MIL-STD-882E Applies to All Your Software

Not Just the Code You Write

11 August 2016

Naval Ordnance Safety and Security Activity (NOSSA)
Indian Head, MD

Douglas J. Bower
douglas.j.bower@navy.mil
(301) 744-6069

Opening Video

Class Objectives

- Based on the requirements put forth in MIL-STD-882E
- Function Hazard Analysis
- Software Criticality/Level of Rigor
- Conducting and documenting the results of the Software Safety Analyses and Testing, including:
  - Requirements Analysis
  - Architecture Analysis
  - Design Analysis
  - Code Analysis
  - In-depth Safety Specific Testing
- Operating Systems and Other Non-Developmental Software
- Assessing the Remaining Safety Risk attributed to the system software

Using an example missile system developed by M&M Missile Company

Functional Hazard Analysis (FHA)

Purpose: The FHA is primarily used to identify and classify the system functions and the safety consequences of functional failure or malfunction, (i.e. hazards (MIL-STD-882E).

- These consequences will be classified in terms of severity for the purpose of identifying the safety-critical functions (SCFs), safety-critical item (SCIs), safety-related functions (SRFs), and safety-related items (SRIs) of the system.
- SCFs, SCIs, SRFs, and SRIs will be allocated or mapped to the system design architecture in terms of hardware, software, and human interfaces to the system.
- The FHA is also used to identify environmental and health related consequences of functional failure or malfunction.
- The initial FHA should be accomplished as early as possible in the Systems Engineering (SE) process to enable the engineer to quickly account for the physical and functional elements of the system for hazard analysis purposes; identify and document SCFs, SCIs, SRFs, and SRIs; allocate and partition SCFs and SRFs in the software design architecture; and identify requirements and constraints to the design team.

FHA Methodology

FHA considers the following to identify and evaluate functions within a system:

- Decomposition of the system and its related subsystems to the major component level.
- Functional description of each subsystem and component identified.
- Functional description of interfaces between subsystems and components. Interfaces should be assessed in terms of connectivity and functional inputs and outputs.
- Hazards associated with loss of function, degraded function or malfunction, or functioning out of time or out of sequence for the subsystems, components, and interfaces. The list of hazards should consider the next effect in a possible mishap sequence and the final mishap outcome.
- An assessment of the risk associated with each identified failure of a function, subsystem, or component.

FHA Methodology (continued)

- An assessment of whether the functions identified are to be implemented in the design hardware, software, or human control interfaces. This assessment should map the functions to their implementing hardware or software components.
- Functions allocated to software should be mapped to the lowest level of technical design or configuration item prior to coding (e.g., implementing modules or use cases).
- An assessment of Software Control Category (SCC) for each Safety-significant Software Function (SSSF). Assign a Software Criticality Index (SwCI) for each SSSF mapped to the software design architecture.
- A list of requirements and constraints (to be included in the specifications) that, when successfully implemented, will eliminate the hazard or reduce the risk. These requirements could be in the form of fault tolerance, detection, isolation, annunciation, or recovery. (i.e. Derived Requirements).

FHA Worksheet

Hazard ID #	Life-Cycle Phase	Activity	State/ Mode	Function	Functional Failure	Hazard Description	System Item(s)	Causal Factor Description	Mishap(s)	Effect(s)	Existing Mitigations	Software Control Category [2]	Rationale for SCC [2]	Initial MRI [3]/ SW Hazard Severity [1]	Software Criticality Index [2]	Target MRI [3]	Causal Factor Risk Level	Recommended Mitigations	Comments	Follow-On Actions
Identifier used to reference specific hazard	*The life-cycle phase for which the risk and risk assessment*	*The actions performed within a life-cycle phase*	*The State and/or Mode of the system for*	*The one of the system functions (implicit,*	*The detailed description for the specific failure mode of*	*The detailed description of the conditions under which hazardous*	*A functional or physical portion of a system designed, used or integrated to accomplish aspect of th system task mission*	*The detailed description of the failures, conditions, or events that*	*The event or series of events where hazardous energy release*	*The results of the mishap to include injury or death, damage to*	*Controls that are already planned or existing to mitigate the*	*The degree of autonomy, command and control authority, and*	*Provide the rationale for the SCC as it is not always evident*	*The first assessment of the potential risk of an identified*	*The level of analysis rigor required for risk assessment defined by the software control category and the mishap severity of the MRI*	*The projected risk the PM plans to achieve by implementing one or more of the designated recommended mitigations. This field should remain blank if no recommended mitigations are identified*	*The projected mishap risk level associated with the existence of the specific causal factor and its potential to realize the hazard and mishap*	*Controls that would reduce the Mishap risk potential. The goal should always be to eliminate the hazard if possible. When a hazard cannot be eliminated, the associated risk should be reduced to the lowest acceptable level by applying the system safety design order of precedence*	*Any important information and relevant information not captured elsewhere*	*Assigned or designated actions necessary to identify or better understand or characterize risk (e. g., perform FTA, perform software code analysis)*

Notes: 1 Assess severity for the system level mishap that could result from the failure of the function
2 Only applies to Software functions
3 Only applies to Hardware functions

FHA Worksheet

Hazard ID #	Life-Cycle Phase	Activity	State/ Mode	Function	Functional Failure	Hazard Description
Identifier used to reference specific hazard	*The life-cycle phase for which the risk and risk assessment apply*	*The actions performed within a life-cycle phase*	*The State and/or Mode of the system for the hazard of concern*	*The one of the system functions (implicit, implied or derived)*	*The detailed description for the specific failure mode of the function analyzed*	*The detailed description of the conditions under which hazardous energy may be released in an uncontrolled or inadvertent way*

FHA Worksheet

System Item(s)	Causal Factor Description	Mishap(s)	Effect(s)	Existing Mitigations	Software Control Category [2]	Rationale for SCC [2]	Initial MRI [3]/ SW Hazard Severity [1]
A functional or physical portion of a system designed, used or integrated to accomplish one aspect of the system task or mission	*The detailed description of the failures, conditions, or events that contribute either directly or indirectly to the existence of a hazard*	*The event or series of events where hazardous energy release could negatively effect equipment, personnel or environment; accident*	*The results of the mishap to include injury or death, damage to equipment and property, or damage to the environment*	*Controls that are already planned or existing to mitigate the risk*	*The degree of autonomy, command and control authority, and redundant fault tolerance of a software function in context with its system behavior*	*Provide the rationale for the SCC as it is not always evident*	*The first assessment of the potential risk of an identified hazard to establish a fixed baseline for the hazard. This may have come from the PHA*

