JOINT SERVICES-SOFTWARE SAFETY AUTHORITIES

Joint Software System Safety Engineering Handbook and MIL-STD 882E Implementation Process and Tasks

Status and Tutorial

Presented at International System Safety Conference San Diego, CA

August 2015

Don Hanline – AMCOM Safety Office
Jay Ball – NAVAIR
Steve Mattern – Bastion Technologies, Inc.
Agenda

• Introduction
  – Overview and Purpose
  – Current Status

• Implementation Process and Tasks
  – Sections 1-2 - Abstract, Task Outline and Process
  – Section 3 - Process and Process Tasks for Software System Safety
  – Section 4 – Acronym List
  – Path Ahead/Notional Schedule
  – Appendix A – Preferred Level-of-Rigor (LOR) Table
  – Appendix B – (TBD) Task Tailoring

• Comments/Recommendation from JS-SSA

• Open Discussion
General Disclaimer

This briefing/tutorial presents an initial draft of one of the ongoing initiatives of the Joint Services – Software Safety Authorities (JS-SSA). As such, nothing in this tutorial should be considered an official position or directive of the JS-SSA or any of its supporting Services.
Purpose

• Condense the content of the 300+ page JSSSEH into a “workable” requirements implementation guide
  – Capture the “requirements” for SwSS from the existing JSSSEH
  – Develop an overarching LOR table with traceability to JSSSEH, MIL-STD-882E, and/or “best practices”
    ➢ Emphasis on “What” – Requirements
    ➢ “How” provided by JSSSEH and MIL-STD-882E
  – Provide consumers with an implementable version of the JSSSEH that meets the SwSS requirements of MIL-STD-882E

...all in 50 pages or less!!!
Introduction

- The Software System Safety Process described represents the defined tasks within the acquisition phases where they should (or should have) occurred – These tasks may require “reality adjustments” for your individual program.

- This is requires the whole “team” to accomplish, especially Subject Matter Experts (SMEs) within the development team (Chief Engineer/SW Architect, SW Lead, Systems Engineering, Safety, Test, QA, Management, Customer, etc.)

- Each Program can tailor the process within the bounds of logic and reason with the Stakeholders review and approval.

- The process and process tasks presented represent what is considered reasonable from a JSSSHE perspective to meet the criteria of MIL-STD 882E

- Each individual process task accomplished must produce the engineering evidence necessary to prove that the tasks were successfully implemented.

If evidence was lacking on a legacy program…it must be produced now!
Overview – Why?

- Contractors often specify they plan to use the JSSSEH as “guidance” for their SwSS program in their SSPP.
  - JSSSEH is called out in MIL-STD-882E (along with AOP-52)
  - JSSSEH is not a Requirements document
  - JSSSEH is a 300+ page guideline/textbook
  - AOP-52 is similar in size and structure, but is required for munitions software
- Number 1 comment against JSSSEH and AOP-52 is that they are “too big”
- Fundamental question: How are the Stakeholder and Developer turning JSSSEH guidance into Software System Safety (SSS) requirements that are defined, implemented, traceable, and auditable?
Current Status

• An initial DRAFT Implementation Guide has been developed and provided to the JS-SSA members for coordination and review
  – JS-SSA feedback has been received and is being reviewed, adjudicated and integrated
  – The graphics/flow diagrams have been updated. SwSS “requirements” are highlighted
  – Two sections to the current document – Main body text (similar to a SSPP) plus a baseline LOR task table
  – Implementation Guide Tasks are Mapped to JSSSEH and MIL-STD-882E
  – Task mapping to NAVAIR Excel LOR tables complete
  – 54 pages in length (36 p. main body, 18 p. LOR table)
Software System Safety Implementation Process and Tasks Supporting MIL-STD-882E

Section 1 – Abstract
Section 2 - Task Outline and Process
Section -1: Abstract

- Software System Safety Engineering (SSSE) focuses on two primary objectives:
  - Design, code, test, and support software with the appropriate LOR to instill a confidence, or the “assurance” of safe software
  - Define the necessary safety requirements for the design, code, and test 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 (JSSSEH).

