWG Minutes
SSE-7: Perform a System Hazard Analysis (SHA) and accomplish an in-depth causal, interface, and failure mode analysis of the identified hazards to identify specific hardware, software, and human-related causes and the safety mitigating requirements to eliminate or control them. Section 3.6 Perform System and Subsystem Hazard Analyses [MIL-STD-882E, Task 205]	Developer System Safety	Developer Software Safety	**PR**					System Hazard Analysis

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	→ Level-Of-Rigor →					Representative Artifacts Produced
			Baseline	4	3	2	1	
SSE-8: Perform Sub-System Hazard Analysis and accomplish an in-depth causal, interface, and failure mode analysis of the identified hazards to identify specific hardware, software, and human-related causes and the safety mitigating requirements to eliminate or control them. Section 3.6 Perform System and Subsystem Hazard Analyses [MIL-STD-882E, Task 204]	Developer System Safety	Developer Software Safety	**PR, AD**					Subsystem Hazard Analysis for individual subsystems
SSE-9: Perform initial Fault Tree Analysis (FTA)/Event Tree/Logic Diagram on prioritized hazards Section 3. 6 Perform System and Subsystem Hazard Analyses [Best Practice]	Developer System Safety	Developer Software Safety	**PR, AD**					Fault Tree Analysis on prioritized (by SSWG) mishaps or hazards
SSE-10: Perform a System-of-System Hazard Analysis (SoS) to identify unique SoS hazards [MIL-STD-882E, Task 209]	Developer(s) System Safety	Developer(s) Software Safety	**PR, AD**					SoS Hazard Analysis
SSE-11: Perform an Operating and Support Hazard Analysis to identify hazards from the long term operation, maintenance, and support of the application, and to identify mitigating requirements [MIL-STD-882E, Task 206]	Developer System Safety	Developer Software Safety	**PR, AD**					Operating and Support Hazard Analysis (O&SHA)
SSE-12: Review of all Software Trouble Reports for safety applicability to safety-significant functions and mishaps/hazards (STR) Section 3.12.3 Monitor Test Defects and Corrective Actions, 3.12.4 Review Final Software Test Results [MIL-STD-882E, Task 304]	Developer Software Safety	Developer Software Development and Test Developer Software Design Architect	**PR**					STR Review Results **NOTE:** *Refer to subsequent Life-Cycle (LC) Support Tasks required to support sustainment after design is put under Configuration Control*
SSE-13: Produce Safety Case or Safety Assessment Report as directed by the customer (SAR). Ensure the SAR captures all of the relevant SSS elements applicable to the system assessed	Developer System Safety	Developer Software Safety	**PR, AD**					Safety Case Safety Assessment Report SSWG Minutes

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR
AD: As directed by Customer/Contract
IV&V: Independent Verification and Validation
N/A: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Section 3.0 Process and Process Tasks for Software System Safety [MIL-STD-882E, Task 301]								
SSE-14: Maintain records of compliance with the tailored program safety requirements. [MIL-STD-882E, Tasks 102, and 104]	Developer Software Safety Software Quality Assurance Developer Software Development & Test	Acquirer SSWG Review and Approval	**PR, AD**					All System Safety and Software Safety Engineering Artifacts SSWG Minutes Software Quality Assurance (SQA) Audits and Results
REQUIREMENTS PHASE (RP) TASKS								
RP-1: Review generic software safety requirements from other standards, including coding standards or industry best practice and identify the software safety requirements that are deemed appropriate for the system/software. Tag and track these Software Safety Requirements in the Requirements Traceability Management (RTM) tool Section 3.5.3 Identify and Tag Generic Safety Significant Requirements [MIL-STD-882E, Task 203]	Developer System Safety Developer Software Safety	Acquirer Review and Approval		**R**	**R**	**R**	**R**	List of Generic Safety-significant Requirements documented in RTM tool SSWG Minutes
RP-2: Review System Requirements Specification (SRS), and identify the functional requirements that contribute to the hazards. Tag and track these safety-significant Requirements in the Requirements Traceability Management (RTM) tool Section 3.5.1 Review System and Functional Specifications Section 3.5.2 Identify and Tag Contributing Safety Significant Requirements (CSSR) [MIL-STD-882E, Task 203]	Developer System Safety Developer Software Safety			**R**	**R**	**R**	**R**	List of Contributing Safety-significant Requirements documented in the RTM Tool
RP-3: From the FHA and the PHA Analyses, derive high-level safety requirements to mitigate	Developer System			**R**	**R**	**R**	**R**	List of Derived Safety-significant