Notes: 1 Assess severity for the system level mishap that could result from the failure of the function
2 Only applies to Software functions
3 Only applies to Hardware functions

FHA Worksheet

Software Criticality Index [2]	Target MRI [3]	Causal Factor Risk Level	Recommended Mitigations	Comments	Follow-On Actions
The level of analysis rigor required for risk assessment defined by the software control category and the mishap severity of the MRI	*The projected risk the PM plans to achieve by implementing one or more of the designated recommended mitigations. This field should remain blank if no recommended mitigations are identified*	*The projected mishap risk level associated with the existence of the specific causal factor and its potential to realize the hazard and mishap*	*Controls that would reduce the Mishap risk potential. The goal should always be to eliminate the hazard if possible. When a hazard cannot be eliminated, the associated risk should be reduced to the lowest acceptable level by applying the system safety design order of precedence*	*Any important information and relevant information not captured elsewhere*	*Assigned or designated actions necessary to identify or better understand or characterize risk (e. g., perform FTA, perform software code analysis)*

Notes: 1 Assess severity for the system level mishap that could result from the failure of the function
2 Only applies to Software functions
3 Only applies to Hardware functions

Example Missile System

- Application
  - Examples and practical exercises using Robin Hood Missile System (RHMS)

The RHMS is a software intensive system: all functions related to the pointing and firing of missiles are under the control of software. The RHMS consists of three major subsystems: the Bow Launcher, the Arrow missile, and the Archer Fire Control System (AFCS).

Exercise #1

- Identify five Top-Level Mishaps associated with the Robin Hood Missile System (RHMS)
  - Arrow Missile
  - Bow Launcher
  - Archer Fire Control System

Exercise #1
- Possible Answer

- Identify five Top-Level Mishaps associated with the RHMS (Missile & Launcher)
  1. Inadvertent/Early Ignition of Rocket Motor
  2. Inadvertent Warhead Detonation
  3. Loss of Flight Control
  4. Missile Engages Incorrect Target
  5. Launch Abort/Restrained Firing

  … There are may be others.

Software Safety Criticality

- Degree to which the software has influence on the safety related aspects of a system
  - Level of Control
    - Considers what other interlocks (both hardware and separate independent software) exist in the system
    - The ability of the software to assert the safety critical actions of the system
  - Mishap Severity
    - Determined in the same manner as hardware or system mishap severity
    - Level of personnel injury and/or equipment damage
- Drives the level of rigor of analysis and testing which needs to be applied

Level of Control

Level	Name	Description
1	Autonomous (AT)	Software functionality that exercises autonomous control authority over potentially safety-significant hardware systems, subsystems, or components without the possibility of predetermined safe detection and intervention by a control entity to preclude the occurrence of a mishap or hazard. (This definition includes complex system/software functionality with multiple subsystems, interacting parallel processors, multiple interfaces, and safety-critical functions that are time critical.)
2	Semi-Autonomous (SAT)	Software functionality that exercises control authority over potentially safety-significant hardware systems, subsystems, or components, allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the mishap or hazard. (This definition includes the control of moderately complex system/software functionality, no parallel processing, or few interfaces, but other safety systems/mechanisms can partially mitigate. System and software fault detection and annunciation notifies the control entity of the need for required safety actions.) Software item that displays safety-significant information requiring immediate operator entity to execute a predetermined action for mitigation or control over a mishap or hazard. Software exception, failure, fault, or delay will allow, or fail to prevent, mishap occurrence. (This definition assumes that the safety-critical display information may be time critical, but the time available does not exceed the time required for adequate control entity response and hazard control.)
3	Redundant Fault Tolerant (RFT)	Software functionality that issues commands over safety significant hardware systems, subsystems, or components requiring a control entity to complete the command function. The system detection and functional reaction includes redundant, independent fault tolerant mechanisms for each defined hazardous condition. (This definition assumes that there is adequate fault detection, annunciation, tolerance, and system recovery to prevent the hazard occurrence if software fails, malfunctions, or degrades. There are redundant sources of safety-significant information, and mitigating functionality can respond within any time-critical period.) Software that generates information of a safety-critical nature used to make critical decisions. The system includes several redundant, independent fault tolerant mechanisms for each hazardous condition, detection, and display.
4	Influential	Software generates information of a safety-related nature used to make decisions by the operator, but does not require operator action to avoid a mishap.
5	No Safety Impact (NSI)	Software functionality that does not possess command or control authority over safety-significant hardware systems, subsystems, or components and does not provide safety-significant information. Software does not provide safety-significant or time sensitive data or information that requires control entity interaction. Software does not transport or resolve communication of safety-significant or time sensitive data.

Mishap Severity

Description	Severity Category	Mishap Result Criteria
Catastrophic	1	Could result in one or more of the following: death, permanent total disability, irreversible significant environmental impact, or monetary loss equal to or exceeding $10M.
Critical	2	Could result in one or more of the following: permanent partial disability, injuries or occupational illness that may result in hospitalization of at least three personnel, reversible significant environmental impact, or monetary loss equal to or exceeding $1M but less than $10M.
Marginal	3	Could result in one or more of the following: injury or occupational illness resulting in one or more lost work day(s), reversible moderate environmental impact, or monetary loss equal to or exceeding $100K but less than $1M.
Negligible	4	Could result in one or more of the following: injury or occupational illness not resulting in a lost work day, minimal environmental impact, or monetary loss less than $100K.

MIL-STD-882E: Table I.

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
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.

- MIL-STD-882E describes the software safety in the context of “software contribution to system risk”
  - To avoid the misconception that software analyses are performed without system context
  - To ensure all software safety issues have clearly defined system mishap context

What is the process for defining software contribution to system risk?