- Requirements to meet the Software System Safety Engineering objectives are specified in MIL-STD-882E.

- The objective of this document is to provide DOD Customers and contractors with 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.
Section-2: Task Outline and Process

- The document and tasks are organized in a WBS-like format
- The “requirements” embedded within MIL-STD-882E and JSSSEH that must be met are highlighted on the process diagrams/figures
- Requirements to meet the Software System Safety Engineering objectives are specified in MIL-STD-882E.
- The subtasks are described in the document text briefly, but how the subtasks are accomplished is left up to the Stakeholder and software developer. What is presented is one recommended way of meeting the requirements.
- A tailorable recommended LOR task table is provided in Appendix A that can be used to allocate LOR tasks to the responsible teams that will be required to implement and execute them (i.e. Stakeholder, System Safety, Software System Safety (SSS), SW development, Test, SW Quality)
1.0 Software System Safety Abstract

2.0 Specialty Task Outline and Process

3.0 Process and Process Tasks for Software System Safety
   3.1 Process Task 1.0: Prepare System Safety Program Plan (SSPP)
     3.1.1 Process Subtask 1.1: Obtain Inputs from the System Safety Management Plan (SSMP)
     3.1.2 Process Subtask 1.2: Obtain Inputs from Compliance Documents
     3.1.3 Process Subtask 1.3: Prepare the SSPP
     3.1.4 Process Subtask 1.4: Obtain Stakeholder Approval of the SSPP
   3.2 Process Task 2.0: Prepare Level-of-Rigor Table Definitions
     3.2.1 Process Subtask 2.1: Obtain Inputs from Compliance Documents
     3.2.2 Process Subtask 2.2: Prepare Level-of-Rigor Table
     3.2.3 Process Subtask 2.3: Obtain LOR Concurrence from Software Development and Test
     3.2.4 Process Subtask 2.4: Integrate LOR Tasks in Appropriate Contractor Documents
     3.2.5 Process Subtask 2.5: Obtain LOR Approval from Stakeholder
   ...through Section 3.14.6
Section 3.0: Process Tasks for Software System Safety

• It is essential for the DOD Stakeholder to adequately specify the System Safety Engineering (SSE) and SSSE tasks and artifacts necessary to meet the requirements of MIL-STD 882E.

• The Stakeholder must adequately plan for the tasks and requirements that will be required and implemented by the Developer.
  – The Stakeholder’s plan is generally documented in the System Safety Management Plan (SSMP) – Ref. JSSSEH Para. 4.2.1
Section 3.0: Process Tasks for Software System Safety

...Continued

• The Stakeholder must:
  – Develop the SSMP
  – Define/Tailor the Mishap Risk Matrix, SW Criticality Matrix and associated input definitions
  – Charter the SSWG
  – Develop Safety input to the Request for Proposal (RFP), Statement of Work (SOW), and other contractual documentation (Tasks, Contract Deliverable Items (CDRLs), required analyses, and frequency of delivery/update

Required Analyses include: MIL-STD-882E Tasks 102 SSPP, 106 HTS, 201/202 PHL/PHA, 203 SRHA, 204 SSHA, 205 SHA, 206 O&SHA, 208 FHA, 301 SAR. Task 209 SoS Hazard Analysis if applicable
Section 3.0: Process Tasks for Software System Safety

...Continued

• The Stakeholder must:
  – Define Stakeholder specification safety requirements
  – Provide safety requirements to other relevant documentation (ex. TEMP, SEMP)
  – Work with the PM to ensure the safety program is adequately funded and staffed. (Use MIL-STD-882E Table VI)
  – Ensure Stakeholder Safety is part of the Configuration Control Process (MIL-STD-882E Task 304)
  – Perform analyses required to define System Level Mishap and interface/contributions of supporting system elements

If the Stakeholder does not specify the requirements… the Developer will likely not accomplish them
A VERY important activity in SwSS
...but also the one that is generally the least successful
Section 3.1

3.1 Process Task 1.0: Prepare System Safety Program Plan (SSPP)
[Ref: JSSSEH Paragraph 4.2.1, Fig. 4-6, and MIL-STD-882E Task 102]

Each Subtask is briefly described in text sections of the document
System Safety Program Plan

Contractor Contract Language

- Contractor responsible for obtaining an understanding of the best practices and expectations of the Stakeholder
- Best Practices correctly communicated in the contract proposal

Purpose

- Sufficient process, process tasks, and engineering artifacts defined in SSPP
- To establish the agreed expectations of the processes, tasks, and products accomplished by the Contractor
- Again, Contracts will ONLY accomplish what is on contract (“SHALLs”)…period!!