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR — **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation — **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
identified hazards and failure modes. Tag and track these mitigating safety-significant Requirements in the Requirements Traceability Management (RTM) tool. Section 3.5.4 Identify and Tag Mitigating Safety Significant Requirements (MSSR) [MIL-STD-882E, Task 202, 203, and 208]	Safety Developer Software Safety							(high-level) Requirements
RP-4: Assign the Software Criticality and LOR to each Safety-significant Requirement based on the software control category and the highest severity of the associated hazard (functional contribution) of the safety-significant function the requirements are intending to implement. Section 3.4.8.4 Assign the Criticality LOR to the Safety Significant Function [MIL-STD-882E, Task 203]	Developer Software Safety	Developer Software Requirements		R	R	R	R	LOR Assignments to requirements based upon previously defined software function SwCI.
RP-5: Create traceability matrix from safety-significant requirements (generic, contributing or mitigating requirements) to identified hazards Section 3.5 Initiate Safety Requirements Hazard Analysis (SRHA) Section 3.8 Perform Safety Requirements Traceability [MIL-STD-882E, Task 203]	Developer System Safety Developer Software Safety	Acquirer Review and Approval		R	R	R	R	Requirements-to-Hazards Traceability Artifact SSWG Minutes
RP-6: Ensure that SwSS requirements (generic, contributing, or mitigating) are flowed down and traceable to the lower level SRS safety requirements, as they are developed. Section 3. 5 Initiate Safety Requirements Hazard Analysis (SRHA) [Summary Task for RP-1 to RP-3]	Developer System Safety	Developer Software Safety		R	R	R	R	Requirements-to-Specifications Traceability Artifacts

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR — **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation — **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
RP-7: Derive requirements to insure that safety-significant interfaces are validated and controlled at all times [Best Practice]	Developer Software Requirements	Developer Software Safety Developer Software Design Architect			**R**	**R**	**R**	Functional and Physical Design Interface Analysis
RP-8: Coordinated Safety-significant Requirements Review for correctness and completeness [Best Practice]	Developer Software Requirements	Developer Software Safety Developer Software Design Architect			**R**	**R**	**R**	Safety Requirements Review
RP-9: Derive requirements for a fault tolerant design and tag as Derived Safety-significant Requirements [Best Practice]	Developer Software Requirements	Developer Software Safety Developer Software Design Architect			**R**	**R**	**R**	Derived Fault Tolerant Requirements
RP-10: Independent review of all Contributing, Generic, and Mitigating Software Safety Requirements [Best Practice]	Someone Other Than the Developer	Independent Software Safety	**IV&V, AD**					Independent Safety Requirements Review
RP-11: Define the verification method (inspection, demonstration, analysis, or test) for each Safety-Significant Requirement [Best Practice]	Developer Software Requirements	Developer Software Safety Developer Software Design Architect		**R**	**R**	**R**	**R**	Requirements Traceability Matrix (includes verification method) Software Development and Test Artifacts
DESIGN PHASE (DP) TASKS								
DP-1: Update all analyses (PHA, FHA, SRHA, Subsystem Hazard Analysis (SSHA), SHA, and FTA) for depth and fidelity based on the maturing design concepts in the design phase of the program Section 3.3 – 3.7 [MIL-STD-882E, Tasks 202, 203, 204, 205, 208]	Developer System Safety	Developer Software Safety Acquirer Review and Approval	**PR, AD**					Updated safety engineering analysis artifacts Acquirer Approval
DP-2: From DP-1, identify and add to the SRS generic safety and coding standard requirements	Contactor System	Developer Software			**R**	**R**	**R**	List of Safety-Specific Requirements Considered to be

