Software Safety Analysis

Procedures

International System Safety Training Symposium

St. Louis, Missouri

4 - 8 August 2014

Authored by:
Ms. Carolyn Tilghman
Ms. Mei Chu Li
Naval Surface Warfare Center, Dahlgren Division
Engagement System Safety Branch (G71)
Dahlgren, Virginia 22448-5155

Mr. Michael Zemore
Naval Surface Warfare Center, Dahlgren Division
Chief Engineer, Systems Safety Engineering Division (G704)
Dahlgren, Virginia 22448-5155

Software Safety Analysis

Purpose:

The main objective of system safety engineering, which includes software system safety, is the application of engineering and management principles, criteria, and techniques to optimize all aspects of safety within the constraints of operational effectiveness, time, and cost throughout all phases of the system life cycle.

It is essential to perform system safety engineering tasks on safety-significant systems to reduce safety risk in all aspects of a program. These tasks include software system safety activities involving the design, code, test, Independent Verification and Validation (IV&V), operation and maintenance, and change control functions within the software engineering development and deployment processes.

A software specification error, design flaw, or lack of initial safety design requirements can contribute to or cause a system failure or erroneous human decision. Death, injury, monetary loss, or environmental damage can result. To achieve an acceptable level of safety for software, software safety engineering must be emphasized early in the functional definition, requirements definition, and system conceptual design process. Safety-significant software must then receive continuous emphasis from management and a continuing integrated engineering analysis and testing process throughout the development life cycles of the system.

The Software Safety Analysis Process, when executed, identifies the specific contributions of the software to cause, influence, contribute to, or mitigate system level hazard and mishap occurrence. Figure 1 shows the high level process. The following sections describe the tasks contained in the process.

Methodology:

A successful software system safety engineering effort is based on an effective system level hazard analysis process that definitively identifies safety-significant software and mishap contributors, a software safety analysis process that adds rigor and robustness for safety significant software, and thorough execution of safety analysis and verification tasks defined as Level of Rigor (LOR). Emphasis is placed on the context of the "system" and how software contributes to or mitigates failures, hazards, and mishaps. In most instances, the software safety engineers will perform the software safety analysis and verification process in conjunction with the software development, software test, and IV&V team(s) to ensure that the implementation of the safety-significant requirements has not introduced additional safety concerns. The software safety analysis and verification process identifies software contributors to hazards and provides mitigation recommendations to the appropriate risk management authority. Executing the required LOR tasks increases the confidence that the software will perform as specified, while reducing the number of contributors to hazards that may exist in the system and reducing the likelihood of software initiating a propagation pathway to a hazardous condition or mishap (Military Standard (MIL-STD)-882E).

The goal of the System Definition and Software Safety Planning is to establish the System/ Software Safety Program by defining the program safety scope and tasks. This task is the same as

definition and planning for non-software safety programs with the inclusion of the following process tasks.

A necessary part of establishing the scope of the software safety program is to Determine Software Criticality Index (SwCI) for each system software Safety-Significant Function (SSF). The previously conducted system Functional Hazard Analysis (FHA) identifies SSFs early in the design process. The FHA is not a software unique task or tool. Each SSF in the FHA that has software interaction is assessed to determine the mishap severity to which it might contribute based on the available system definition. Then each of these software functions are assessed to determine the Software Control Category (SCC) as defined in MIL-STD-882E, *Department of Defense Standard Practice System Safety*, Table IV. Each function may have a variety of SCCs throughout different Computer Software Configuration Items (CSCIs) or subsystems within the system. Together the severity and SCC determine the SwCI and required safety analysis and verification tasks based on MIL-STD-882E, Table V, Level of Rigor. Each possible task is discussed below, but the LOR determines if it is conducted for a particular function.

The Software Requirements Hazard Analysis (SRHA) identifies and derives software safety requirements to safely implement functionality and mitigate hazards in support of the preliminary design. Perform an SRHA, applicable to the software program being developed, to include developing mitigation requirements and generic safety requirements. Update the Safety Requirements Verification Matrix (SRVM) with verification strategies and requirements.

