20028
Joint Software Systems Safety Engineering Handbook Implementation Guide

Robert E. Smith, CSP
Booz Allen Hamilton

20th Annual NDIA Systems Engineering Conference
Springfield, VA

25 October 2017

BLUF

- **System Safety, to include <u>Software Safety</u>, is required for acquisition programs IAW DoDI 5000.02 and MIL-STD-882E**
- **Detailed guidance for software safety is provided in the Joint Software Systems Safety Engineering Handbook (JSSSEH) Version 1.0 published 27 August 2010 as referenced in MIL-STD-882E**
  - Comprehensive handbook, although lengthy at 344 pgs
  - Acquisition Programs unfamiliar with software safety find it difficult to extract software safety techniques and processes in order to satisfy MIL-STD-882E Software Level of Rigor (LOR) requirements
  - Programs typically re-state the LOR table from MIL-STD-882E, Table V in their Safety Plans and do not identify and specify the artifacts and Objective Quality Evidence (OQE) to be produced for all LOR tasks
  - Could result in not performing a comprehensive software safety program and therefore not fully characterizing software's contribution to system risk
- **Joint Boards recognized this concern and developed a JSSSEH Implementation Guide on 1 April 2016 to further assist programs, and was endorsed by the Joint Services Weapon Safety Review (JSWSR) Boards on 29 June 2016**
- **Revised Implementation Guide (Rev A) issued 17 October 2017**

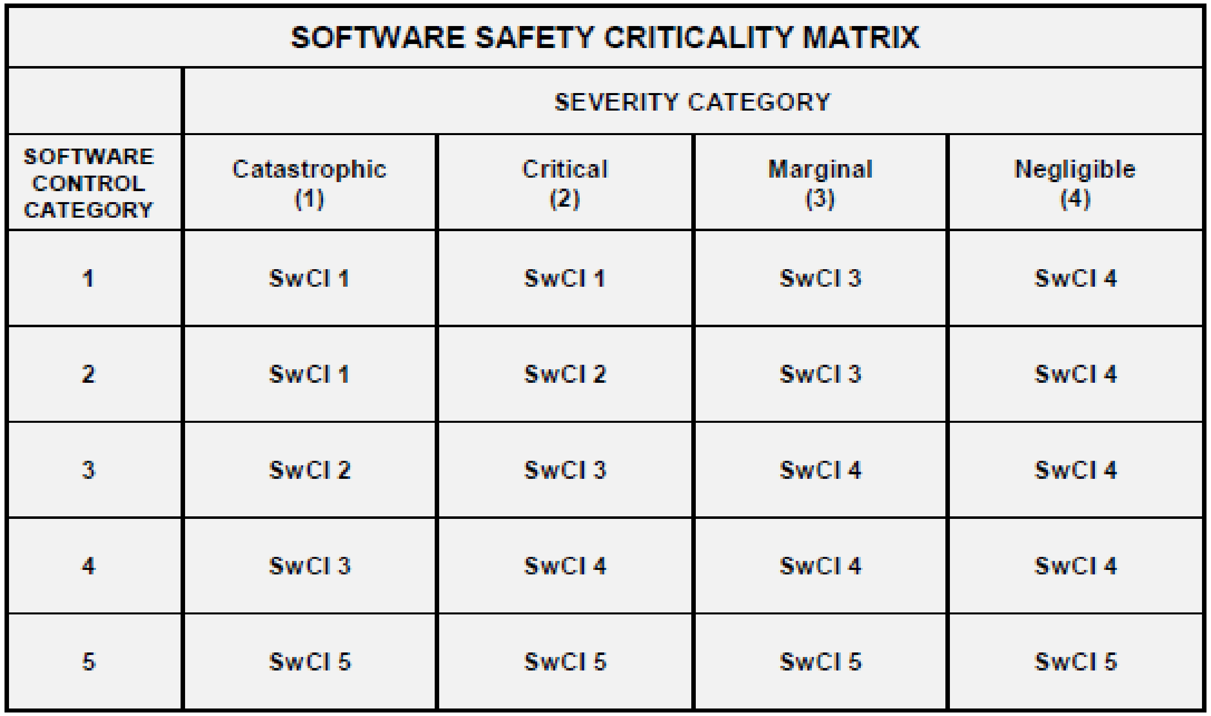
Software Safety Requirements

- **<u>Software Safety</u> is required for acquisition programs**
  - DoDI 5000.02, Enclosure 3, Para 11 – SOFTWARE "…The SEP should address the following: software unique risks; inclusion of software in technical reviews; identification, tracking, and reporting of metrics for software technical performance, process, progress, and quality; **<u>software safety</u>** and security considerations; and software development resources."
  - DoDI 5000.02, Enclosure 3, Para 16 - ENVIRONMENT, SAFETY, AND OCCUPATIONAL HEALTH (ESOH) "The Program Manager will integrate ESOH risk management into the overall systems engineering process for all engineering activities throughout the system's life cycle. As part of risk reduction, the Program Manager will eliminate ESOH hazards where possible, and manage ESOH risks where hazards cannot be eliminated. **<u>The Program Manager will use the methodology in MIL-STD-882E</u>**..."
  - MIL-STD-882E, Section 4.4 Software contribution to system risk. "The assessment of risk for software, and consequently software-controlled or software-intensive systems, cannot rely solely on the risk severity and probability….. Therefore, **<u>another approach shall be used for the assessment of software's contributions to system risk</u>** that considers the potential risk severity and the degree of control that software exercises over the hardware."