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR — **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation — **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
for fault detection, isolation, annunciation, and tolerance, error logging, and safe state transitions, and tag these as Mitigating Safety-significant Requirements Section 3.5.3 [Best Practice]	Safety	Safety Developer Hardware and Software Design Engineering Developer Software Development and Test Developer Software Design Architect						Safety Best Practice
DP-3: From DP-1 and DP-2, identify and add to the SRS mitigating software requirements for hazards causal factors, and tag these as Mitigating Software Safety Requirements or defects against existing high-level safety-significant Requirements. Section 3.5.4 [Best Practice]	Contactor System Safety	Developer Software Safety Developer Software Design Architect		R	R	R	R	Derived Safety Requirements OR Defects against existing Safety Requirements
DP-4: From DP-2 and DP-3, document the newly derived safety-significant requirements in the RTM tool, and track, and trace these requirements to design implementation. Section 3.5.8 [Best Practice]	Contactor System Safety	Developer Software Safety Developer Software Design Architect		R	R	R	R	RTM Tool Update Software Design Artifacts
DP-5: Review the design for compliance with the corporate safety design standards and guidelines, and Acquirer directed best practices (i.e., STANAG 4404, Appendix E of the JSSSEH, etc.) [Directed Best Practice]	Acquirer System and Software Safety Developer System and Software Safety	Developer Software Design Architect Acquirer SSWG Review and Approval			R	R	R	As directed Assessment of Compliance Artifact
DP-6: Review of the user interface design for safety-significant issues [Best Practice]	Developer System Safety Developer Software Safety	Developer Hardware and Software Design Engineering Developer Human Factors				R	R	Assessment of User Interfaces with Software Functionality
DP-7: Create traceability from all safety-significant requirements to the system and	Developer Software	Developer Software		R	R	R	R	Safety Requirements-to-design

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR
AD: As directed by Customer/Contract
IV&V: Independent Verification and Validation
N/A: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
software architecture Section 3.5 Section 3.8 [Best Practice]	Design Architect	Safety						Traceability
DP-8: Functionally partition all implementations of high LOR requirements from lower LOR requirements in the design [Best Practice]	Developer Software Design Architect	Developer Software Safety				**R**	**R**	Functionally Partitioned Design in Design Documentation Artifacts
DP-9: Assess design's stress tolerant (i.e., memory, processing through-put, timing, etc.). Make appropriate recommendations to update requirements for stress tolerant design. [Best Practice]	Developer Software Design Architect	Developer Software Safety Developer Software Requirements and Design				**R**	**R**	Stress Tolerant Design
DP-10: Perform Design Interface Analysis to evaluate internal and external interfaces of safety-critical units to ensure functional and physical compatibility across the interface. [Best Practice]	Developer Software Design Architect	Developer Software Safety				**R**	**R**	Verification that the design controls the functional and physical interfaces with safety-significant functionality
DP-11: Analyze all safety functional threads to ensure that all paths lead to their desired outcomes and that there is no dead/unused code, unused/undesired entry/exit points into/out of the software thread [Best Practice]	Developer Software Design Architect	Developer Software Safety					**R**	Safety (functional) Thread Analysis
DP-12: Verify that every variable and functional statement in safety-critical modules of code have a predefined behavior that fulfill the criteria of the functional objective [Best Practice]	Developer Software Design Architect	Developer Software Safety					**R**	Safety-specific Behavioral Review Results for Safety-Critical Modules of Code
DP-13: Independent Safety Review of Requirements-to-Design for Safety Coverage [Best Practice]	Someone Other Than System Safety Team	Independent Software Safety Independent Software Design	**IV&V, AD**					Independent Safety Review of Requirements-to-Design Coverage Artifact
IMPLEMENTATION (CODING) PHASE (IP)								

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR
AD: As directed by Customer/Contract
IV&V: Independent Verification and Validation
N/A: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
TASKS								
IP-1: Update existing FTA/Event Tree/Logic Diagram on prioritized hazards [MIL-STD-882E]	Developer System Safety	Developer Software Safety	**R, AD**					Updated FTA/Event Tree/Logic Diagram on Prioritized Hazards
IP-2: Update all Hazard Analyses to include the in-depth causal analysis that reflects the mature(ing) design Section 3.3 – 3.7 [MIL-STD-882E, Tasks 204, 205]	Developer System Safety	Developer Software Safety	**PR, AD**					Updated Hazard Analysis
IP-3: Update Safety Case or SAR as required by Customer [MIL-STD-882E, Task 301]	Developer System Safety	Developer Software Safety	**PR, AD**					Updated Safety Case or Safety Assessment Report
IP-4: Participate in Test Readiness Reviews [Best Practice]	Developer System Safety	Developer Software Safety		**R**	**R**	**R**	**R**	Test Readiness Review Artifacts
IP-5: Mark safety-significant code header with the appropriate safety-criticality or LOR assignment [Best Practice]	Developer Software Developer	Developer Software Safety		**R**	**R**	**R**	**R**	Code Headers Reflect Correct Safety Significance
IP-6: Perform reviews of code for compliance with safety-significant coding standards and guidelines (e.g., Motor Industry Software Reliability Association (MISRA)) [Best Practice]	Developer Software Developer	Developer Software Safety Developer Software Quality Assurance (QA)	**PR**					Artifacts Demonstrating Compliance with Best Practices for Safety-Critical Code Development
IP-7: Perform detailed code walkthroughs and analysis of safety-critical code Section 3.9 Perform Code Level Safety Analysis [Best Practice]	Developer Software Design Architect Developer Software Developer	Developer Software Safety					**R**	Code Level Review Results
IP-8: Create traceability from code to safety-significant design requirements	Developer Software Design Architect Developer Software	Developer Software Safety			**R**	**R**	**R**	Requirements-to-Code Traceability