Software Architectural Hazard Analysis is typically a pre-Preliminary Design Review task conducted on available architectural documents and requirements. The highest level system requirements define the minimum performance specifications for the collective system. All aspects of the system must be harmonized to most effectively produce a satisfactorily implementation. Software architecture defines constrains to the possible software implementations. Typically software architecture defines the software structure, including components and interfaces, and the intended rules of software execution. The safety practitioner should focus on the parts of the architecture that would prevent or hinder implementation of safety requirements as identified in the SRHA. Generic safety requirements from the *Joint Software System Safety Engineering Handbook (JSSSEH)* and Allied Ordnance Publication (AOP)-52 that do not appear in the current system requirements are also evaluated in the same fashion. In the Hazard Tracking System (HTS), each hazard developed from the Preliminary Hazard Analysis (PHA), FHA, and SRHA that embodies the computing or software architecture, or relies on the architecture as mitigation, is updated with the architecture analysis results. If the architecture eliminates the reasonable implementation of an existing mitigation called out in a hazard record, that mitigation is removed as an existing mitigation and the risk assessments must be updated to reflect. Such a mitigation may be a candidate for a recommended architectural change or a new mitigation may be suggested based on the architectural analysis results. As with the other hazard analyses, architectural analysis also seeks out where the architecture may introduce additional hazards or causal factors. Software Architectural Hazard Analysis is addressed in the JSSSEH, Section 4.3.6, Preliminary Software Design, Subsystem Hazard Analysis (SSHA).

Software Design Hazard Analysis is typically a pre-Critical Design Review task which improves on the results of the architectural analysis by considering the specific planned implementations. Each pre-existing hazard in the HTS is reviewed to determine if design choices have provided additional mitigations or precluded any currently documented mitigation. The generic safety requirements are a

source for recognizing parts of the design which may create additional hazards, causal factors, and mitigations. Software design analysis stops short of actual implementation at the code level. Implementation analysis is conducted as part of code analysis. Software Design Hazard Analysis is addressed in JSSSEH in Section 4.3.7, Detailed Software Design, SSHA.

Code is a sequence of instructions written using some human-readable computer language to execute a specified task with a computer. Code Level Hazard Analysis is performed to analyze program code and system interfaces for events, faults, and conditions that could cause or contribute to hazards. Using the results of the Software Design Hazard Analysis, the analyst begins with an analysis of the code to verify the integrity of all safety-significant data variables while ensuring that the software exhibits strong data typing for all safety-significant variables. The analyst also analyzes the error processing implementation and error handling associated with internal processing and interface message traffic to identify any potential failure modes that would result in undesirable results including safety-significant data erroneously used or transferred. A variety of code analysis techniques (e.g., a forward flow technique is a forward trace of the flow of execution of the software in response to the occurrence of an event to see if a causal factor, hazard, or mishap can occur; a backward flow technique is a backward trace from unwanted hazardous events to see if software can cause an event to happen under credible circumstances) could be used to analyze the code. Refer to JSSSEH, Section 4.3.7 for additional analysis techniques.

Operator Documentation Safety Review provides safety review and inputs to user documentation (e.g., Safety Note(s), Warnings, Cautions, Advisories, Workarounds, etc…) to identify new hazards and evaluate adequacy of procedures used to mitigate/control hazards.

Software Safety Testing and Verification verifies and validates all the testable Software Safety Significant Requirements, in accordance with their assigned LOR, through Software Developmental and System Integration Testing, verifies the implementation of mitigations, uncovers safety anomalies, and identifies and/or verifies the resolution of hazards.

Formal Review provides evidence to show that the system and software safety analysis approach addressed the necessary analytical and verification tasks commensurate with software control and mishap potential, all remaining software risk in the system for all operational environments is well understood, and all risk is accepted by the appropriate authority as required. A formal review typically occurs early in system safety program definition and prior to major milestones including test firings and operational use.

Sub Processes outlined in the Software Safety Analysis process include:

Defect Resolution is an analysis process that covers the review of all issues and enhancements documented against the system to identify potential safety impact, define associated risk, and develop safety mitigation.

Regression Testing ensures that system changes to resolve defects or introduce new functionality has not invalidated previously verified safety functionality or introduced new hazards.

Fleet Anomaly Reporting ensures all feedback from the user is evaluated for safety impact. All feedback with safety impact initiates a safety assessment where risk is defined and mitigations are devised.

References:

1. AOP-52, Edition 1, *Guidance on Software Safety Design and Assessment of Muniton-Related Computing Systems Ratification, Draft 1*.

2. Joint Software System Safety Engineering Workgroup, *Joint Software System Safety Engineering Handbook, Version 1.0*, Published 27 August 2010.