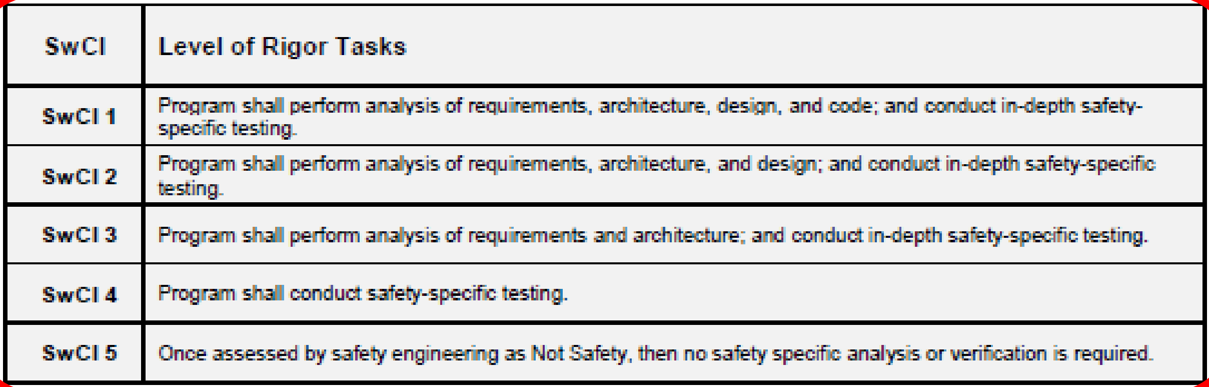
Common Approaches to Software Safety

- MIL-STD-882E references the JSSSEH and Section 4.4.2 includes a note to “Consult the Joint Software Systems Safety Engineering Handbook and AOP 52 for additional guidance on how to conduct required software analyses.”
- The JSSSEH is a lengthy document making it difficult for programs not familiar with software safety activities to extract detailed LOR tasks and tailor for particular program needs
- Programs often default to only referencing or reusing the LOR table from MIL-STD-882E (i.e., Table V) as their software safety approach in their System Safety Management Plans (SSMPs) and/or System Safety Program Plans (SSPPs)
- May result in not performing the specific LOR tasks that comprise a comprehensive software safety program, resulting in failure to assess software’s contribution to system risk(s)

MIL-STD-882E, Table V, Software Safety Criticality Matrix



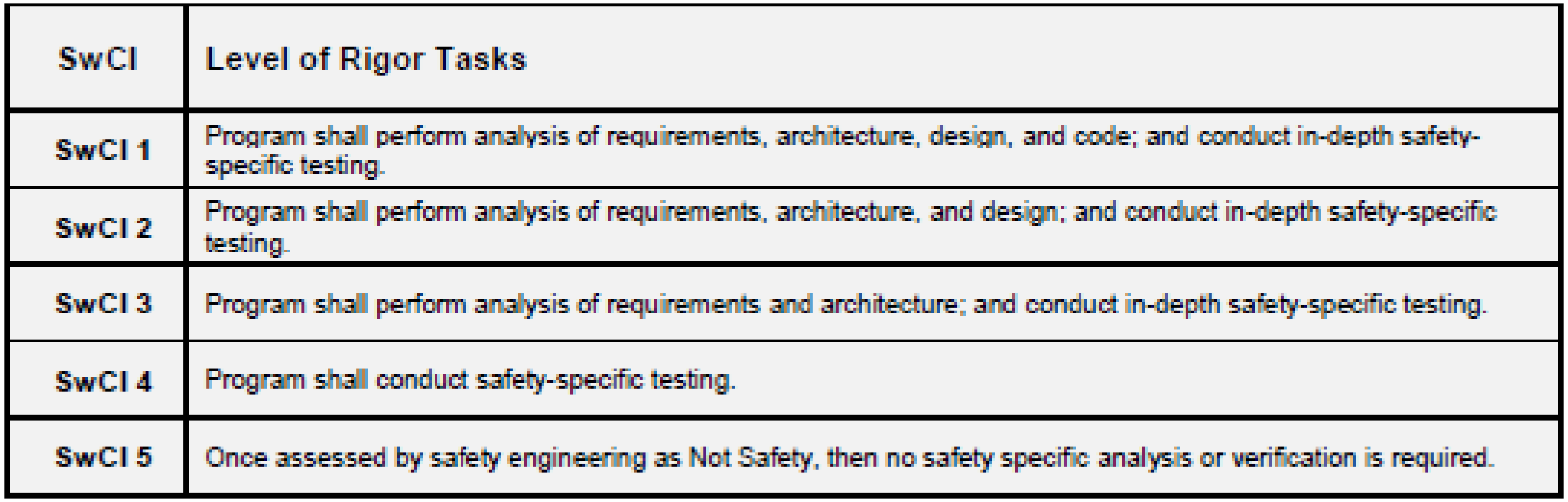
SOFTWARE SAFETY CRITICALITY MATRIX				
	SEVERITY CATEGORY			
SOFTWARE CONTROL CATEGORY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
1	SwCI 1	SwCI 1	SwCI 3	SwCI 4
2	SwCI 1	SwCI 2	SwCI 3	SwCI 4
3	SwCI 2	SwCI 3	SwCI 4	SwCI 4
4	SwCI 3	SwCI 4	SwCI 4	SwCI 4
5	SwCI 5	SwCI 5	SwCI 5	SwCI 5