Benefits

- Establish Stakeholder and Contractor expectations early
- Justify resources required to conduct the safety program
- Provides better and more complete requirements to the design teams in the early phases of design
- Cheaper to do it right the first time as compared to mishap cleanup and “fixing the system” later

Evidence Artifact(s)

- Response to the Request for Proposal
- System Safety Program Plan (SSPP)
Section 3.2

3.2 Process Task 2.0: Prepare Level-Of-Rigor Task Table Definitions

[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]
**Level-of-Rigor Definition**

**Benefits**
- Adds structure and discipline to software development processes
- Defines an auditable set of software assurance process tasks and design requirements for safety-critical software
- Proven best-practice approach that is tailorable
- Assists in the production of engineering artifacts required for fielding and/or air worthiness certification

**Purpose**
- Defines and documents the specific process tasks and design requirements deemed appropriate for safety-significant functions
- Establishes a structured and disciplined approach for the design, code, test and verification of safety-significant software
- Note: See LOR Table Version 4.0

**Evidence Artifact(s)**
- Level-of-Rigor Task Table approved by the Stakeholder and the SSWG
Design Phase Process Flow

Process Inputs
- System Baseline Documentation
- User Inputs
  - Similar Systems
  - Mishap Data
- Design Handbooks
- Software Control Categories
  - Software Criticality
  - Severity Definitions

Preliminary Design Review
- LOR Allocation
  - For SCF's
  - Defined Design Requirements
  - Defined Process Requirements

Functional Hazard Analysis
- Safety-Significant Functions

Preliminary Hazard Analysis
- Inputs
- High-Level Safety Requirements

Requirements and Preliminary Design Phases
Software Safety Process
- Top-Level Mishaps
- Prelim Hazard List
- Preliminary Hazard Analysis

Safety Deliverables or Artifacts
- Safety-Significant Functions (SSF) List
- Software Requirements Specifications
- Software Planning Docs
  - SDP
  - STP
  - CMP
  - SQAP

= MIL-STD 882E Requirement
Section 3.3

Process Task 3.3: Preliminary Hazard Analysis (PHA)

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

ID and document in PHL/PHA All potential hazards and TLMs
Document SSWG approved hazards and TLMs in the HTS
Preliminary Hazard Analysis

**Purpose**

- Define the top-level mishaps and their associated hazards of the defined system baseline
- Consideration of defined CONOPS and operational usages
- Identification of initial hazard failure modes and causal factors
- Identification of additional high-level safety mitigation requirements

**Evidence Artifact(s)**

- Compete and Accurate PHA

**Benefits**

- Early identification of system and subsystem level hazards, failure modes, and causes
- Early identification of hazard mitigation requirements for the design team (design, safety systems, warnings/cautions/advisories, and personnel procedures and training)
- Becomes the basis for further safety analyses
PHA Supported by Causal Analysis (based on design maturity)

Analysis to include the software contributions to failure
Section 3.4

3.4 Process Task 4.0: Functional Hazard Analysis (FHA)

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

Safety review authorities expect an FHA as part of the program’s objective evidence to obtain review acceptance and concurrence
Accomplishing the FHA

<table>
<thead>
<tr>
<th>System Decomposition</th>
<th>Functional Description</th>
<th>Individual Functions</th>
<th>Consequence of Failure</th>
<th>Severity of Consequence</th>
<th>Safety-Significant Functions</th>
<th>Assignment of SCC and LOR</th>
<th>Map to Software Design</th>
<th>Failure Mitigation Requirements</th>
</tr>
</thead>
</table>

- If the analyst (or the Stakeholder) requires more than this simple example format can provide, add the appropriate columns to the format to identify and track the information required.
- System decomposition can be done in a WBS-like structure which may aid in structure, flow, traceability and assignment of responsibilities.
- Document failure consequences. 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)
Accomplishing the FHA (continued)

- After all functional failure consequences are adequately identified and documented, each must be assessed against the mishap severity definitions as defined by the SSPP.