3. MIL-STD-882E, *Department of Defense Standard Practice for System Safety*, 11 May 2012.

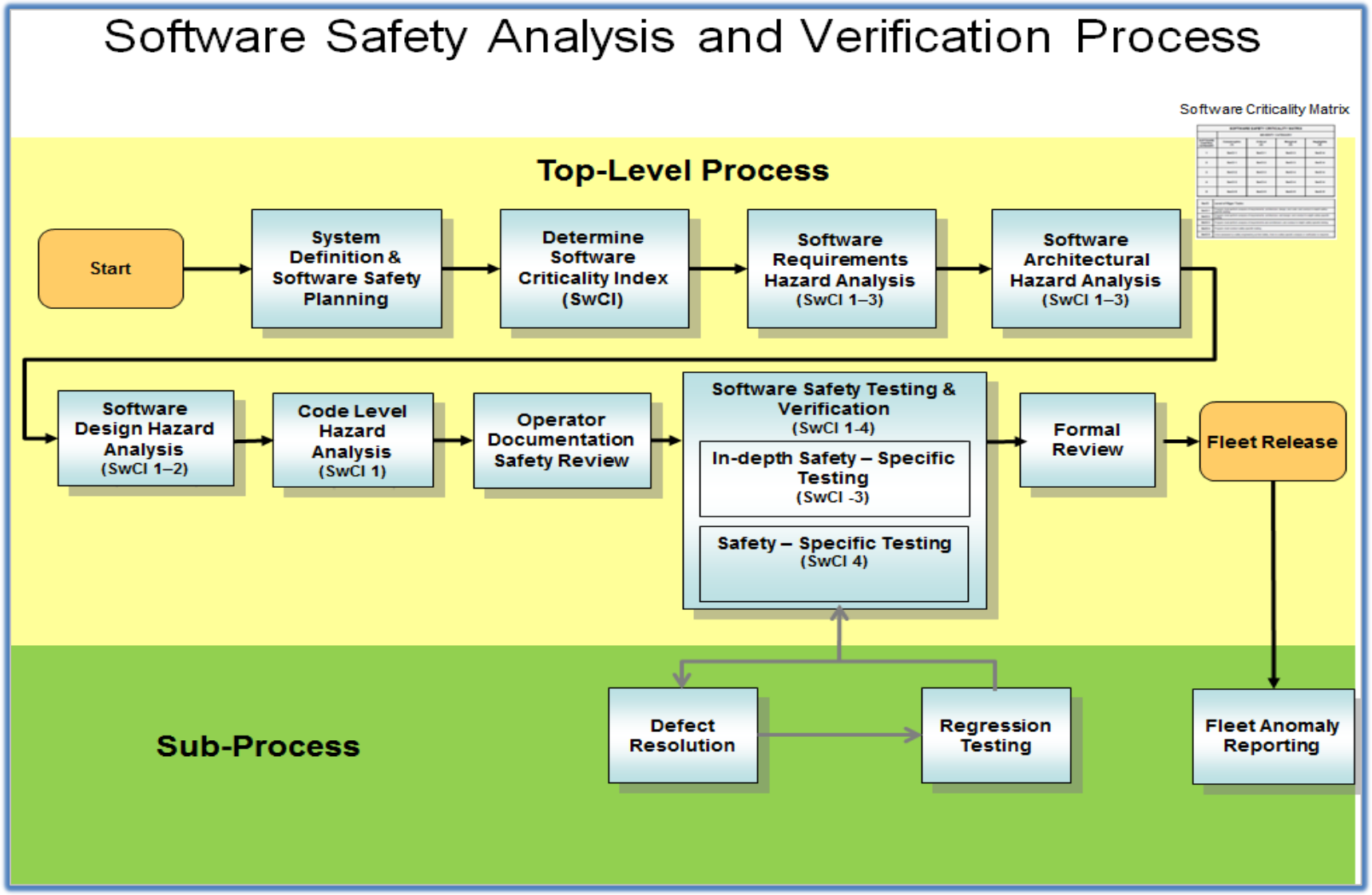

Figure 1. Software Safety Analysis Process

Table 1. Software Safety Analysis Process Tasks

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
1. System Definition and Software Safety Planning.	1a. Establish the System/Software Safety program and define program safety scope.		• Develop System Safety Project Planning to develop/tailor a System Safety Management Plan (SSMP) and System Safety Program Plan (SSPP). • Ensure Software Safety is included. If required, update documents for Software Safety tasking.	• Software Safety Analysis and Verification Process
1b. Conduct a Functional Hazard Analysis (FHA). Trace Safety-Significant Functions (SSFs) to software functions for each subsystem.	1b.1. Gather each subsystem Computer Software Configuration Item (CSCI) list and functional software descriptions. Review the list of system level SSFs. For each SSF, use system documentation (e.g., System/Subsystem Specifications (SSS), Critical Item Development Specification (CIDS), System/Software Design Document (SDD), etc.) to define any specific portions within the SSF that require functional flowdown. For each SSF or portion within the SSF, flow the SSF (or portion) to the subsystem CSCI(s) and subsystem software function(s).	Done as part of the FHA or Preliminary Hazard Analysis (PHA). If not performed in these analyses, it is required before continuation with software safety criticality assessments.	• Initial FHA should be conducted during the System Definition and Software Safety Planning. • The associated subsystems software functions are those involved in performing the function. • This identifies all software functions within each subsystem (and their location via CSCI). The resulting software functions are considered Safety Significant Software Functions.	• PHA • FHA • List of SSFs • Requirements - to assist in defining Portions within a SSF • List of Subsystem CSCI • Software Functional Descriptions

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
2. Determine the Software Criticality Index (SwCI) for each software function, associated safety analysis and verification tasks (i.e., Level of Rigor (LOR)).	2a. Record the SSF that will be evaluated and flowed to the software function. You will do this for each SSF to ensure complete assessment of all SSFs for software contributions.	Record in the Software (SW) Criticality Analysis Worksheet (provided in the tutorial).		• SSFs • Specifications • Software Safety Analysis and Verification Process