SwCI	Level of Rigor Tasks
SwCI 1	Program shall perform analysis of requirements, architecture, design, and code; and conduct in-depth safety-specific testing.
SwCI 2	Program shall perform analysis of requirements, architecture, and design; and conduct in-depth safety-specific testing.
SwCI 3	Program shall perform analysis of requirements and architecture; and conduct in-depth safety-specific testing.
SwCI 4	Program shall conduct safety-specific testing.
SwCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

High Level, overarching LOR tasks

MIL-STD-882E, Table V, Level of Rigor Tasks



SwCI	Level of Rigor Tasks
SwCI 1	Program shall perform analysis of requirements, architecture, design, and code; and conduct in-depth safety-specific testing.
SwCI 2	Program shall perform analysis of requirements, architecture, and design; and conduct in-depth safety-specific testing.
SwCI 3	Program shall perform analysis of requirements and architecture; and conduct in-depth safety-specific testing.
SwCI 4	Program shall conduct safety-specific testing.
SwCI 5	Once assessed by safety engineering as Not Safety, then no safety specific analysis or verification is required.

- **Note that the LOR tasks table contains no details on the specific tasks, artifacts and Objective Quality Evidence (OQE) to be produced for LOR (e.g., requirements analysis, architecture analysis, design analysis, safety-specific testing, and code analysis)**
- **The JSSSEH includes these details, but not in a specific location**
- **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 Software System Safety Engineering (SSSE) Requirements**

Implementation Guide Overview

- **Developed by the Joint Services – Software Safety Authorities (JS-SSA) Sub-Working Group in support of the JSWSR Boards on 1 April 2016 - endorsed by the JSWSR Boards on 29 June 2016**
- **Titled "Software System Safety Implementation Process and Tasks Supporting MIL-STD-882E With Joint Software System Safety Engineering Handbook References"**
  - Short name – "Implementation Guide"
- **Provides implementation guidance for Software System Safety program requirements specified in MIL-STD-882E and guidance detailed in the JSSSEH**
- **Updated in 2017 to address identified errors, Service comments and create more direct alignment with the Tasks in MIL-STD-882E**
- **Released as "Revision A" on 17 October 2017**

Implementation Guide Outline and Methodology

- The implementable process task requirements are presented as a decomposition of parent and children activities, similar to a Work Breakdown Structure (WBS)
- Parent tasks are graphically represented depicting inputs to the tasks and the products that the task would typically 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 are specified and cross referenced to both the applicable MIL-STD-882E requirements and JSSSEH sections and paragraphs that provide guidance on meeting the requirements

Process Tasks (2016 Guide)