- For an individual function, there may be multiple severity consequences, and severity of consequences for that function. The function takes on the worst-case severity consequence as determined by the analysis.

- Those with a safety consequence will be referred to as Safety-Significant functions 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

- Assign a SW control category (SCC), LOR, map functions to SW design and identify failure mitigation requirements
**Functional Hazard Analysis (FHA)**

### Purpose
- Decomposition of the system functionally
- Identification of all system functionality (all subsystems)
- Defines consequences of functional failure, malfunction, or functioning out-of-time/sequence
- Defines the severity of the consequences
- Identifies safety significant functions

### Benefits
- Assesses “Everything” (reduces the likelihood of missing something)
- Identifies and justifies the safety-significant functions of the system
- Mechanism for assignment of Software Control Categories (SCC) to safety-significant functions
- Mechanism for the assignment of Level-of-Rigor (LOR) to each safety-significant function
- Provides insight for Mishap and Hazard identification

### Evidence Artifact(s)
- Complete and accurate FHA
Section 3.5

3.5 Process Task 5.0: LOR Allocations to Safety-significant functions

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

Must be thoroughly documented in the safety analysis artifacts
...Typically integrated in the FHA
A Product of the FHA

Purpose

• The FHA defines all safety-significant (both safety-critical and safety-related) functions of the system

• The term safety-significant is meaningless if there is no willingness to add robustness to the process of design, code, test, and verification of safety-significant requirements

Benefits

• Identifies and justifies all safety-significant functions of the system

• Assesses “everything”...reduces the likelihood of missing something

• Allows for the assignment of SCC and LOR to requirements that are defined to implement the functions

• Allows for Stakeholder and SSWG review and concurrence

Evidence Artifact(s)

• Complete and justified list of safety-significant functions from the FHA

The Requirements that Implement Safety-Significant Functions Shall Also Be Tagged as “Safety-Significant” and Assigned a SwCI/LOR

Presentation for Informational Purposes Only
Section 3.6

3.6 Process Task 6.0: Preliminary Safety Requirements Analysis (SRA)

Ref: MIL-STD 882E, Task 203, and JSSSEH paragraph 4.3.5

System Safety Engineering is a “Requirements-based activity”
Design Engineering only wants our justified safety requirements!
This task is critically important...yet, often not accomplished well or in time!
Categories of Safety Requirements (to analyze)

- Process Task 6.0 is analogous to the Safety Requirements/Criteria Analysis in prior versions of MIL-STD-882
  
  — CSSRs: Contributing Safety-significant Requirements: Requirements whose failure can contribute to a mishap (i.e. Issue Fire Command, Radiate). CSSRs require derived lower level requirements that provide actual mitigations to mishaps
  
  — GSSRs: Generic Safety-significant Requirements: High level HW and SW requirements (ex. JSSSEH Appendix E, Transition to a Safe State)

  — MSSRs: Mitigating Safety-significant Requirements: Requirements that specifically provide mitigation of identified hazard and mishap causes. Requires completion of supporting hazard analyses. Derived requirements (ex. If the unmanned vehicle loses communication with the operator station for more than xx seconds, the vehicle shall stop in place.)
Process Task 6.0 (continued)

— Integrate Hazards from PHA (HTS): Hazards identified in the PHA need to be assessed to determine those that require further, in-depth analysis

— Categorize Hazards with Initial HRI: Hazard Risk Index. Depending on the TLM that the hazard(s) contribute to, the HRI may vary for the same hazard across multiple systems/subsystems

— Derive lower level requirements: Of particular importance in SSS. High level CSSRs, GSSRs, MSSRs must be decomposed into specific SRS requirements, Use Case Steps, etc. to be implementable and verifiable in SW

— Assign Final HRI: After Mitigations identified and mapped to hazards, a “target” Final HRI may be assigned (Final HRI is only after verification)