- To define software contribution to system risk:
  1. Flow the system level SSFs to the software function
     - Determines which software functions contribute to the SSF
     - Performed during the FHA (typically) or PHA
  2. Evaluate each software-safety function for mishap severity potential (i.e., Catastrophic, Critical, Marginal, or Negligible)
     - Utilizing the hazards from previous analyses (e.g., PHA, SSHA)
  3. Evaluate each software safety function for level of autonomy (i.e., SCC 1-5)
  4. Derive the SwCI and associated LOR tasks using MIL-STD-882E Table V
  5. Execute the LOR:
     - Safety in software design, development, and verification processes
     - Software safety analytical and verification tasks
  6. All identified risk is defined and associated to hazard and system level mishaps

Exercise #2

- Perform a preliminary Functional Hazard Analysis (FHA) of the Arrow Missile in the context of the Robin Hood Missile System (RHMS)
- Based on the information provided in the handouts

FHA Worksheet

Hazard ID #	Life-Cycle Phase	Activity	State/ Mode	Function	Functional Failure	Hazard Description
1	Tactical Operation	Missile engage-ment	Post launch	Boost Phase Autopilot	Operates out of sequence	Loss of flight control before safe separation from launch platform
2	Tactical Operation	Missile engage-ment	Mid course	Mid course guidance	Fails to operate	Loss of flight control after safe separation
3	Tactical Operation	Missile engage-ment	Terminal	Navigation	Operates at wrong time	Navigation error results in erroneous flight path

FHA Worksheet

System Item(s)	Causal Factor Description	Mishap(s)	Effect(s)	Existing Mitigations	Software Control Category [2]	Rationale for SCC [2]	Initial MRI [3]/ SW Hazard Severity [1]
Autopilot	Incorrect command generated by the autopilot	Loss of flight control prior to safe separation	Death or severe injury to personnel	None - Software is operating autonomously	1	Software has complete control of missile flight	I
Fletching guidance and control unit (FGCU)	Incorrect guidance commend given to autopilot	Loss of flight control after safe separation	Loss of missile	Self destruct capability	2	Errant missile will self destruct	III
Navigation System	Incorrect missile location sent to FGCU	Missile engages wrong target	Death or severe injury to personnel	Navigation system has GPS and inertial guidance	2	All contained within same software component	I

Notes: 1 Assess severity for the system level mishap that could result from the failure of the function
2 Only applies to Software functions
3 Only applies to Hardware functions

FHA Worksheet

Software Criticality Index [2]	Target MRI [3]	Causal Factor Risk Level	Recommended Mitigations	Comments	Follow-On Actions
1	N/A				
3	N/A				
1	N/A				

Notes: 1 Assess severity for the system level mishap that could result from the failure of the function
2 Only applies to Software functions
3 Only applies to Hardware functions

Exercise #3

- For these three functions determine the specific types of analysis and testing to be performed

Software Analysis Worksheet

Software Function	Associated Hazard	Software Control Category	Hazard Severity	SwCI	Level of Rigor Tasks Required
Boost Phase Autopilot	N/A	1	I	1	
Mid course guidance	N/A	2	III	3	
Navigation	N/A	2	I	1	

Exercise #3

- For these three functions determine the specific types of analysis and testing to be performed

Software Analysis Worksheet

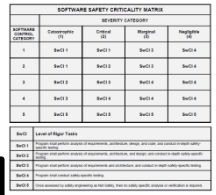
Software Function	Associated Hazard	Software Control Category	Hazard Severity	SwCI	Level of Rigor Tasks Required
Boost Phase Autopilot	N/A	1	I	1	Requirements, architecture, design, and code analysis; and conduct in-depth safety-specific testing
Mid course guidance	N/A	2	III	3	Requirements and architecture analysis; and conduct in-depth safety-specific testing
Navigation	N/A	2	I	1	Requirements, architecture, design, and code analysis; and conduct in-depth safety-specific testing

Exercise #3 - Discussion

- For these three functions determine the specific types of analysis and testing to be performed
  - After completing actions in Ex #3, how would the results of the application of the LoR be documented

Software Safety Analysis and Verification Process

Software Criticality Matrix



Top-Level Process

Start

System Definition and Software Safety Planning

Determine Software Criticality Index (SwCI)

Software Requirements Hazard Analysis (SwCI 1-3)

Software Architectural Hazard Analysis (SwCI 1-3)

Software Design Hazard Analysis (SwCI 1-2)

Code Level Hazard Analysis (SwCI 1)

Operator Documentation Safety Review

Software Testing and Verification (SwCI 1-4)

- Safety - Specific Testing (SwCI 4)
- In-depth Safety - Specific Testing (SwCI 1-3)

Formal Review

Fleet Release

Sub-Process

Defect Resolution

Regression Testing

Fleet Anomaly Reporting

SwCI (1,2, and 3) LOR Task: SW Requirements Hazard Analysis

- Safety Requirements Hazard Analysis (SRHA) is performed on SW as part of the Low-Level SRHA to ensure that there are adequate safety requirements associated with safety-significant SW functionality
  - SW Safety Requirements Trace from System-Level > Sub-System Level > Software Requirements. New safety requirements are derived per SRHA process
- Three Categories of SW Safety Requirements are:
  - Initiating Software Safety Requirements (ISSRs)
    - SW Requirements related to the SSFs that may initiate hazards if not defined and implemented appropriately
  - Generic Software Safety Requirements (GSSRs)
    - Are designed features, constraints, development processes and coding standards that are generally used with SW
  - Mitigating Software Safety Requirements (MSSRs)
    - These requirements mitigate or control mishap or hazard causes to acceptable levels of safety risk with regards to the system's SW
- Safety Requirements Verification Matrix (SRVM)
  - SRVM documents SW Safety Requirements analysis/test results

SwCI (1,2, and 3) LOR Task: Architectural Hazard Analysis

- Architectural Analysis
  - SW Architecture - The organizational structure of a system or Computer Software Configuration Item (CSCI), identifying its components, their interfaces, and concept of execution among them [Reference Allied Ordnance Publication (AOP)-52]
  - Conducting computing system and software architectural hazard analysis:
    - Identify/Define the allocation of System Functions to Architecture
    - Identify/Define the Software and Interface Architecture Requirements
    - Reviewing architecture against software safety-significant requirements (SSRs) and Hazard Tracking Record (HTR) software mitigations to determine which cannot be supported by the current architectures
    - Identify new architecture hazards and define supported mitigations

SW Architectural Diagrams

- Various architectural views per International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE) 42010 (IEEE 1471):
  - Functional/logical viewpoint
  - Code/module viewpoint
  - Development/structural viewpoint
  - Concurrency/process/runtime/thread viewpoint
  - Physical/deployment/installation viewpoint
  - User action/feedback viewpoint
  - Data view/data model