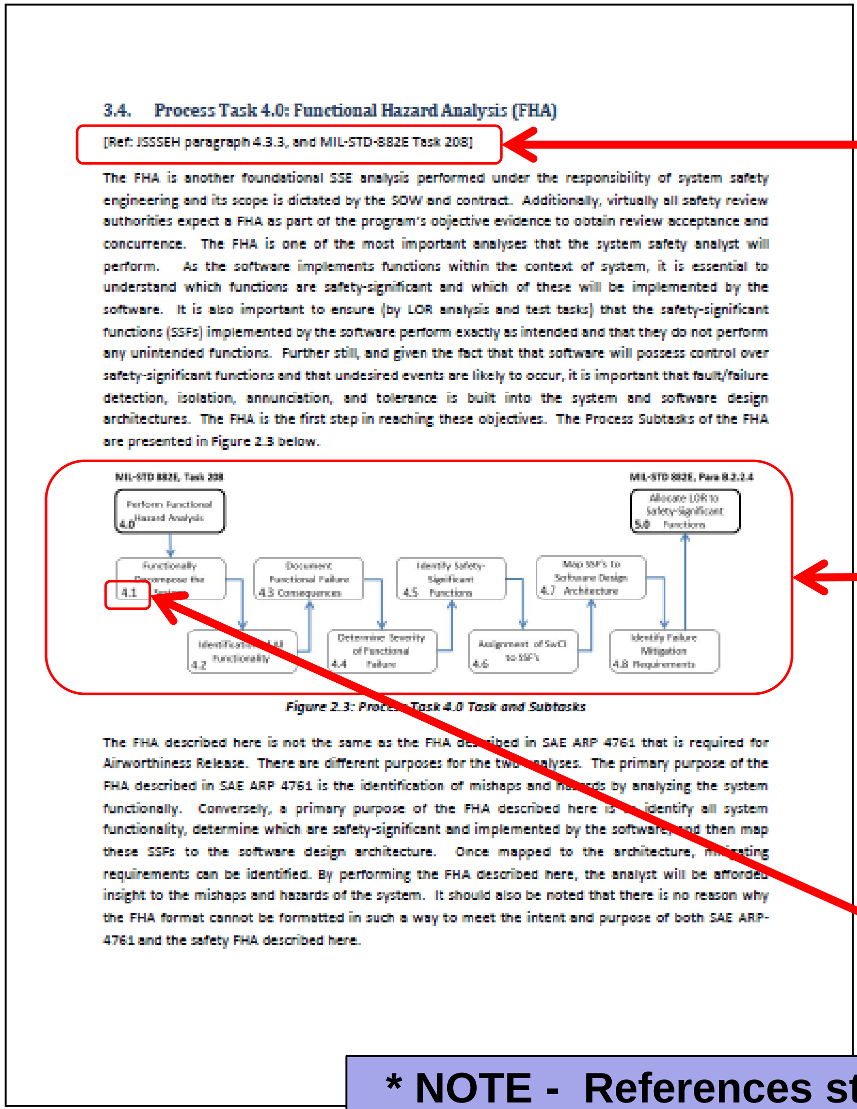
- **14 Process Tasks identified in the Implementation Guide**
  - Process Task 1.0: Prepare the System Safety Management Plan (SSMP)
  - Process Task 2.0: Prepare System Safety Program Plan (SSPP)
  - Process Task 3.0: Preliminary Hazard Analysis
  - Process Task 4.0: Functional Hazard Analysis (FHA)
  - Process Task 5.0: LOR Allocations to Safety-Significant Functions
  - Process Task 6.0: Preliminary Safety Requirements Analysis (SRA)
  - Process Task 7.0: Perform In-Depth Hazard Analysis
  - Process Task 8.0: Perform Detailed Safety Requirements Analysis
  - Process Task 9.0: Perform Safety Requirements Traceability
  - Process Task 10.0: Perform Code-Level Safety Analysis
  - Process Task 11.0: Perform Software Test Planning
  - Process Task 12.0: Monitor Safety-Significant Software Testing
  - Process Task 13.0: Perform Residual Safety Risk Assessment
  - Process Task 14.0: Participate in Life-Cycle Management and Support
- **Each Process Task has Process Subtasks to amplify details and/or additional steps associated with each Task**

Process Tasks (2017 Guide)

- **13* Process Tasks identified in the Implementation Guide**
  - Process Task 1.0: Prepare the System Safety Management Plan (SSMP)
  - Process Task 2.0: Prepare System Safety Program Plan (SSPP)
  - Process Task 3.0: Preliminary Hazard Analysis
  - Process Task 4.0: Functional Hazard Analysis (FHA)*
  - Process Task 5.0: Initiate Safety Requirements Hazard Analysis (SRHA)*
  - Process Task 6.0: Perform System and Subsystem Hazard Analyses*
  - Process Task 7.0: Finalize SRHA*
  - Process Task 8.0: Perform Final Safety Requirements Traceability*
  - Process Task 9.0: Perform Code-Level Safety Analysis
  - Process Task 10.0: Perform Software Test Planning
  - Process Task 11.0: Monitor Safety-Significant Software Testing
  - Process Task 12.0: Perform Safety Risk Assessment*
  - Process Task 13.0: Participate in Life-Cycle Management and Support
- **Each Process Task has Process Subtasks to amplify details and/or additional steps associated with each Task**

*** Changes in 2017: Titles of tasks revised and previous Task 5.0 combined into Task 4.0, and SRA is now System Requirements Hazard Analysis (SRHA)**

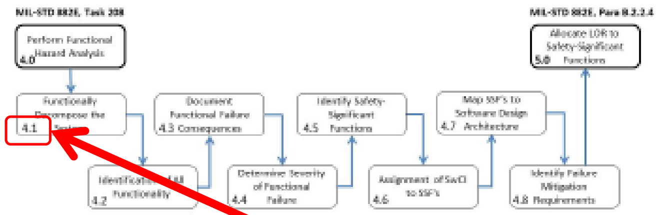
Process Tasks 4.0 – FHA [Partial Example]



3.4. Process Task 4.0: Functional Hazard Analysis (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 to obtain 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 safety-significant functions (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 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
MIL-STD 882E, Para 8.2.2.4
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 Assignment of SwCI to SSF's
4.7 Map SSF's to Software Design Architecture
4.8 Identify Failure Mitigation Requirements
5.0 Allocate LOR to Safety-Significant Functions

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 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, 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.