— Update HTS after each analysis activity
**Safety Requirements Analysis (SRA)**

**Purpose**
- Analysis of all system and subsystem requirements – Tagging contributing safety requirements
- Defining and tagging best practice and early high level safety requirements
- From the PHA and FHA, defining and tagging hazard mitigation requirements

**Benefits**
- Defining safety-significant requirements early in the design phase to allow the designer to mitigate hazards in the design of the system
- Prioritizes the safety requirements based on hazard severity and likelihood of occurrence

**Evidence Artifact(s)**
- Defined safety requirements in the Specifications
- Tagged Safety Requirements in the RTM tool (i.e. DOORS)
- Safety requirements traced back to the hazards that they mitigate
Section 3.7

3.7 Process Task 7.0: Perform In-Depth Hazard Analysis

Ref: JSSSEH paragraph 4.3.6 and MIL-STD-882E Tasks 204-206, and 209
Systems-of-systems
Perform In-Depth Hazard Analysis

Regardless of the SSS analysis techniques used, in-depth hazard analyses must have the ability to allow the analyst to:

- Map or track safety-significant functions 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
- 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)

This includes in-depth causal analysis that becomes the basis for hazard mitigation requirements.
In-Depth Hazard Analysis

**Purpose**

- Account for all top-level mishaps and system-level hazards
- Account for all sub-system-level hazards
- Provides the hazard causal analysis basis to identify and justify all safety-significant requirements to mitigate hazards

**Benefits**

- Provides the engineering evidence that in-depth hazard causal analysis was accomplished and documented, and justifies hazard mitigation requirements
- Provides the engineering evidence that “we’ve done enough” from a safety risk acceptance perspective

**Evidence Artifact(s)**

- Accurate and complete System Hazard Analysis (SHA)
- Accurate and complete Subsystem Hazard Analysis (SSHA)
**Fault Tree Analysis (for prioritized TLMs)**

**Purpose**
- To accomplish top-down, deductive hazard causal analysis
- To “drill-down” through the design to identify failure pathways, failure pathway initiators, and/or failure pathway events
- To provide a mechanism to assess where hazard mitigation is most effective or efficient

**Benefits**
- In-depth hazard causal analysis tool
- Provides for the specific context of failure or failure pathways to hazardous events
- Provides a graphical representation of failure that is easy to communicate to design engineers
- Provides AND and OR Boolean logic to demonstrate and justify what is “enough” to maximize hazard mitigation

**Evidence Artifact(s)**
- Fault Tree Analysis (FTA)
- Other In-depth Hazard Causal Analyses
Section 3.8

3.8 Process Task 8.0: Perform Detailed SRA

Ref: JSSSEH paragraph 4.3.5 and MIL-STD-882E Task 203

This task is not specifically defined as such in MIL-STD 882E
This is basically an update to the existing SRA as a design matures
Section 3.8 (continued)

• Process Task 8.0 ensures that safety requirements analysis is formally completed and adequately documented (and traceable).


  — 8.2 – Do they actually support mishap/hazard mitigation?

  — 8.3 – Testable? Traceable? Function as intended?

  — 8.4 – In lieu of “new” requirements (which Management detests), it may be possible to identify “defects” against existing requirements that can be authored. Also, the primary cause of software defects is poorly defined, ambiguous, unclear, incorrect or missing requirements.

Safety Requirements are the Safety Engineers’ primary product to influence the design
Section 3.9

3.9 Process Task 9.0: Perform Safety Requirements Traceability

Ref: JSSSEH paragraphs 4.3.5.3, 4.3.6.3.3

- Process Task 9.0 is to ensure that there is end-to-end traceability of safety-requirements, from system level capabilities – hazard analysis, design, requirements, implementation and verification
Tracing Safety Requirements

Purpose

• Provide the engineering evidence that safety requirements were designed, coded, tested, and verified in accordance with the level-of-rigor process and tasks

• Provide “confidence” that the system and software is as safe as reasonably possible based on the cost, schedule, and operational constraints of the program

Benefits

• The benefits are the same as the purpose!...plus,

• Provides the basis of residual safety risk assessment

Evidence Artifact(s)