Legend:

PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR
AD: As directed by Customer/Contract
IV&V: Independent Verification and Validation
N/A: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Section 3.8 [Best Practice]	Developer							
IP-9: Participate in acceptance review of safety-significant code [Best Practice]	Developer Software Safety	Developer Software Developer and Software Test			**R**	**R**	**R**	Acceptance Review of Safety significant Software
IP-10: Independent Safety Review of Safety-Significant Code Section 3.9 Perform Code Level Safety Analysis [MIL-STD-882E]	Independent Design	Independent Software Safety	**IV&V, AD**					Safety Code-Level Review
IP-11: Perform detailed code inspections for fault contributions of Safety-Significant Code Section 3.9.4 Analyze LOR-1 Software [Best Practice]	Software Development Team	Software Test Software Safety					**R**	Safety Code-Level Analysis for Fault Management
IP-12: Review unit test plan to ensure that it defines the requirements for testing units of safety-significant code Section 3.10.2 Ensure Safety Functionality is Tested [Best Practice]	Developer Software Safety			**R**	**R**	**R**	**R**	Assessment of Unit Test Plan for Requirements Definition
IP-13: Execute unit tests [Best Practice]	Developer Software Developer			**R**	**R**	**R**	**R**	Documented results of Unit Test Execution
IP-14: Unit test results review [Best Practice]	Developer Software Developer	Developer Software Safety		**R**	**R**	**R**	**R**	Assessment of Unit Test Results
IP-15: Review unit test results and verify that the unit tests provide the required unit test coverage and were executed in compliance with the unit test plan [Best Practice]	Developer Software Test	Developer Software Safety		**R**	**R**	**R**	**R**	Documented results of Unit Test Review
TEST PHASE (TP) TASKS								

Legend:

PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	→ Level-Of-Rigor →					Representative Artifacts Produced
			Baseline	4	3	2	1	
TP-1: Finalize the System Hazard Analysis (SHA) Section 3.6 [MIL-STD-882E, Task 204, 205, 206, 208]	Developer System Safety	Developer Software Safety		**R**	**R**	**R**	**R**	Final Hazard Analysis Artifacts
TP-2: Mark safety-significant test cases with the appropriate LOR [Best Practice]	Developer Software Safety	Developer Software Test		**R**	**R**	**R**	**R**	Evidence within the Safety-Specific Software Test Cases
TP-3: Perform a safety review of each test case 3.10 Perform Software Test Planning [Best Practice]	Developer Software Safety			**R**	**R**	**R**	**R**	Safety Review Results
TP-4: Review all requirements traceability matrices for coverage and completeness [Best Practice]	Developer System Safety	Developer Software Safety			**R**	**R**	**R**	Requirements Traceability Review Results
TP-5: Develop software test case procedures to demonstrate software structure (statement coverage) is achieved [Best Practice]	Developer Software Test	Developer Software Design Architect Developer Software Safety			**R**	**R**	**R**	Evidence within the Software Test Plan Documented Code Structural Coverage evidence
TP-6: Develop software test case procedures to demonstrate software structure (condition/decision coverage(C/DC)) is achieved [Best Practice]	Software Test Software Design	Software Safety				**R**	**R**	Safety-Specific Software Test Cases Documented Code Structural Coverage evidence
TP-7: Develop software test case procedures to demonstrate software structure (modified condition/decision coverage (MC/DC)) is achieved. [Best Practice]	Developer Software Test	Developer Software Design Architect Developer Software Safety					**R**	Safety-Specific Software Test Cases Documented Code Structural Coverage evidence
TP-8: Perform a software structural coverage analysis to demonstrate that the appropriate level of software structural coverage, including data coupling and control coupling, has been	Developer Software Test	Developer Software Design Architect Developer Software Safety				**R**	**R**	Safety-Specific Software Test Cases Documented Code Structural Coverage evidence