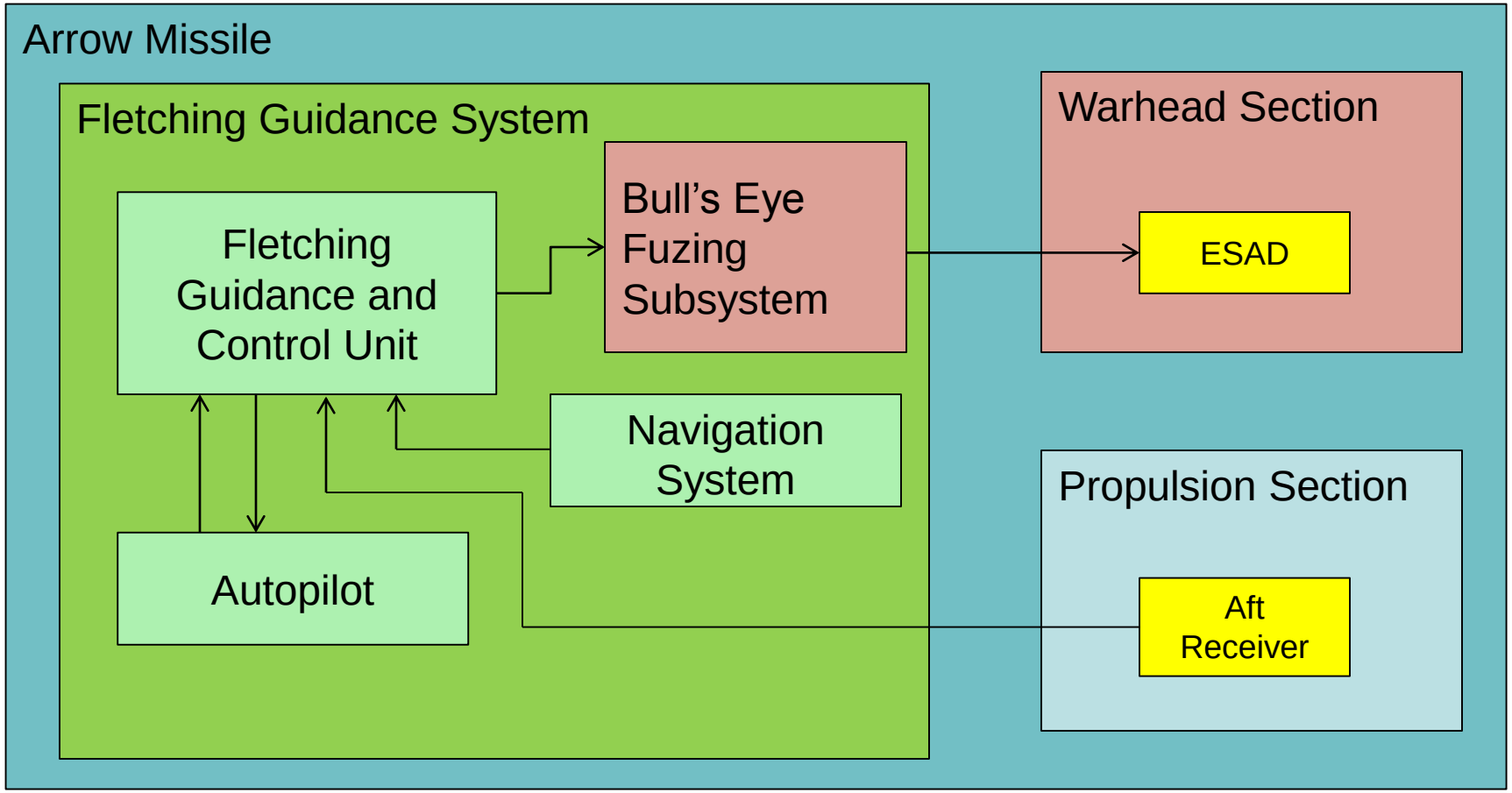
Architectural Analysis is conducted on available documents (requirements and views) and generic requirements

Exercise #4

- For RHMS, develop a top-level architecture diagram for the Arrow Missile, based on the information provided in the handout

Exercise #4

- For RHMS, develop a top-level architecture diagram for Arrow Missile, based on the information provided in the handout



SwCI (1 and 2) LOR Task:
SW Design Hazard Analysis

- SW Design Analysis
  - SW Design - The characteristics of a system or CSCI that are selected by the developer in response to the requirements. Some will match the requirements; others will be elaborations of requirements, such as definitions of all error messages; others will be implementation related, such as decisions, about what software units and logic to use to satisfy the requirements. [Reference AOP-52]
- Conducting SW Design Hazard Analysis:
  - Identify/Define allocation of System Functions to SW Design
  - Identify the correlating SW Interface Design Requirements
  - Review design against software SSRs and HTR software mitigations to determine which cannot be supported by the current design
  - Identify new design hazards and define supported mitigations

Relationship between Architecture and Design Analysis

- Software architecture defines the design constraints so it will be available for use in detailed design
- Analysis of architecture can detect hazards early, when they can be economically mitigated
- Software architecture drives software design, but actual design may exceed architectural intent or fall short
  - Shortfalls with safety impact require risk assessment and (likely) new mitigations
- Safety analysis approach [architecture and design] similar but at different levels of abstraction
  - SSRs are evaluated on 2-pass approach, architectural then design

Joint Software System Safety Engineering Handbook (JSSSEH) Generic Requirements

- E.8.5 **Data Transfer Messages**
- Data transfer messages shall be of a predetermined format and content. Each transfer shall contain a word or character string indicating the message length (if variable), the type of data, and the content of the message. At a minimum, parity checks and checksums shall be used for verification of correct data transfer. CRCs shall be used where practical. *No information from data transfer messages shall be used prior to verification of correct data transfer*.

Analyze **architecture** to ensure it supports verification of safety data

- E.3.13 **Positive Feedback Mechanisms**
- Software control of critical functions *shall have feedback mechanisms* that give positive indications of the function's occurrence.