• Software Design Documents

• Software Test Procedures and Test Results

• Software Quality Assurance Artifacts
Test and Deployment Process Flow

Process Inputs:
- Updated Software Specifications
- Software Code (various builds)

Critical Design Review:
- Perform Code-Level Safety Analysis
- Perform Software Test Planning

Software Test:
- Regression Test

Software Defect Reports

TEST PHASE AND DEPLOYMENT PHASES

Software Safety Process:
- Monitor Safety-Significant Testing
- Perform Residual Safety Risk Assessment

TEST PHASE AND DEPLOYMENT PHASES

Safety Deliverables or Artifacts:
- Software Code Final Build
- Software Test Reports
- Traceability
- Hazard Tracking Database
- Safety Case or Safety Assessment Report

Participate in Lifecycle Management

= MIL-STD 882E Requirement

Presentation for Informational Purposes Only
LOR-1 SW requires Code Analysis. NOTE: Code Analysis is not the same as Code Inspection or Code Review (although those may be steps in the Code Analysis process)

— Another reason that performing the predecessor safety analyses is crucial. The opportunities to identify and minimize LOR-1 functionality are before the design is finalized (i.e. early safety influence based upon 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.
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 exception 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
Code Level Analysis (continued)

- Analyze the Safety-critical code. Specific questions to be answered include, but are not limited to:
  - Is the safety-critical code uniquely identified as such in the module header?
  - Is the intended functionality of the software coded correctly?
  - Have the safety-significant requirements been correctly interpreted and coded?
  - Is the timing and sequencing of the functionality correct?
  - Is the logic for safety-significant functionality accurate and as simple as necessary?
  - Have all nominal and off nominal inputs been accounted for?
  - Are safety-critical 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 the safety-critical code’s interfaces with other code and modules compatible?
  - Are the safety-critical functions and code isolated and/or partitioned from non-safety code where required?

- Any errors or software deficiencies discovered in the safety code-level analysis review must be formally documented and submitted to the software development team for defect resolution.
Section 3.11

3.11 Process Task 11.0: Perform Software Test Planning

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

Prioritized testing based on the criticality of the functions …which includes the testing required by LOR assignment
Software Testing

- 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.
  - The “team”, during the program and software development life-cycle, has the opportunity to lower LOR on functions where the design possesses sufficient mitigation to drive a lower SCC
  - Table VI of MIL-STD-882E is used to ensure that System Safety and SSS keep program management apprised of the implications of failure to adequately meet LOR requirements.
  - It is best practice of software test teams to test each software requirement that is documented in the software requirements specification.
  - All SRS safety requirements must be reviewed to ensure the implementation complies with safety design requirements and mapped to test cases.
Section 3.12

3.12 Process Task 12.0: Monitor Safety-significant Software Testing

Ref: Appendix A – LOR Table, Test Phase Tasks

- SSS should work with the Test Team and SW Quality Assurance Team to ensure safety-significant functionality is tested IAW approved test plans. “Redlines” should be reviewed and approved by Safety.
  - SSS cannot reasonably expect to witness all Safety-significant testing.
  - Key to review test plans and ensure Safety-significant functionality is adequately tested IAW LOR.
**Testing of Safety Requirements**

- 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 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

- Any errors or software deficiencies discovered in the safety code-level analysis review must be formally documented and submitted to the software development team for defect resolution.
Review of Final Software Test Results

• Upon the completion of defined software test cases and procedures, 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 hazard causal factors,
  
  – The software test anomalies and defects are adequately identified, documented, and rectified.
  
  – The software defect resolutions are adequately regression tested.