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR — **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation — **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
achieved. [Best Practice]								
TP-9: Develop software test cases to demonstrate that the software satisfies its requirements and those anomalous conditions or software errors cannot lead to a hazardous condition as identified by the Hazard Analyses from the System Safety process. [Best Practice]	Developer Software Test	Developer Software Design Architect Developer Software Safety			**R**	**R**	**R**	Safety-Specific Software Test Cases
TP-10: Each software requirement identified as safety significant in the System Safety Hazard Analysis process shall be traced to a test case and each test case shall trace back to a software requirement. [Best Practice]	Developer Software Test	Developer Software Design Architect Developer Software Safety				**R**	**R**	Safety-Specific Software Test Cases
TP-11: Develop software test cases to demonstrate the ability of the software to correctly respond to off-nominal, robustness, and failure mode conditions as identified by the system Safety Hazard Analysis process. Off Nominal and robustness conditions that must be considered are: abnormal, out-of-bounds, and invalid variable input values including zero, zero crossing and approaching zero from either direction or similar values of trig functions; proper state transitions and possible disallowed state or mode transitions; system initialization under abnormal and failure conditions; errors in input values or counters associated with time or rate functions and algorithms; failure modes of input data strings and messages; out of range loop counters and other loop failure conditions; exception handling correctness; fault and error handling correctness [Best Practice]	Developer Software Test	Developer Software Safety				**R**	**R**	Safety-Specific Software Test Cases
TP-12: Perform a software test coverage analysis to demonstrate that test case procedures meet	Developer Software Test	Developer Software Safety				**R**	**R**	Safety Requirements-to-Test Cases Trace

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
the requirements based test coverage criteria: a test case exist for each software requirement; test cases satisfy the criteria for normal; robustness, and failure mode testing; all test procedures used to satisfy structural coverage are traced to requirements; and that requirements or structural coverage deficiencies are resolved by identification or new requirements or new test cases. [Best Practice]								
TP-13: Create a safety-significant test report documenting the safety-significant formal testing compliance and execution results [Best Practice]	Developer Software Test	Developer Software Safety		**R**	**R**	**R**	**R**	Safety-Critical Test Report
TP-14: Review safety-significant test results and verify that the safety-significant test cases provide the required test coverage and were executed in compliance with the formal test plans. [Best Practice]	Developer Software Test	Developer Software Safety Developer Software Quality		**R**	**R**	**R**	**R**	Verification of Test Case Implementation
TP-15: Track safety verification failures and participate in test anomaly resolution Section 3.11.3 Process Subtask 11.3: Monitor Test Defects and Corrective Actions [Best Practice]	Developer Software Design Developer Software Test	Developer Software Safety		**R**	**R**	**R**	**R**	Attendance Log
TP-16: Plan, perform, and review functional and Failure Modes and Effects Test (FMET) test plans and procedures. Section 3.11.4 Review Final Software Test Results [Best Practice]	Developer Software Design Developer Software Test	Developer Software Safety		**R**	**R**	**R**	**R**	Regression Test Plans and Procedures
TP-17: Add test cases to the Regression Test Plan to support 100% regression testing for safety-significant functions that have been updated. [Best Practice]	Developer Software Test	Developer Software Design Developer Software Safety		**R**	**R**	**R**	**R**	Regression Test Plans and Procedures