Analyze **architecture** to ensure it supports positive feedback for safety functions

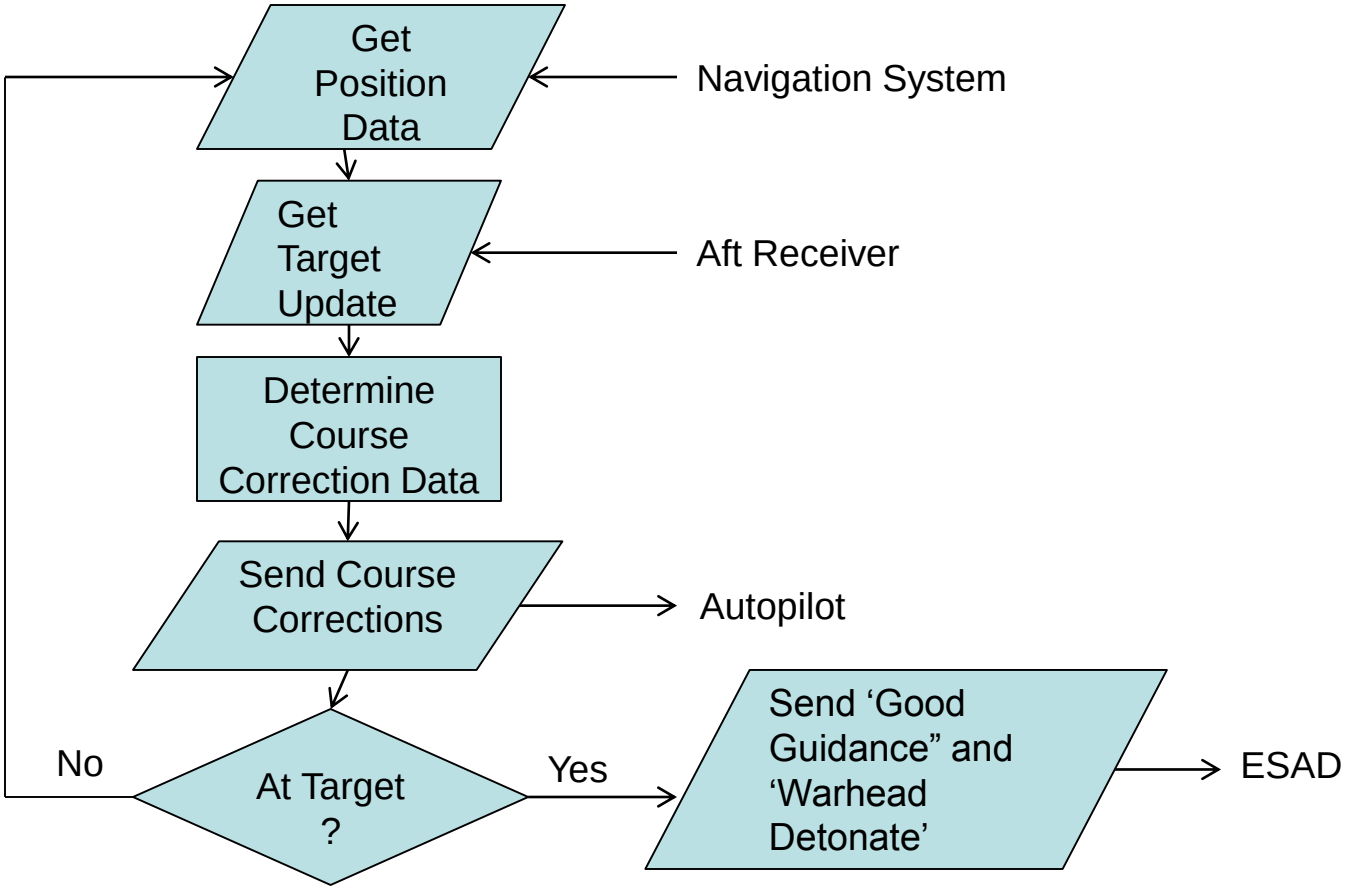
- Use each generic requirement to assess the architectural intent
  - If generic requirements are not supported in architecture, may represent risk
  - If generic requirements are supported in the architecture, record as planned mitigations for hazards

Exercise #5

- For RHMS, complete a design analysis for the **Fletching Guidance and Control Unit**, based on the information provided in the handout

Exercise #5

- For RHMS, complete a design analysis for the **Fletching Guidance and Control Unit**, based on the information provided in the handout



Outline

Perform Code Analysis (Coding experience not necessary, but helpful)

- Data Structure Analysis
- ***Data Flow Analysis***
- ***AOP-52 Compliance***
- ***Software Safety Analysis and Verification Process Flow***

Data Flow Analysis

- The purpose of data flow analysis is to identify errors in the use of data that is accessed by multiple routines
- The following are some examples of errors that can be found via data flow analysis:
  - Data which is utilized by a system prior to being initialized
  - Unused data items
  - Unintended data item modification
  - Failure to accurately update or modify data items

AOP-52 Compliance Assessment

- The purpose of conducting a compliance assessment is to ensure the code follows a set of coding standards. Non-compliance could result in errors that could lead to potential safety impact.
- Compliance requirements can come from a multitude of sources, with AOP-52 and JSSSEH being two of them.
- Three generic requirements from AOP-52 will be the discussed in this last exercise.

AOP-52 Compliance Assessment
Definition of Terms

Term	Definition
Flags and Variables	Flags and variable names shall be unique. Flags and variables shall have a single purpose and shall be defined and initialized prior to use.
Execution Path	Safety Critical Computing System Functions (SCCSFs) shall have one and only one possible path leading to their execution.
Conditional Statements	Conditional statements shall have all possible conditions satisfied and be under full software control (i.e., there shall be no potential unresolved input to the conditional statement). Conditional statements shall be analyzed to ensure that the conditions are reasonable for the task and that all potential conditions are satisfied and not left to a default condition. All condition statements shall be annotated with their purpose and expected outcome for given conditions.

SwCI (1-4) LOR Task:
SW Testing and Verification

Software Safety Testing

- Testing should address not only performance-related SW Requirements, but the SW Safety Significant Requirements as well
- The minimum level of software safety testing depends upon the LOR performed on the associated SW SSF:
  - Two types of Software Testing - MIL-STD-882E Table V defines the LOR where In-Depth Safety-Specific Testing (LOR SwCI 1-3) and Safety-Specific Testing (LOR SwCI 4) are required
  - In all cases:
    - Safety input required to the test plan to ensure test and verification of safety significant software (i.e., Participate in or Witness Testing)
    - Safety Testing should be conducted at Unit, CSCI, (Sub-)System Level, and Test Coverage Analysis
- Leads to Verification and Validation of SW SSR implementation
  - Results recorded in SRVM for LOR SwCI 1-3 only
- Test Plans and Test Reports provide documentation for safety engineer to cite during final risk assessment

Examples of SW Safety Specific Testing (SwCI 4)