	2b. Determine if a Portion of the SSF will be assessed (typically determined with the aid of specifications). If only a portion (given the SSFs are often written in broad terms), record the portion of the SSF that will be assessed.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).		• FHA • SSFs • Specifications
	2c. For each SSF or SSF Portion, record all CSCIs involved.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• There could be multiple CSCIs that implement the portion of the SSF being addressed. Ensure each CSCI is recorded so each will be analyzed appropriately based on the final SwCI.	• FHA • CSCI Descriptions • Software Functional Descriptions • SDD

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
2. Determine the Software Criticality Index (SwCI) for each software function, associated safety analysis and verification tasks (i.e., Level of Rigor (LOR)). (Cont'd)	2d. Using the Functional Flow Diagram (or other data from the FHA), record all software functions within the CSCI involved in the portion of the SSF.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• Each Software Function recorded will be considered a Safety Significant Software Function. • Ensure each software function within each CSCI is recorded. • Ensure all software functions are recorded for each CSCI (there should be no instances of CSCI listed without an associated software function or several software functions. • A good understanding of the software functionality is necessary for this step.	• FHA • SSFs • CSCI Descriptions • Software Functional Descriptions • SDD • Specifications
	2e. Using system hazard worksheets (typically from the PHA or contents of the Hazard Tracking System (HTS)), identify the hazards associated with the specific SSF portion. Record the hazard record number(s) from Hazard Worksheet.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• Review a list of hazards. Reading the hazards descriptions, determine which hazard relates to the Software SSF and record the hazard record number. • Utilize the information on the worksheet if it calls out the SSF number. • A good understanding of the software functionality is necessary for this step. • An analytical approach is necessary for this step as the analyst is determining, based on the software functional description, if faulty processing and poor design could result or contribute to one or many hazards.	• PHA • FHA • SSF Portion description (derived in Step 2b) • HTS • CSCI Descriptions • Software Functional Descriptions