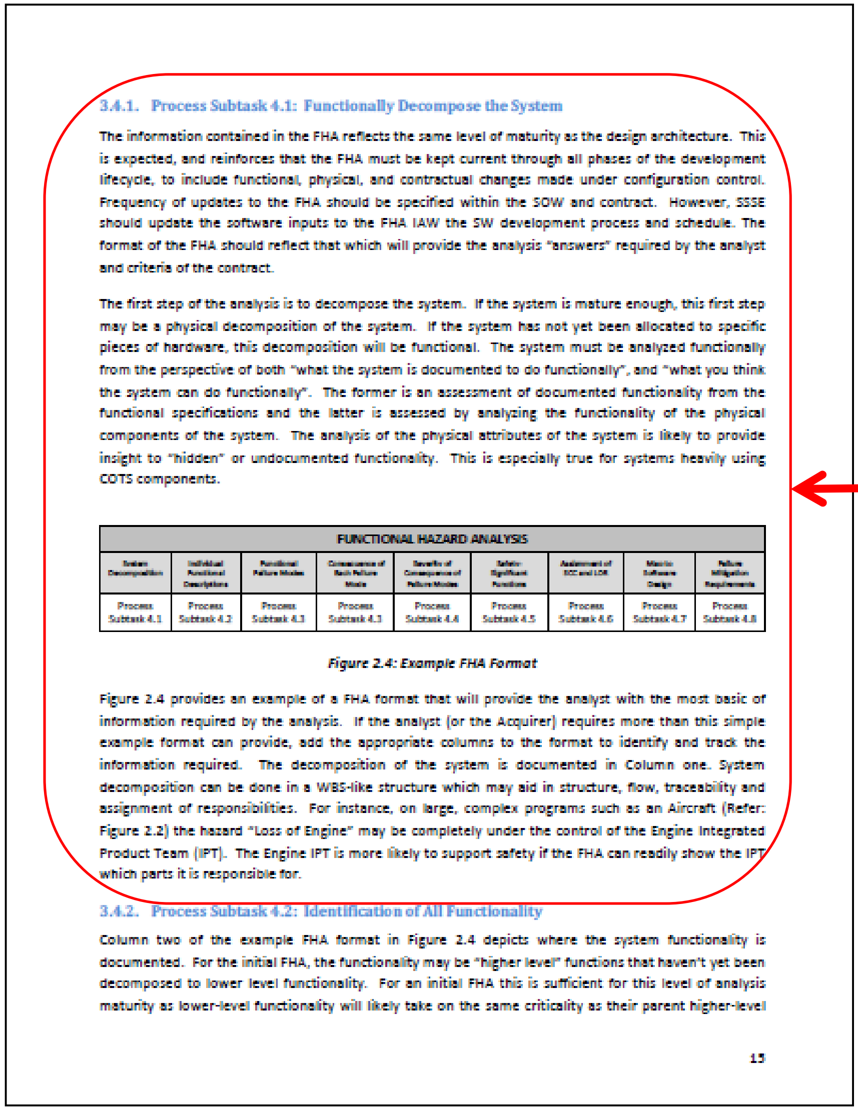
References* to specific sections in JSSSEH and MIL-STD-882E

Process flow diagram* provided

Process Task / Subtask referenced for each step

*** NOTE - References still the same in 2017 Guide. Flow diagrams altered as appropriate.**

Process Tasks 4.0 – FHA [Partial Example]



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 SW 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 COTS components.

FUNCTIONAL HAZARD ANALYSIS								
System Decomposition	Individual Functional Descriptions	Functional Failure Modes	Consequences of Each Failure Mode	Severity of Consequences of Failure Modes	Safety-Significant Functions	Assessment of SCC and LOR	Mods 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 4.6	Process Subtask 4.7	Process Subtask 4.8

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, 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 WBS-like structure which may aid in structure, flow, traceability and assignment of responsibilities. For instance, on large, complex programs such as an Aircraft (Refer: 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.