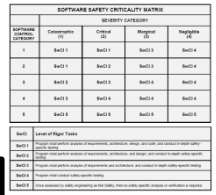
- **Endurance Testing** - Demonstrate the ability of the system to run for a defined period of time without failing (Defined in JSSSEH 4.4.2.6.)
- **User Interface Tests** - Verify the functionality of the user interface (Defined in JSSSEH 4.4.2.7.)
- **Fault Insertion and Failure Testing** - Provide assurance that the software will safely respond to various faults or failures in the hardware and software (Defined in JSSSEH 4.4.2.8.)
- **Safety injected** into: Functional Testing, Physical Testing
- **Go/No-Go Path Testing** - Verify required functionality works in the go-path scenario and with failures incurred
- **Human Integration Testing** - Ensure the operator can safety manage the equipment and workload
- **Regression Testing** - Assures modifications to the SW do not adversely affect the functionality

Examples of SW In-Depth Safety - Specific Testing (SwCI 1-3)

- In addition to Safety Specific Testing, In-Depth Safety testing may include:
  - Path Coverage Testing - Ensure every possible path in the code is executed at least once (Defined in JSSSEH 4.4.2.3.)
  - Requirements-Based Testing - Verify software implements the high-level requirements (Defined in JSSSEH 4.4.1.2. and 4.4.2.1.)
  - Statement Coverage Testing - Verify the “success” path of an IF statement in code is exercised (Defined in JSSSEH 4.4.2.4.)
  - Mutation Testing - Modify code to achieve a specific testing objective
  - Perturbation Testing - Variation of mutation testing in which test team “perturbs” the execution environment to determine the reaction of software
  - Safety injected into Exception Handling, Boundary Handling, and Data Rates Testing
  - Stress Testing - Verify the ability of the system to function under high stress conditions
  - Stability/Endurance Testing - Demonstrate the ability of the system to run for a defined period of time without failing (Defined in JSSSEH 4.4.2.6.)

Software Safety Analysis and Verification Process

Software Criticality Matrix



Top-Level Process

Start

System Definition and Software Safety Planning

Determine Software Criticality Index (SwCI)

Software Requirements Hazard Analysis (SwCI 1-3)

Software Architectural Hazard Analysis (SwCI 1-3)

Software Design Hazard Analysis (SwCI 1-2)

Code Level Hazard Analysis (SwCI 1)

Operator Documentation Safety Review

Software Testing and Verification (SwCI 1-4)

- Safety - Specific Testing (SwCI 4)
- In-depth Safety - Specific Testing (SwCI 1-3)

Formal Review

Fleet Release

Sub-Process

Defect Resolution

Regression Testing

Fleet Anomaly Reporting

SW Safety Formal Review

- Formal Review - is done to provide documented evidence that the software contribution to system risk is defined and all remaining risks are accepted
  - This review is done after the LOR tasks have been completed and the System risk is updated with SW contribution and documented within the context of a Safety Assessment Report (SAR) and Mishap Assessment Report (MAR)
  - OQE that substantiates the completion of all LOR tasks must be provided in the Technical Data Package. NOTE: This is not merely a checklist of task completion, but is the actual analytical products.
  - The system risk, including SW's contribution, is presented within a Technical Data Package for review by the appropriate Safety Authority

Operating Systems and Other Non-Developmental Software

- Operating System and Development Environment Considerations
- WIN 10 Specific Considerations

Operating System and Development Environment Considerations

- COTS Operating Environment
- COTS Hardware Impacts
- Programming Language
- Development Paradigm
- COTS Software Hazards

COTS Operating Environment

- COTS Interoperability Hazard Analysis
  - Ensure the safety requirements for the COTS Operating Environment (OE) software are addressed in the System Requirements
  - Ensure proactive involvement of Safety in COTS OE software selection

COTS Operating Environment

- Potential COTS OE impacts to safety
  - Message delivery – How will safety critical message delivery be guaranteed in the COTS OE?
  - Initialization / Failover / Faildown (casualty configurations) – How will safety critical state information be maintained throughout the system through initialization, failover, and faildown conditions?

COTS Operating Environment

- Key steps for COTS Interoperability Hazard Analysis:
  - Analyzing functional & behavioral characteristics of COTS
  - Analyzing dead, unused, or inactivated options in COTS
  - Linking hazards/causal factors to COTS requirements (and the reverse)
  - Developing mitigation requirements for COTS
  - Establishing COTS level of rigor

Emphasis should be placed on identifying mitigations related to safety critical message delivery and in software initialization, failover, and faildown during safety critical system events.

COTS Operating Environment

- Key steps for Design & Implementation Hazard Analysis:
  - Matching existing hazards to COTS
  - Analyzing COTS for introduction of new hazards
  - Analyzing COTS in the system interfaces
  - Developing new or modified requirements and software safety test cases
  - Creating and coordinating mitigations and test cases
  - Documenting special COTS implementation safety requirements and COTS SCI

Emphasis should be placed on ensuring adequate mitigations to ensure safety critical message delivery and safe software initialization, failover, and faildown.

COTS Operating Environment

- Key steps for Software Test & Validation:
  - Supporting software and integration testing
  - Providing safety support for IV&V
  - Verifying new or modified safety requirements and safety related functions
  - Providing evidence to support safety analysis and verification of the COTS operating environment changes

Emphasis should be placed on testing mitigations that ensure safety critical message delivery and in software initialization, failover, and faildown during safety critical system events.

COTS Operating Environment

- Key steps for Regression Testing:
  - Analyzing safety functionality and safety data and structure to be maintained through any change
  - Developing new or modified tests to verify and validate safety functionality and safety data and structure

Emphasis should be placed on testing mitigations that ensure safety critical message delivery and in software initialization, failover, and faildown during safety critical system events.

COTS Operating Environment

- Key steps for Analysis and Verification:
  - Supporting customer integration and certification testing
  - Providing safety evidence for safety review authority to support request for operational use

Emphasis should be placed on documenting validated mitigations that ensure safety critical message delivery and in software initialization, failover, and faildown during safety critical system events.

COTS Hardware Impacts

- Return to Design & Implementation Hazard Analysis
  - Ensure the safety requirements for any COTS hardware are addressed in the System Requirements
  - Ensure proactive involvement of Safety in COTS hardware selection

COTS Hardware Impacts

- Key potential COTS Hardware impacts to safety:
  - Data marshalling of legacy messages (big endian/little endian word and bit conversion of legacy data fields)
    - Hand-crafted, message-specific data marshalling might be needed
    - Potential for scrambling data fields in legacy messages
    - Potential for introduction of unacceptable latencies to message processing between big and little endian processors
  - Equipment management (detecting and responding to equipment failures)
    - What are potential safety impacts of equipment failures?
    - How will equipment failures be handled? (automatically? operator alert?)
    - How will mitigations be tested?