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR
AD: As directed by Customer/Contract
IV&V: Independent Verification and Validation
N/A: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	→ Level-Of-Rigor →					Representative Artifacts Produced
			Baseline	4	3	2	1	
TP-18: Perform 100% regression testing for any safety-significant functions that have been updated Regression testing must include testing of non-partitioned non-safety-significant software at the same LOR as the safety-significant software. [Best Practice]	Developer Software Test	Developer Software Safety			**R**	**R**	**R**	Regression Test Results or Report
TP-19: Perform hazard risk assessments based upon results of verification activities. Section 3.12 Perform Safety Risk Assessment [MIL-STD-882E, Task 301]	Developer System Safety	Developer Software Safety		**R**	**R**	**R**	**R**	Safety Risk Assessment
TP-20: Gain accreditation and validation of models and simulations that are used to support software system safety verification in accordance with DoDI 5000.61, DoD Modeling and Simulation (M&S) Verification, Validation, and Accreditation (VV&A). [Best Practice]	Developer Software Engineering					**R**	**R**	
TP-21: Validate models and simulations against actual hardware and data. [Best Practice]	Developer Software Engineering					**R**	**R**	
TP-22: SwSS personnel shall support the system safety risk assessment process. Section 3.12 Perform Safety Risk Assessment [Best Practice]	Developer and Acquirer SSWG			**R**	**R**	**R**	**R**	Problem Reports, adjudications, SSWG Minutes
LIFE CYCLE (LC) SUPPORT PHASE TASKS								
LC-1: Review of all Engineering Change Proposals for safety applicability to safety-significant functions and mishaps/hazards (ECP) Section 3.13 Participate in Life-cycle	Developer System Safety Team	Developer Software Safety	**PR**					ECP Review Results with System Safety assessment indicated

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR | **AD**: As directed by Customer/Contract
IV&V: Independent Verification and Validation | **N/A**: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Management and Support [MIL-STD-882E, Task 304]								
LC-2: Identify and track safety-significant requirements to mitigate the safety risk potential of the Software Trouble Reports (STRs) or Engineering Change Proposals (ECPs) being processed. Section 3.12.3 Assess Partial Mitigation or Failure to Mitigate, Section 3.13 Participate in Life-Cycle Management and Support [MIL-STD-882E, Task 304]	Developer System Safety Team	Developer Software Safety	**PR**					
LC-3: Update SSHA, SHA, and FHA (as required) Section 3.13.6 Update all Safety Related Artifacts [MIL-STD-882E, Task 204, 205, 208]	Developer System Safety Team	Developer Software Safety	**R, AD**					Updated FHA Updated SSHA and SHA **NOTE:** *If the safety analysis has NOT been accomplished on the system (e.g., legacy system), then it must be accomplished now)*
LC-4: Update the FTA (as required) Section 3.13.6 [MIL-STD-882E]	Developer System Safety Team	Developer Software Safety	**R, AD**					Updated FTA
LC-5: Update the HTS (as required) Section 3.13.6 [MIL-STD-882E]	Developer System Safety Team	Developer Software Safety	**R AD**					Updated HTS
LC-6: Participate in Configuration Management process and Configuration Control Board (CCB) [Best Practice]	Developer System Safety Team	Developer Software Safety	**PR**					Safety Review and Approval of ECPs and STRs
LC-7: Mark safety-significant software ECPs and items in CM with the appropriate LOR as determined by the SwCI [Best Practice]	Developer Software Safety	Developer Configuration Management Developer Software Design Architect	**PR**					ECP Review Results

Legend:
PR: Prerequisite Requirement– Required regardless of LOR or required in order to assess and determine LOR
ACQ: Task Requirement Performed by the Acquirer
R: Required for assigned LOR
AD: As directed by Customer/Contract
IV&V: Independent Verification and Validation
N/A: Not Applicable for this program or LOR

Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
LC-8: Review problem reporting/defect tracking, change control, and change review activities for safety impact and compliance [Best Practice]	Developer Software Safety	Developer Configuration Management Developer Software Design Architect	**PR**					Software Trouble Report and Defect Tracking Results
LC-9: Document the results of any Safety Reviews [Best Practice]	Developer System Safety	Developer Software Safety	**PR**					Safety Review Results
Note: Regression testing must be performed on all changed or modified safety-significant software in system sustainment. See TP-18 and TP-19				**R**	**R**	**R**	**R**	Regression Test Plans Regression Test Results
LC-10: Review Test Reports [Safety, Test] [Best Practice]	Developer System Safety Team	Developer Software Safety	**PR**					Test Report Review
LC-11: Review and give signature approval on safety-significant CRs [Best Practice]	Developer System Safety Team	Developer Software Safety	**PR**					Safety-significant CR Signature from Software Safety
LC-12: Independently review safety-significant code changes implemented by ECPs within the CM process [Best Practice]	Someone Other Than Designer/Developer		**IV&V, AD**					Independent Safety Review Results