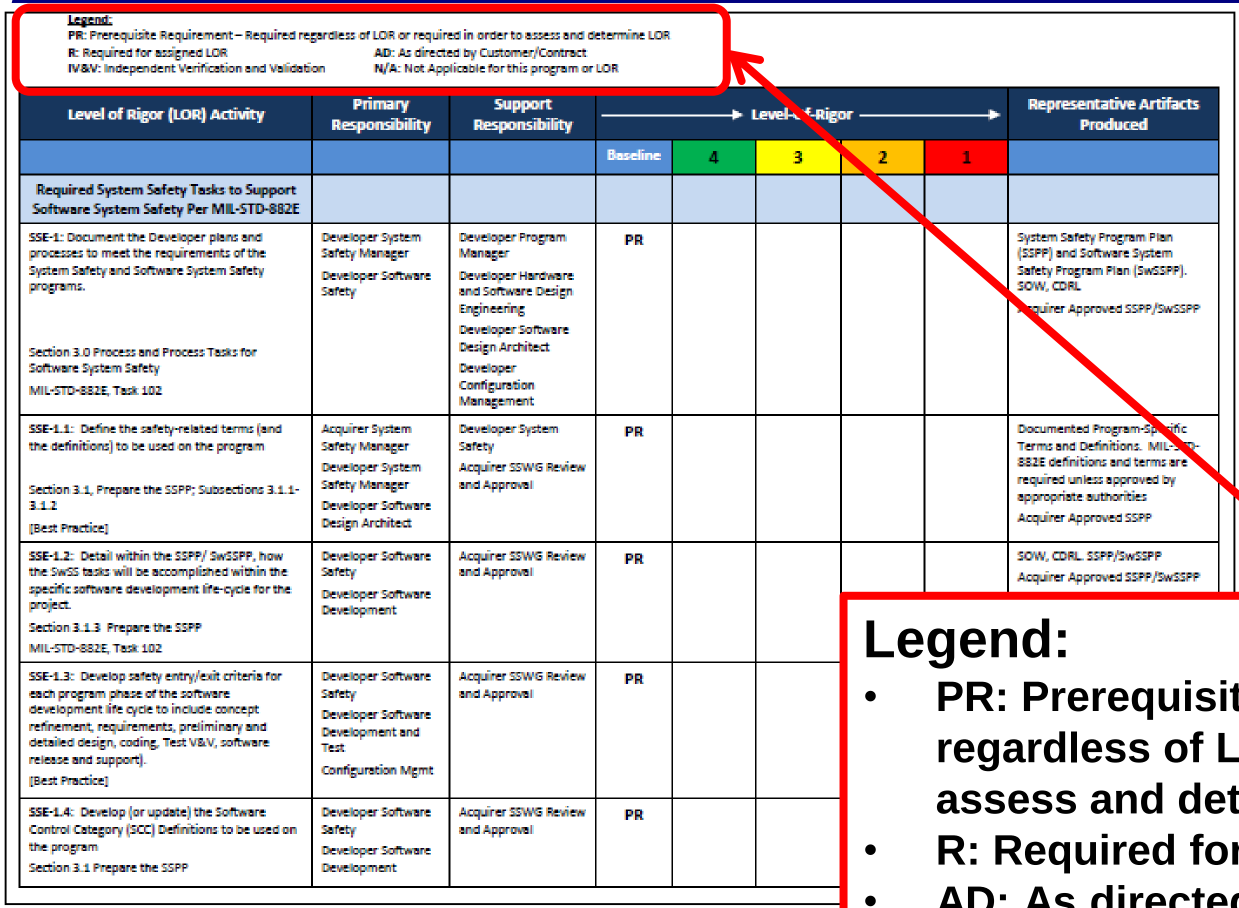
3.4.2. Process Subtask 4.2: Identification of All Functionality

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 haven't 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

15

Process Task / Subtask described in detail in subsequent paragraphs

Appendix A – LOR Task Table [Partial]



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

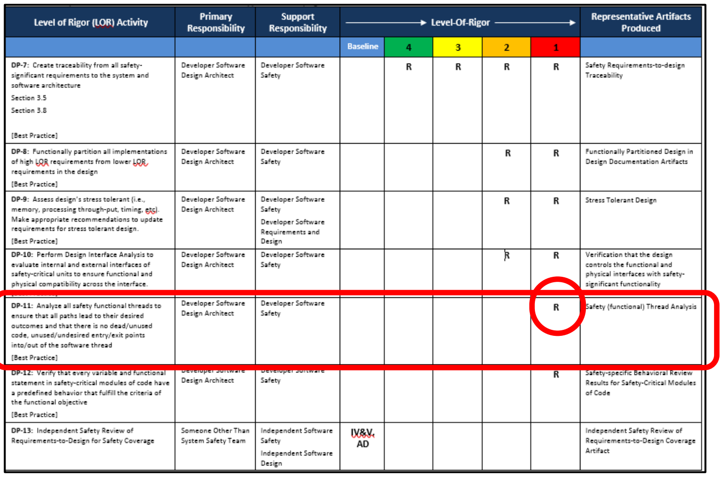
Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-of-Rigor					Representative Artifacts Produced
			Baseline	4	3	2	1	
Required System Safety Tasks to Support Software System Safety Per MIL-STD-882E								
SSE-1: Document the Developer plans and processes to meet the requirements of the System Safety and Software System Safety programs. Section 3.0 Process and Process Tasks for Software System Safety MIL-STD-882E, Task 102	Developer System Safety Manager Developer Software Safety	Developer Program Manager Developer Hardware and Software Design Engineering Developer Software Design Architect Developer Configuration Management	PR					System Safety Program Plan (SSPP) and Software System Safety Program Plan (SwSSPP). SOW, CDRL Acquirer Approved SSPP/SwSSPP
SSE-1.1: Define the safety-related terms (and the definitions) to be used on the program Section 3.1, Prepare the SSPP; Subsections 3.1.1-3.1.2 [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 3.1.3 Prepare the SSPP 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 V&V, software release and support). [Best Practice]	Developer Software Safety Developer Software Development and Test Configuration Mgmt	Acquirer SSWG Review and Approval	PR					
SSE-1.4: Develop (or update) the Software Control Category (SCC) Definitions to be used on the program Section 3.1 Prepare the SSPP	Developer Software Safety Developer Software Development	Acquirer SSWG Review and Approval	PR					