COTS Hardware Impacts

- Key potential COTS Hardware impacts to safety (cont'd):
  - Replacement of legacy interfaces with COTS
    - Potential loss of legacy protocols/ message validation
    - Potential for introduction of unacceptable latencies with introduction of NICs, etc. (more "hops" in the communications)
    - Potential negative latency impacts from loss of hard-wired, private, point-to-point communications

COTS Hardware Impacts

- Key steps for Design & Implementation Hazard Analysis:
  - Analyzing COTS for introduction of new hazards
  - Developing new or modified requirements and software safety test cases
  - Creating and coordinating mitigations and test cases
  - Documenting special COTS implementation safety requirements and COTS SCI

Emphasis should be placed on identifying software mitigations for any potential hazards introduced by data marshalling (i.e., scrambled data fields in legacy messages, message latencies), if applicable, or COTS equipment failures (e.g., failures induced by temperature, shock, or humidity).

COTS Hardware Impacts

- Key steps for Software Test & Validation:
  - Supporting software and integration testing
  - Providing safety support for IV&V
  - Analyzing test results related to data marshalling or equipment failure response
  - Verifying new or modified safety requirements and safety related functions
  - Providing evidence to support safety analysis and verification of the COTS operating environment changes

Emphasis should be placed on testing data marshalling (i.e., scrambled data fields in legacy messages, unacceptable latencies), if applicable, and COTS equipment failures (e.g., failures induced by temperature, shock, or humidity).

COTS Hardware Impacts

- Key steps for Regression Testing:
  - Analyzing safety functionality and safety data and structure to be maintained through any change
  - Developing new or modified tests to verify and validate safety functionality and safety data and structure

Emphasis should be placed on testing data marshalling (i.e., scrambled data fields in legacy messages, unacceptable latencies), if applicable, and COTS equipment failures (e.g., failures induced by temperature, shock, or humidity).

COTS Hardware Impacts

- Key steps for analysis and verification:
  - Supporting customer integration and certification testing
  - Providing safety evidence for safety review authority to support request for operational use

Emphasis should be placed on documenting validated mitigations that ensure safe response for any potential hazards introduced by data marshalling or COTS equipment failures.

Programming Language

- Design & Implementation Hazard Analysis
  - Ensure proactive involvement of Safety in programming language selection
    - A primary C++ issue: thread safety
    - A primary Java issue: guaranteeing latencies

Programming Language

- Key potential Programming Language impacts to safety
  - For Java:
    - How will safety critical timing requirements be guaranteed?
  - For multi-threaded C++ and Java:
    - How will thread safety be analyzed?
      - Has potential for deadlock of threads been eliminated (two or more threads mutually blocking each other "forever" over two or more shared resources)?
      - Are ***all*** safety critical shared data/resources protected through a proper mutual exclusion mechanism (e.g., spin locks, semaphores, monitors, critical sections)?

Programming Language

- Key steps for Design & Implementation Hazard Analysis:
  - Analysis for introduction of new hazards
  - Developing new or modified requirements and software safety test cases
  - Creating and coordinating mitigations and test cases

Emphasis should be placed on identifying software mitigations for any potential hazards introduced by Java or C++ (e.g., timing impacts from Java, corruption of shared data or deadlock from incorrect thread synchronization).

Programming Language

- Key steps for Software Test & Validation:
  - Supporting software and integration testing
  - Providing safety support for IV&V
  - Analyzing test results related to safety critical timing and safety critical processing under extreme load and stress
  - Verifying new or modified safety requirements and safety related functions

Emphasis should be placed on testing software for any potential hazards introduced by Java or C++ (e.g., timing impacts from Java, corruption of shared data or deadlock from incorrect thread synchronization).

Programming Language

- Key steps for Regression Testing:
  - Analyzing safety functionality and safety data and structure to be maintained through any change
  - Developing new or modified tests to verify and validate safety functionality and safety data and structure under extreme load and stress

Emphasis should be placed on testing software for any potential hazards introduced by Java or C++ (e.g., timing impacts from Java, corruption of shared data or deadlock from incorrect thread synchronization).

Programming Language

- Key steps for Analysis and Verification:
  - Supporting customer integration and certification testing
  - Providing safety evidence for safety review authority to support request for operational use

Emphasis should be placed on documenting validated safety critical timing and functionality under load and stress that ensure absence of impacts from introduction of Java or C++.

Development Paradigm

- Design & Implementation Hazard Analysis
  - Functional granularity of applications (more processes, more inter-process communication)
  - Move from function-oriented to object-oriented (how functionality is mapped to objects, and vice versa)
  - Move toward data-centric system (DDS PubSub, SOA)

Development Paradigm

- Key potential Development Paradigm impacts to safety
  - Moving to object-oriented design:
    - How will legacy functional requirements be mapped to new object-oriented design? (Will internal application complexity increase?)
  - Moving to more processes:
    - How will safety-related system-level timing be guaranteed?
    - How will safety-related system-level (distributed) control be maintained?

Development Paradigm

- Key steps for Design & Implementation Hazard Analysis:
  - Analysis for introduction of new hazards
  - Developing new or modified requirements and software safety test cases
  - Creating and coordinating mitigations and test cases

Emphasis should be placed on identifying software mitigations for any potential hazards introduced by new paradigm (e.g., end-to-end timing, distribution of system control).

Development Paradigm

- Key steps for Software Test & Validation:
  - Supporting software and integration testing
  - Providing safety support for IV&V
  - Analyzing test results related to safety critical timing and safety critical processing under extreme load and stress
  - Verifying new or modified safety requirements and safety related functions

Emphasis should be placed on testing software for any potential hazards introduced by the development paradigm (e.g., end-to-end timing, distributed functionality, loss system control).

Development Paradigm

- Key steps for Regression Testing:
  - Analyzing safety functionality and safety data and structure to be maintained through any change
  - Developing new or modified tests to verify and validate safety functionality and safety data and structure under extreme load and stress

Emphasis should be placed on testing software for any potential hazards introduced by new paradigm (e.g., end-to-end timing, system control).

Development Paradigm

- Key steps for Analysis and Verification:
  - Supporting customer integration and certification testing
  - Providing safety evidence for safety review authority to support request for operational use

Emphasis should be placed on documenting validated safety critical timing and functionality under load and stress that ensure absence of impacts from introduction of new paradigm.