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
2. Determine the Software Criticality Index (SwCI) for each software function, associated safety analysis and verification tasks (i.e., Level of Rigor (LOR)). (Cont'd)	2f. Using the hazard worksheet, identify the hazards associated with the specific SSF portion. Record severity from Hazard Worksheet.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• From the hazard record(s) identified in Step 2e, record the severity level associated with the mishap occurrence (i.e., severity portion of the Mishap Risk Index (MRI)). If multiple hazards are identified, ensure all considerations are included within the final assessment of the severity. It is likely the severity will be worse case, but one could also segregate the function even more to allow definitive software criticality indices for the software functions. • Utilize the information on the worksheet if it calls out the SSF number.	• PHA • FHA • HTS • CSCI Descriptions • Software Functional Descriptions
	2g. Determine Software Control Category for each software function associated with the SSF or SSF thread.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• Define the level of control each function has for that SSF or SSF thread. • Review the description of the software function and understand the software function to the point where you can assess its level of control and autonomy over energy release. Control assessment based on independence, redundancy, safety interlocks, intervention, etc. are typically evaluated at the system level in determining the Software Control Category (SCC) for the software function. It will not always be clear without ambiguity, so typically the analyst will rely on the application of sound engineering processes and judgment in defining the SCC. • Repeat this process for each software function defined within the SFF portion.	• List of SSFs • Software functional descriptions • Military Standard (MIL-STD)-882E, SCCs, Table IV

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
2. Determine the Software Criticality Index (SwCI) for each software function, associated safety analysis and verification tasks (i.e., Level of Rigor (LOR)). (Cont'd)	2h. Determine the SwCI for each of the safety significant software functions using MIL-STD-882E.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• Given the MIL-STD-882E Software Safety Criticality Matrix (SSCM), define the resultant SwCI for each safety significant software function, using Level of Mishap Severity and SCC recorded in the worksheets. • Repeat this process for each software function defined within the SFF portion.	• MIL-STD-882E, SSCM, Table V
	2i. Determine the Level of Rigor and associated safety analysis and verification tasks for each software SSF.	Record in the SW Criticality Analysis Worksheet (provided in the tutorial).	• Given the SwCI for each software SSF, define the associated LOR for each software SSF and the specific analysis and verification tasks listed in MIL-STD-882E. • Repeat this process for each software function defined within the SFF portion.	• MIL-STD-882E, SSCM, Table V, LOR Tasks
	2j. Evaluate the Software Safety Literature Compilation document (provided in the tutorial) to assist with architecture and implementation safety decisions.	Programs HTS.	• Review the Software Safety Literature Compilation document to address architecture and implementation strategies for risk assessment and communication to the software engineering teams.	• Software Safety Literature Compilation document
3. Perform Software Requirements Hazard Analysis (SwCI 1-3).	3a. Perform Software Requirements Hazard Analysis (SRHA) for each software function assessed as SwCI 1-3.		• Develop SRHA for software system/subsystem. • If SRHA has been developed, ensure Software Safety requirements are included.	• Specifications • Software Safety Analysis and Verification Process

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
4. Conduct Software Architectural Hazard Analysis (SwCI 1-3).	4a. Collect available architectural documentation and diagrams for the subsystem of the software function being analyzed.		• Look for documentation that explains what the solution constraints are. • Architectural Analysis is performed on available documents (requirement and architectural views) and generic requirements.	• SSFs Interface and Function Architecture Diagrams • Specifications • Subject Matter Experts (SMEs) • Software Safety Analysis and Verification Process
	4b. Determine Architectural concepts planned for generic requirements.			• SRHA • Joint Software System Safety Engineering Handbook (JSSSEH) Generic Requirements (Appendix E) • Allied Ordnance Publication (AOP)-52
	4c. Review all architectural data against current hazards and functional hazard analysis to determine if additional risk is created or if risk is adequately mitigated by the architecture.	Record as a software Causal Factor or hazard in the programs HTS.	• For each applicable hazard and SW Generic Requirement, review architecture for additional causal factors by: Selecting a generic requirement from list and compare to applicable Architecture Diagram. • Determine if the architecture depicted supports or prevents implementation of the generic requirement. • Determine if any casual factors to existing hazards or new hazards are created. • If required, derive new software safety requirements; provide safety updates to specifications as required. See SRHA training for guidance on safety requirements.	• FHA • PHA • SRHA • Safety Requirements Verification Matrix (SRVM) • HTS • SSFs Interface and Function Architecture Diagram