• SSS must integrate this evidence into the hazard control portion of the HTS.
Verification of Safety Requirements Implementation

**Documented Evidences**

**Purpose**

- Account for the engineering evidence that mishaps and hazards have been mitigated and controlled through the implementation and verification of safety-significant requirements

**Benefits**

- Provides the engineering evidence that in-depth hazard mitigation was accomplished and documented
- Provides the engineering evidence to accurately support the residual safety risk statements for materiel release and air worthiness certification

**Evidence Artifact(s)**

- Accurate and complete SSHA and SHA records that account for and document the traceability of mitigation to implemented and verified safety requirements

*Presentation for Informational Purposes Only*
Section 3.13

3.13 Process Subtask 13.0: Perform Residual Risk Assessment

Ref: DoDI 5000.02, MIL-STD-882E, Task 301, Appendix B and JSSSEH paragraph 4.4.4

MIL-STD 882E, Task 301
- Perform Residual Safety Risk 13.0 Assessment
  - Reassess All Documented Hazards 13.1
  - Verify Mitigations 13.2
  - Assess Partial or Failure to Mitigate 13.3
  - Assess Residual Safety Risk 13.4

MIL-STD 882E, Task 304
- Participate in Life Cycle 14.0 Management
  - Document and Communicate Safety Risk 13.5

Presentation for Informational Purposes Only
Residual Risk Assessment

- Each Service has its own residual risk documentation and acceptance process:
  - Even for Joint Programs, each Service accepts residual risks separately
  - Mishap and hazard records in the HTS must be “closed” and the residual risks agreed to and accepted
  - Mishaps and hazards that don’t have verified controls/mitigations for causes must be assessed as residual risks
- The “amount” of residual risk is dependent on a number of factors, such as:
  - Mishap severity
  - Number of other, independent verified controls that mitigate mishap
  - Boolean relationship of mitigations (.AND. vs .OR.)
  - Ability of system to detect, isolate and recover from failure
Section 3.14

3.14 Process Subtask 14.0: Life-cycle Management and Support

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

This life cycle phase is where many safety engineers reside. If the analyses was not completed previously...it’s time to do it NOW!
System Sustainment

• System safety and SSS requirements don’t end with initial fielding:
  – SW, in particular, will constantly be modified throughout the post-fielding life-cycle of the system
  – Changes include correction of defects/deficiencies identified, upgrades and new capabilities, evolutionary development, pre-planned SW drops

• SSS must be familiar with, and an active participant, in the configuration management process.

• Each evolution of software must be assessed for safety impact and may be subject to Service or JS-SSA review requirements

• SSS must review every change request pertaining to the system software and provide input to the System Safety representative on the CCB.

At a minimum, the safety engineer must be a member of the Configuration Control Board (CCB) with signature authority over all Engineering Change Proposal (ECP) Actions
Review of ALL Software Trouble Reports

**Purpose**

- Ensure that system safety and software system safety engineers review ALL STR’s to assess the applicability to mishaps, hazards, and safety risk
- To identify those STR’s that have safety applicability for safety analysis
- To initiate and complete safety hazard analysis on all safety-applicable STR’s

**Benefits**

- Provides the justification basis of safety risk applicability for each STR
- Limits the safety engineering tasks to only those with safety applicability
- Ensures that safety risk applicability is communicated to the system and software designers
- Ensures that new hazard causal factors are not introduced
- Ensures that previous hazard causal factors remain mitigated as intended

**Evidence Artifact(s)**

- Safety analysis of each STR
- Rationale for/against safety applicability of STRs
System Sustainment (continued)

• The system safety and software system safety processes still apply to the Life-cycle support phases:
  – Assess all changes to the system
  – Assess changes against documented hazards (includes closed hazards and hazards that have accepted residual risks)
  – Identify and document any new mishaps, hazards, causes, or mitigations
  – Ensure safety-significant functionality is tested (or regression) tested IAW LOR
  – Document and communicate safety risk (includes new residual risk acceptance)
  – Update applicable safety analyses and artifacts

Ensuring that adequate safety resources are applied during the “sustainment” phases of a program is one of the great challenges. Typically, during sustainment, budgets are reduced and resources are focused on design, development and production.
**Tailoring the LOR Table** *(example)*