COTS Software Hazards

- Document plausible COTS software service failures and their potential safety impact
  - Consider *transient* and *persistent* service failures
- Document mitigations needed for those service failures with safety impact
- Map the mitigations to system requirements or design
- Ensure the mitigations will be adequately tested

COTS Software Hazards

- Use a FMECA-style approach to identify potential hazards or causal factors
  - Identify new failure modes and their potential safety impacts (e.g., failures of omission, commission, incorrect result, late service, early service)
  - Identify mitigations for failures with safety impacts
  - Match mitigations with requirements and design
  - Identify testing for the mitigations (e.g., fault injection)
- Identify lost legacy mitigations
  - Identify replacement mitigations for legacy (Non-COTS) failures with safety impacts
  - Match mitigations with requirements and design
  - Identify testing for the mitigations (e.g., fault injection)

COTS Software Hazards
New Mitigations

- COTS software often brings new (perhaps better) means of mitigating non-COTS hazards
- Document COTS software mitigations for non-COTS hazards
- Ensure the mitigations will be adequately tested

Windows® Specific Considerations

- Risk Reduction Hierarchy
- Safety Considerations When Assessing the Use of Windows®

Where would you considerer using Windows® operating system within the Robin Hood Missile System?

Risk Reduction Hierarchy

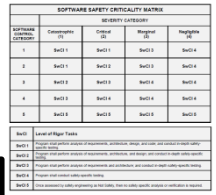
- The goal of this policy document is to provide means by which the software developer/maintainer can minimize the safety risk associated with use of Windows® . To that end one should try to eliminate or at least mitigate the safety risk of the using Windows® by utilizing the following risk reduction hierarchy and safety risk assessment process.
- 1. Do not use Windows® in any manner which could affect safety significant processing (i.e. select a different OS for safety significant processors).
- 2. Identify and employ techniques to isolate the Windows® OS from the safety critical processing and prevent Windows® anomalies from leading to or contributing to any type of mishap.
- 3. Analyze the behavior of Windows® and its influence on the safety significant functions and validate analysis results through tests.
- 4. Implement procedural mitigations to reduce the impact of Windows® anomalies on safety signification functions.
- For any of the above, document the safety risk. Any lack of detailed design information associated with Windows® is likely to translate to additional safety risk depending on the mitigation method selected from the above hierarchy.

Safety Considerations When Assessing the Use of Windows®

- Design Considerations
  - What steps were taken to eliminate or minimize the influence of the OS on the safety critical processing?
  - Describe the design techniques employed to mitigate the safety impact of the OS and potential OS failures.
- Architecture Analysis
  - How does the system architecture contribute to the safety impact of using Windows® ?
  - Does the system design include wrappers or other forms of protection between the safety significant application and the OS?
- Objectives of Safety Analysis and Testing
  - Determine the potential for OS instability causing safety significant functions to operate anomalously.
  - Determine the effect of memory management failures, including those related to data corruption and execution paths excursions.
  - Determine the potential for OS failure (system crash) to inhibit the operation of safety significant hazard detecting and mitigating functions.
- Further Considerations
  - Windows® is a general purpose, performance based operating systems, that is not designed with safety significant use in mind. Many performance based decisions, such as lack of strict memory management, introduce significant risk when carrying out safety significant functions.
  - Windows® is prone to security vulnerabilities, which could also translate to safety vulnerabilities. Additionally, the need for security based upgrades to the OS could lead to functional uncertainties, which could become safety

Assessing the Remaining Safety Risk attributed to the system software

Software Criticality Matrix



Top-Level Process

Start

System Definition and Software Safety Planning

Determine Software Criticality Index (SwCI)

Software Requirements Hazard Analysis (SwCI 1-3)

Software Architectural Hazard Analysis (SwCI 1-3)

Software Design Hazard Analysis (SwCI 1-2)

Code Level Hazard Analysis (SwCI 1)

Operator Documentation Safety Review

Software Testing and Verification (SwCI 1-4)

- Safety - Specific Testing (SwCI 4)
- In-depth Safety - Specific Testing (SwCI 1-3)

Formal Review

Fleet Release

Sub-Process

Defect Resolution

Regression Testing

Fleet Anomaly Reporting

Incomplete LoR Risk Assessment

RELATIONSHIP BETWEEN SwCI, RISK LEVEL, LOR Tasks, AND RISK		
Software Criticality Index (SwCI)	**Risk Level**	**Software LOR Tasks and Risk Assessment/Acceptance**
SwCI 1	**High**	• If SwCI 1 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as HIGH and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SwCI 1 LOR tasks or prepare a formal risk assessment for acceptance of a HIGH risk.
SwCI 2	**Serious**	• If SwCI 2 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as SERIOUS and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SwCI 2 LOR tasks or prepare a formal risk assessment for acceptance of a SERIOUS risk.
SwCI 3	**Medium**	• If SwCI 3 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as MEDIUM and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SwCI 3 LOR tasks or prepare a formal risk assessment for acceptance of a MEDIUM risk.
SwCI 4	**Low**	• If SwCI 4 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as LOW and provided to the PM for decision. The PM shall document the decision of whether to expend the resources required to implement SwCI 4 LOR tasks or prepare a formal risk assessment for acceptance of a LOW risk.
SwCI 5	**Not Safety**	• No safety-specific analyses or testing is required.

Software Hazard Causal Factor Risk Assessment Criteria

Risk Levels	Description of Risk Criteria
	A software implementation or software design defect that upon occurring during normal or credible off-nominal operations or tests:
High	• Can lead directly to a catastrophic or critical mishap, or • Places the system in a condition where no independent functioning interlocks preclude the potential occurrence of a catastrophic or critical mishap.
Serious	• Can lead directly to a marginal or negligible mishap, or • Places the system in a condition where only one independent functioning interlock or human action remains to preclude the potential occurrence of a catastrophic or critical hazard.
Medium	• Influences a marginal or negligible mishap, reducing the system to a single point of failure, or • Places the system in a condition where two independent functioning interlocks or human actions remain to preclude the potential occurrence of a catastrophic or critical hazard.
Low	• Influences a catastrophic or critical mishap, but where three independent functioning interlocks or human actions remain, or • Would be a causal factor for a marginal or negligible mishap, but two independent functioning interlocks or human actions remain. • A software degradation of a safety critical function that is not categorized as high, serious, or medium safety risk. • A requirement that, if implemented, would negatively impact safety; however code is implemented safely.