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
5. Conduct Software Design Hazard Analysis (SwCI 1-2).	5a. Collect available design documentation.		• Documentation that explains how the architectural approach was implemented within the design. Not the specific code, but implementation approach or details of the approach.	• PHA • Updates to Specifications • SW Design Document • SMEs • Software Safety Analysis and Verification Process
	5b. Determine design details planned for generic requirements.	Record as a software Causal Factor or hazard in the programs HTS.	• Based on SRHA and the software safety activities to date. Assess design plans for potential casual factors to existing hazards, new hazards.	• JSSSEH
	5c. Review all design data against current hazards and previous architectural analysis to determine if additional risk is created or risk is mitigated by the design details.	Record as a software Causal Factor or hazard in the programs HTS.	• The approach is to use a conceptualized Control Flows that the safety engineer used during the architectural analysis and compare to the more detailed design flow that lays out implementation plans for that functional area. Examples include internal and external interfacing, interrupts and handling of faults. The intent is to ensure the design captures all safety consideration from the architectural flow without the addition of causal factors or hazards. • Ensure the design supports the expected concept of execution (e.g., Safety approved architectural control flow diagram). • If causal factors or hazards are created by not supporting the concept of execution previously planned, record as a causal factor or hazard in the program's hazard tracking system.	• Previously Reviewed Architecture Design Documents • FHA • PHA • SRHA • SRVM • HTS • SSFs Design Diagrams

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
6. Conduct Code Level Hazard Analysis (SwCI 1).	6a. Assess the variety of code analysis techniques available and define those planned for the analysis.		• Refer to JSSSEH and AOP-52 for techniques. • Evaluate the availability, cost and effectiveness of Code Analysis Software tools prior to the decision to use. • Analyze code to identify hazards and causal factors and ensure compliance with software safety requirements. • Analyze code to verify and validate safety functionality and adequacy of mitigations.	• Software Code • JSSSEH • AOP-52 • Software Safety Analysis and Verification Process
	6b. Conduct Data Structure Analysis (a technique common and typical in code analysis).	Data Structure Analysis Worksheet within SW Code Analysis Worksheet (provided in the tutorial).	• Review safety significant data items to verify consistency and accuracy of data items used in software code. • Data item, either Global or Local can be determined in the variable declaration section of the code. • Review Applicable Hazards and determine what could happen if the data items get corrupted.	• Process Flow Diagram • Software code • Safety-Significant Data Items • HTS • SRVM • JSSSEH 4.3.7.3.2.1 • AOP-52 • Standard Agreement (STANAG) 4404
	6c. Conduct Data Flow Analysis - For each safety significant data item (a technique common and typical in code analysis).	Data Flow Analysis Worksheet within SW Code Analysis Worksheet (provided in the tutorial).	• Understand how the data items are used and how the data are passed between functions/routines. • Review how used to identify errors in the use of data that is accessed by multiple routines.	• Process Flow Diagram • Software code • HTS • SRVM • JSSSEH 4.3.7.3.2.2 • AOP-52 • STANAG 4404

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
6. Conduct Code Level Hazard Analysis (SwCI 1). (Cont'd)	6d. Conduct Compliance Checklist. For the given function, identify a non-compliance issue and record the rationale and potential safety impact.	Code Compliance Assessment Worksheet within SW Code Analysis Worksheet (provided in the tutorial).	• Tailor Code Compliance Checklist as needed. • Identify in the code which area are not in compliance with the coding standard. • If not in compliance, potential safety impacts exist, i.e., causal factors and/or hazards that can lead to a mishap. • Identify and propose code change for compliance.	• Process Flow Diagram • Safety significant software code • AOP-52 • STANAG 4404
	6e. Conduct any additional code analysis techniques as defined in Step 6.a.		• Record any identified causal factors or hazards in the program's hazard tracking system.	
7. Perform Operator Documentation Safety Review.	7a. Review operator documentation to identify new hazards.	Programs HTS.	• Review computer program operator's manual(s) and software-related procedural workarounds. • Identify hazards associated with procedures.	• Software requirements • SRVM • Software Safety Analysis and Verification Process
	7b. Review operator documentation to ensure adequacy of procedural controls.	Programs HTS.	• Review computer program operator's manual(s) and software-related procedural workarounds. • Identify discrepancies and hazards associated with user workarounds.	• Software requirements • SRVM • G70 Software Safety Analysis and Verification Process
	7c. Assess impact of workarounds.	Programs HTS.	• Using results from 7a and 7b, as well as any software-related operator hazards identified in prior analyses, perform an assessment of the accumulation of hazards and/or workarounds and the impacts to the operator. • Provide mitigations to address the hazard (e.g., documentation revision, design modification, etc.). • Update hazard tracking system as warranted.	• Software requirements • SRVM • PHA • FHA • Software Safety Analysis and Verification Process