<table>
<thead>
<tr>
<th>Level of Rigor (LOR) Activity</th>
<th>Primary Responsibility</th>
<th>Support Responsibility</th>
<th>Level-Of-Rigor</th>
<th>Representative Artifacts Produced</th>
</tr>
</thead>
<tbody>
<tr>
<td>defects against existing high-level safety-significant Requirements. Section 3.6.4 [Best Practice]</td>
<td>Contactor System Safety</td>
<td>Contractor Software Safety Contractor Software Design</td>
<td>R</td>
<td>Safety Requirements</td>
</tr>
<tr>
<td>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]</td>
<td>Contactor System Safety</td>
<td>Contractor Software Safety</td>
<td>R</td>
<td>RTM Tool Update Software Design Artifacts</td>
</tr>
<tr>
<td>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 JSSEH, etc) [Directed Best Practice]</td>
<td>Stakeholder System and Software Safety Contractor System and Software Safety</td>
<td>Contractor Software Design Stakeholder SSWG Review and Approval</td>
<td>R</td>
<td>As directed Assessment of Compliance Artifact</td>
</tr>
<tr>
<td>DP-6: Review of the user interface design for safety-significant issues [Best Practice]</td>
<td>Contactor System Safety Contractor Software Safety</td>
<td>Contractor Hardware and Software Design Contractor Human Factors</td>
<td>R</td>
<td>Assessment of User Interfaces with Software Functionality</td>
</tr>
<tr>
<td>DP-7: Create traceability from all safety-significant requirements to the design architecture Section 3.6 [Best Practice]</td>
<td>Contractor Software Design</td>
<td>Contractor Software Safety</td>
<td>R</td>
<td>Safety Requirements-to-design Traceability</td>
</tr>
</tbody>
</table>

LOR “Tailoring” Accomplished in the LOR Columns …Not the elimination of Row Tasks
**Purpose**

- To capture and integrate all Level-of-Rigor process tasks for the design, code, test, and verification of safety-significant functions/requirements into the defined software planning documents
- Ensure agreement between the safety team and the software development and test team

**Benefits**

- Establishes an integrated software development and test, and system safety process for developing safety-significant software
- Established and structured and disciplined approach to be planned and priced during bid & proposal activities
- Allows for an auditable process by software quality assurance
- Defines a proven standard practice for fielding and air worthiness certification

**Evidence Artifact(s)**

- LOR integrated in the SDP
- LOR integrated in the TDP
Path Ahead with Notional Schedule

– Develop “final” JS-SSA version of the document
  • Obtain concurrence from the JS-SSA
  • Add detailed guidance on tailoring (Appendix B)
– Determine what the end game document will be (Requirements, Guidelines, Replace JSSSEH, Recommendations, etc.)
  • Impacts Level-of-Effort, Review, Time, Schedule, Approval, etc. – i.e. Cost and who will fund the effort?
– Send JS-SSA version out for next level (TBD) review
– Tentative goal of end of CY 2015 for next draft
– Possible update tutorial/panel at 2016 ISSC (TBD)
Key (Critical and Substantive) JS-SSA Review Comments

- Summary of items requiring adjudication before a draft for “public release”

<table>
<thead>
<tr>
<th>Issue Summary</th>
<th>Required Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instances of out-of-sequence events in the process flow charts</td>
<td>Correct Process Flow Charts and corresponding text</td>
</tr>
<tr>
<td>Some differences between the scope and limits of the analysis vs. what MIL-STD-882E tasks ask for</td>
<td>Clarify areas where inputs to MIL-STD-882E requirements (ex. RAC in FHA) are not artifacts of the analysis (ex. FHA doesn’t assess probability) and how differences can be bridged</td>
</tr>
<tr>
<td>Integrate the update of the Hazard Tracking Database after each major analysis task</td>
<td>Correct missing HTS update in the Process Flow Charts and corresponding text</td>
</tr>
<tr>
<td>Numerous instances of using safety terms from sources other than MIL-STD 882E</td>
<td>Correct terminology to be consistent with MIL-STD 882E</td>
</tr>
<tr>
<td>Numerous instances where rewording of a sentences or paragraph would bring clarity to the reader</td>
<td>Reword sentences or paragraphs to bring clarity to the reader</td>
</tr>
<tr>
<td>Numerous minor editorial corrections</td>
<td>Final editing of the text</td>
</tr>
</tbody>
</table>

There were no submitted comments that questioned our sanity or philosophy!

Presentation for Informational Purposes Only