Legend:

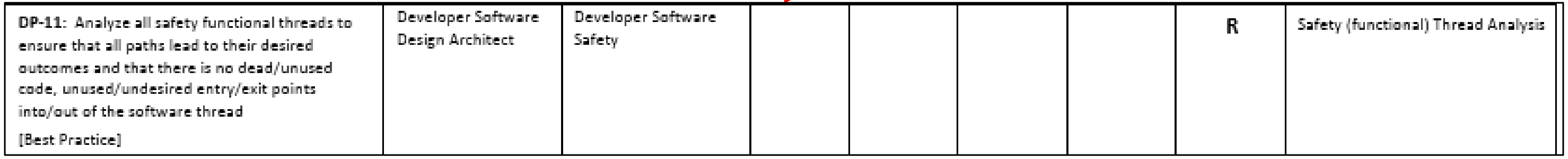
- PR: Prerequisite Requirement – Required regardless of LOR or required in order to assess and determine LOR
- R: Required for assigned LOR
- AD: As directed by Customer/Contract
- IV&V: Independent Verification and Validation
- N/A: Not Applicable for this program or LOR

LOR 1 Example [Partial]

- **Table indicates required ("R") LOR activities for LOR 1, 2, 3, and 4**
- **E.g., Design Practice (DP)-11: Analyze all safety functional threads…**
  - Required only for LOR 1
  - One of many LOR 1 activities required ("R") for LOR 1
  - Appendix A specifies the LOR activity, primary and support activities, applicable LOR, and artifact(s) to be produced



Level of Rigor (LOR) Activity	Primary Responsibility	Support Responsibility	Level-Of-Rigor					Representative Artifacts Produced
			Baseline	🟩 4	🟨 3	🟧 2	🟥 1	
DP-7: Create traceability from all safety-significant requirements to the system and software architecture Section 3.5 Section 3.8 [Best Practice]	Developer Software Design Architect	Developer Software Safety		R	R	R	R	Safety Requirements-to-design 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	Someone Other Than System Safety Team	Independent Software Safety Independent Software Design	IV&V, AD					Independent Safety Review of Requirements-to-Design Coverage Artifact



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

Change Management

- **JS-SSA meets twice annually**
- **Approved path for changes:**
  - Any user can submit comments
  - Comments collected from 4th QTR FY until end of 2nd QTR FY (comments, corrections, additions, deletions, etc.)
  - **Submit comments to JS-SSA Chair**
  - Proposed changes adjudicated between the Service JS-SSA Implementation Guide (IG) IPT
  - Changes approved by the JS-SSA Sub-group will then be integrated into the Implementation Guide and a new revision released in time for the Fall meeting (or end of year)
- **100+ proposed changes submitted during the FY2017 review period**
- **Proposed changes were adjudicated via email and in a face-to-face meeting April 2017**
- **Draft JS-SSA IG update distributed to Working Group and approved in August 2017**
- **Release of 2017 Guide Update (Rev A) on 17 October 2017**

2017 Summary of Changes

- **Numerous changes between 2016 Guide and 2017 Guide**
- **Two “Critical” changes to the Implementation Guide**
  - Less emphasis and more controls on tailoring of LOR table by contractors (Section 2.0)
    - Changed from: “The LOR table should be tailored for any given program as agreed to by the Acquirer and Developer.”
    - To: “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.”
  - Allows risks to be carried over, if appropriate, from one contractual activity to another following a reassessment (Section 3.2.4.2)
    - Changed from: “Risk accepted in one contractual activity should never be carried over as the baseline for the next contractual activity.”
    - To: “Risk acceptance performed in one contractual activity should be reassessed for the next contractual activity.”