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
8. SW Safety Testing and Verification (SwCI 1-4).	8a. Determine safety testing required (safety-specific, or in-depth safety testing) in accordance with the SwCI for the SW under test.		• Use AOP-52 and the JSSSEH as guides in determining the appropriate testing necessary to support the SwCI. • Safety should be involved in all levels of testing and integration. • Safety testing required to identify new hazards and causal factors while verifying safety requirements and adequacy of mitigations. • Ensure use of SRHA and Subsystem Hazard Analysis (SSHA) to fully verify all safety requirements.	• MIL-STD-882E, Table V • JSSSEH • AOP 52 • Software Safety Analysis and Verification Process
	8b. Support Software Safety Testing Events.	Programs HTS.	• Safety input is required to the test plan and test procedures to ensure adequate test and verification of safety significant software. • Participate in test readiness reviews. • Safety practitioners may actively participate in the actual testing or serve as a witness thereto. • Record any test approach, objectives, issues or concerns within the HTS.	• Software Safety Significant Requirements • Software Safety Analysis and Verification Process
	8c. Analyze test results.	Programs HTS.	• Verify SW Safety Significant Requirements in test results/data. • Verify resolution of previously identified anomalies, if warranted. • Identify/document anomalies in test results/data. • Evaluate anomalies for safety impact; assess risk associated with safety anomalies.	• Software Safety Analysis and Verification Process

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
8. SW Safety Testing and Verification (SwCI 1-4). (Cont'd)	8d. Document Software Safety test results.	Programs Safety Requirements Verification (and Validation) Matrix. Programs HTS.	• Update SRVM with software safety verification results. • Update hazard tracking system with details of any failed safety requirements verifications including updated risk assessment. • Update hazard tracking system with details of any inadequate mitigation including updated risk assessment.	• Software Safety Analysis and Verification Process
9. Formal Review.	9a. Document software contribution to final safety risk in Mishap Assessment Reports (MARs), and obtain risk acceptance.	MAR form.	• Provides documented evidence that the software contribution to system risk is defined, and that risk is accepted by the appropriate authorities.	• Department of Defense Instruction (DoDI) 5000.02 • Naval Sea Systems Command Instruction (NAVSEAINST) 5000.08 • NAVSEAINST 5100.12B • NAVSEAINST 8020.6E • Software Safety Analysis and Verification Process
	9b. Develop Technical Data package in support of formal review.		• Include software safety analysis results and risk contributions in data packages supporting the review. • Provide safety support at other reviews, as required (e.g., formal software acceptance panels, certification panels, mission readiness reviews, etc.). • Respond to actions/findings from review authorities, as required.	• DoDI 5000.02 • NAVSEAINST 5000.08 • NAVSEAINST 5100.12B • NAVSEAINST 8020.6E • Software Safety Analysis and Verification Process

Table 1. Software Safety Analysis Process Tasks (Continued)

Task	Subtask	Worksheet/Template (if applicable)	Considerations	Reference
Sub-Processes				
10. Defect Resolution. (Sub-process to Software Safety Testing and Verification)	10a. Review each anomaly for safety.	Programs HTS. Programs SRHA and SRVM.	• Analyze anomalies for new causal factors, hazards, failed mitigations. Update hazard analyses as appropriate. • Add or modify requirements for new mitigations as required.	• Anomaly report • HTS • Specifications • Software Safety Analysis and Verification Process
	10b. Analyze change proposals (e.g., Trouble Report resolutions or Engineering Change Proposals (ECPs) that implement resolutions to software issues) for new hazards.	Programs HTS. Programs SRHA and SRVM.	• Analyze proposed changes for new hazards, develop mitigations, and update hazard analyses. • Add or modify requirements for new mitigations as required.	• Change Proposals (defect resolution plan) • HTS • Specifications • Software Safety Analysis and Verification Process
11. Regression Testing. (Sub-process to SW Safety Testing and Verification)	11a. As required by SwCI, develop tests.	Programs HTS.	• Assess software modified through defect resolution for adequate safety mitigation and potential negative influences on safety functionality.	• SRVM • HTS • ECPs • Software Safety Analysis and Verification Process
12. Fleet Anomaly Reporting. (Sub-process to Fleet Release)		Programs HTS.	• Assess software defects reported by the fleet for safety. • If safety, identify risks and mitigations • Update HTS.	• Fleet Anomaly Reports • HTS • Software Safety Analysis and Verification Process