16987
Software Safety Analysis – A New Requirement?

Robert E. Smith, CSP, Booz Allen Hamilton
NDIA Systems Engineering Conference
Springfield, VA
October 30, 2014

Agenda

- Purpose
- Why is Software Safety Analysis Important?
- Software Safety Assessment Standards and Policy
- Key Definitions
- DoDI 5000.02 Documentation Requirement – Scenarios
- General Software Safety Steps
- Software Safety Analysis and Verification Process
- Conclusion

Purpose

- Describe why software safety analysis is important
- Identify DoDI 5000.02 requirements for documentation of the analysis process used and the results
- Describe two scenarios where a software safety analysis is required
  - Programs where there is a hardware and software component
  - Programs where there is only a software component
- Describe the basic structure of the software safety analysis process and the MIL-STD-882E specified Level of Rigor (LOR) tasks

Why is Software Safety Analysis Important?

- Software can cause, influence, contribute to, or mitigate hazards that may lead to mishaps
- The system safety and software system safety analysis processes identify and mitigate the software contributors to hazards and mishaps
  - Successful execution of pre-defined LOR tasks increases the confidence that the software will perform as specified to software performance requirements, while reducing the number of contributors to hazards that may exist in the system
  - Both processes are essential in reducing the likelihood of software initiating a propagation pathway to a hazardous condition or mishap

Software Safety Assessment Standards and Policy

- The need and requirement for the software safety assessment is not new -- it has been included in government and industry safety and software safety standards, but the requirement has become more explicit over time
  - MIL-STD-882B, System Safety Program Requirements, MAR 1984; MIL-STD-882C, System Safety Program Requirements, JAN 1993; and MIL-STD-882E, Standard Practice for System Safety, APR 2012: Para 4.4 **Software contribution to system risk**
  - DoDI 5000.02 since 2001; Interim Release NOV 2013, Enclosure (3), Systems Engineering:
    - Para 11, Software. "The SEP should address the following: software unique risks; inclusion of software in technical reviews; identification, tracking, and reporting of metrics for software technical performance, process, progress, and quality; **software safety** and security considerations; and software development resources."
    - Para 16, ESOH. "The Program Manager **will use the methodology in MIL-STD-882E**, 'DoD Standard Practice for System Safety.'"
  - DoD Joint Software Systems Safety Engineering Handbook (JSSSEH) Version 1.0, AUG 2010
  - Allied Ordnance Publication (AOP)-52 (EDITION 1) – Guidance On Software Safety Design and Assessment of Munitions-Related Computing Systems, DEC 2008
  - NASA-STD 8719.13 Software Safety Standard, MAY 2013

Key Definitions

- Safety Significant
  - A term applied to a condition, event, operation, process, or item that is identified as either safety critical or safety related
- Safety Critical
  - A term applied to a condition, event, operation, process, or item whose mishap severity consequence is either Catastrophic or Critical (e.g., safety-critical function, safety-critical path, and safety-critical component)
- Safety Related
  - A term applied to a condition, event, operation, process, or item whose mishap severity consequence is either Marginal or Negligible

DoDI 5000.02 Documentation Requirement – Scenario One

1. The Program has responsibility for development, integration or upgrade for hardware controlled by software (e.g., the aircraft that relies upon an operational flight program)
   - The system safety and software system safety analysis processes identify and mitigate the software contributors to system hazards and mishaps
   - Document the planning for software safety analysis, as part of overall Environment, Safety, and Occupational Health (ESOH) planning, in the Systems Engineering Plan (SEP)
   - Document the software safety analysis and risk assessment results in the Programmatic Environment, Safety, and Occupational Health Evaluation (PESHE)

DoDI 5000.02 Documentation Requirement – Scenario Two

2. The Program has responsibility for development of the software package only and has no responsibility for how the software will be applied (e.g., a software program that allows the collection and distribution of medical information, personnel information (including personally identifiable information))

- In this case, the software safety analysis has to take into account how the software package will be used in order to determine if software could contribute to the risk of a mishap occurring
- Document the planning for software system safety analyses in the SEP
- If the software package can contribute to mishap risk, the analyses follow the same process as Scenario One - Document the software safety analysis and risk assessment results in the PESHE
- If the software cannot contribute to the risk of a mishap occurring, document the rationale for this determination as the results of the software safety analysis in the PESHE (for the software package program)

General Software Safety Steps*

- Step 1 – Start with an identified hazard and system risk assessment
- Step 2 – Perform Software analysis to determine degree of software control for the identified hazard (Software Control Category (SCC))
- Step 3 – Using the SCC and the severity category for the identified system hazard, determine Software Criticality Index (SwCI) and Level of Rigor (LOR) required to evaluate impact of software on the system risk
- Step 4 – Review LOR tasks execution
  - Step 4a - If LOR tasks not completed, assign risk level to hazard based on MIL-STD-882E, Table VI
  - Step 4b – If LOR tasks are completed successfully, use results to reassess system risk of identified hazard

** These are the general software safety steps assuming Scenario One has been determined.*

If Scenario Two has been determined, document the rationale for this determination as the results of the software safety analysis in the PESHE (for the software package program)

Step 2 - Software Control Categories (SCC)

Same definitions as used in the JSSSEH

TABLE IV. SOFTWARE CONTROL CATEGORIES		
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.)*

Step 2 - SCC (cont)

Same definitions as used in the JSSSEH

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.

Step 3 - Software Safety Criticality Matrix (SSCM)

TABLE V. Software safety criticality matrix

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

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

Step 4 - Relationship Between SwCI and Risk

TABLE VI. Relationship between SwCI, risk level, LOR tasks, and risk

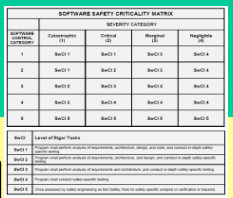
RELATIONSHIP BETWEEN SwCI, RISK LEVEL, LOR Tasks, AND RISK		
Software Criticality Index (SwCI)	**Risk Level**	**Software LOR Tasks and Risk Assessment/Acceptance**
SwCI 1	**High**	• If SwCI 1 LOR tasks are unspecified or incomplete, the contributions to system risk will be documented as HIGH and provided 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.

Characterizes the System Safety responsibilities to the PM for software system safety. Life-cycle independent

Software Safety Analysis and Verification Process

Top-Level Process

Software Criticality Matrix



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

Conclusion

- MIL-STD-882E and DoDI 5000.02 make Software System Safety Engineering and Analysis a clear requirement
  - It is important that software be analyzed within the context of the system it functions in
  - A successful software system safety engineering activity is based on a hazard analysis process, a safety-significant software development process, and LOR tasks
  - Emphasis is placed on the context of the “system” and how software contributes to or mitigates failures and mishaps
  - The software system safety effort should be performed in conjunction with the system safety, software development, software test, configuration management, and Independent Verification and Validation team(s)

DoDI 5000.02 identifies requirements for documentation of the software safety analysis process used and their results

Questions?

Robert E. Smith, CSP
Booz Allen Hamilton
1550 Crystal Drive, Suite 1100
Arlington, VA 22202-4158
703-412-7661
smith_bob@bah.com