2017 Summary of Changes (cont.)

- **Four "Significant" changes to the Implementation Guide**
  - SSMP tasks added to the LOR table in Appendix A as "Acquirer" activities
  - Removed requirement that Contractors must comply with future versions of DODI 5000.02 and MIL-STD-882, just the versions under contract
  - Clarified purpose of document as defining the processes and tasks needed to implement a MIL-STD-882E compliant SSSE program
  - Made the current Process Task 5.0 "LOR Allocations to Safety-Significant Functions" a subtask of draft Process Task 4.0 "FHA"
- **Majority of remaining changes are relatively minor and designed to resolve known inconsistencies and improve alignment with MIL-STD-882E**
  - Primarily changes to the process flow figures and associated paragraphs detailing the subtasks for the analyses/reports (PHA, SRA, etc.) to better define tasks and processes
  - Many editorial and administrative corrections
- **Changed "Hazard Risk Index" to "Risk Assessment Code"**
- **Changes to the LOR table in Appendix A**

2017 Summary of Changes – Appendix A

- **Seven new Baseline LOR SSE-related activities detailing Acquirer (i.e., "ACQ-#") responsibilities**
- **Some activity descriptions updated and enhanced, but overall, no other new activities added**

Level-Of-Rigor Activity / Task Type	2016 IG	2017 IG	Change
Acquirer (ACQ-#.#)	0	7	+7
System Safety Engineering (SSE-#.#)	22	22	-
Requirements Phase (RP-#)	11	11	-
Design Phase (DP-#)	13	13	-
Implementation (Coding) Phase (IP-#)	15	15	-
Test Phase (DP-#)	23	23	-
Life Cycle Support Phase (LC-#)	12	12	-
TOTAL ACTIVITIES / TASKS	**96**	**103**	**+7**

2017 Summary of Changes – Appendix A (cont.)

- **Several activities now required to be performed at lower LOR to align with MIL-STD-882E Table V LOR requirements**

Level-Of-Rigor	2016 IG	2017 IG	Change
Baseline	42	49	+7
1	54	54	-
2	47	49	+2
3	35	38	+3
4	20	27	+7
TOTAL ***(LOR 1 + Baseline)***	96	103	+7

Conclusion

- **Software Safety is required for acquisition programs IAW DoDI 5000.02 and MIL-STD-882E**
- **Additional guidance for software safety is provided in the JSSSEH Version 1.0 published 27 August 2010 as referenced in MIL-STD-882E**
- **Joint Boards developed a JSSSEH Implementation Guide on 1 April 2016 to further assist program perform software safety, and was endorsed by the JSWSR Boards on 29 June 2016**
- **2017 Implementation Guide Update (Rev A) release on 17 October 2017**
- **Implementation Guide will be updated annually, as required**

Implementation Guide assists in performing a comprehensive software safety program to fully characterize software's contribution to system risk

Resources (location of documents)

- **DAU Acquisition Community Connection Site, ESOH Community**
  - https://acc.dau.mil/ESOH

    ---or---
  - https://www.dau.mil/cop/esoh/Pages/Default.aspx
  - look under the “Resources” section

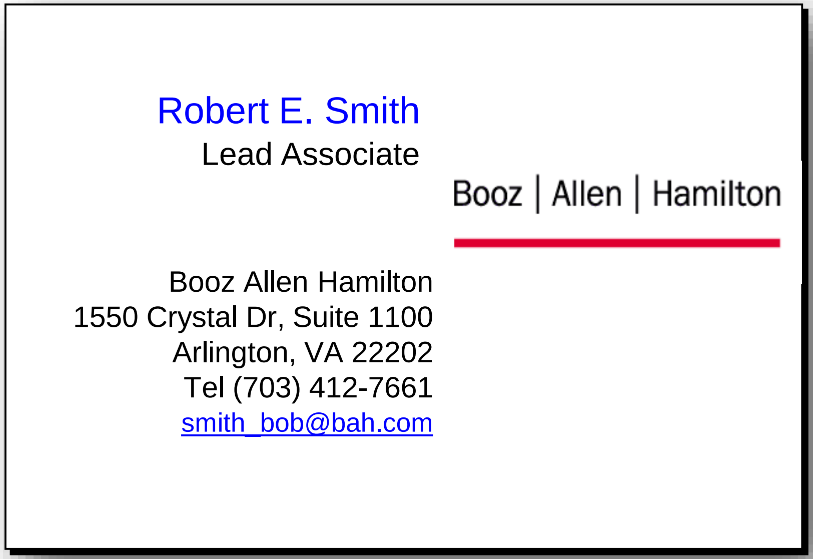
- **DoD Joint Software System Safety Engineering Handbook, 2010**
  - https://www.dau.mil/cop/esoh/DAU Sponsored Documents/SOFTWARE SYSTEM SAFETY HDBK 2010.pdf

- **Software System Safety Implementation Process and Tasks Supporting (a.k.a. “Implementation Guide”)**
  - https://www.dau.mil/cop/esoh/DAU%20Sponsored%20Documents/JSWRBs%20Endorsement%20JS%20SSA%20Software%20System%20Safety%20Implementation%20Guide%2029JUN2016.pdf

JSSSEH IG 2016

JSSSEH IG 2017 Rev A

Questions?



Robert E. Smith
Lead Associate

Booz | Allen | Hamilton

Booz Allen Hamilton
1550 Crystal Dr, Suite 1100
Arlington, VA 22202
Tel (703) 412-7661
smith_bob@bah.com