14794
MIL-STD-882E:
Software System Safety Process in 882E

Robert E. Smith, CSP, Booz Allen Hamilton
NDIA Systems Engineering Conference
San Diego, CA
October 24, 2012

Agenda

- Purpose
- Software Safety Section Origination
- Software Safety-related Definitions
- Software Control Categories
- Software Contribution to Risk
- Software Hazard Causal Factor Risk Assessment Criteria
- Conclusion

Purpose

- To provide an overview of the Software System Safety (SwSS) process as specified in MIL-STD-882E:
- Section 4.4 – Software Contribution to System Risk (part of the mandatory system safety process)
- Appendix B – Software System Safety Engineering and Analysis

Software Safety Section Origination

- The 882E software safety section is based on established software safety standard practice
  - DoD Joint Software Systems Safety Engineering Handbook (JSSSEH) Version 1.0 Published August 27, 2010
  - Allied Ordnance Publication (AOP)-52 (EDITION 1) – Guidance On Software Safety Design and Assessment of Munitions-Related Computing Systems dated 9 December 2008
- The existing Joint Software System Safety IPT provided Subject Matter Expertise for the 882 effort

General Software Safety Steps

- Step 1 – Start with an identified hazard and system risk assessment
- Step 2 – Perform Software Assessment 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 Table VI
  - Step 4b – If LOR tasks are completed successfully, use results to reassess system risk of identified hazard

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

Conclusion

- MIL-STD-882E makes Software System Safety Engineering and Analysis a clear requirement
- Follows the published and recognized guidelines utilized by government and industry for Software System Safety
- Documents a system safety risk assessment compliant approach for software contributions to system risks

MIL-STD-882E Requires Software System Safety Analyses

Questions?

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

Backups

Table 4-3: Example LOR Tasks or Requirements

Design Requirements	Process Tasks	Test Tasks
Fault Tolerant Design	Design Reviews	Safety-Significant Function Testing
Fault Detection	Safety Reviews	Functional Thread Testing
Fault Isolation	Design Walkthroughs	Limited Regression Testing
Fault Annunciation	Code Walkthroughs	100% Regression Testing
Fault Recovery	Independent Reviews	Failure Modes and Effects Testing
Warnings Cautions, and Advisories	Independent Walkthroughs	Safety-Critical Interface Testing
Redundancy	Traceability of Safety-Significant Requirements to Design	COTS, Government Off-the-Shelf Input, Output Test, and Verification
Independence	Traceability of Safety-Significant Requirements to Code	Independent Testing of Prioritized Safety-Related Functions
Functional Partitioning	Traceability of Safety-Significant Requirements to Test	Functional Qualification Testing
Physical Partitioning	Safety Test Results Review	Verification and Validation
Design Safety Standards	Software Quality Assurance Inspections and Audits	Independent Verification and Validation
Design Safety Guidelines	Traceability of Safety-Significant Requirements to Hazards	Full Screening of All COTS Features
Design Safety Lessons Learned	Specific Software Language Requirements	
Full COTS Features Disclosure and Analysis		

Table 4-4: Example of Specific LOR Tasks

Level of Rigor	Requirements Tasks	Design Tasks	Implementation Tasks	Test and V&V Tasks	Lifecycle Support Tasks
LOR-1 High Risk	• All LOR-2 and -3 Tasks • Create requirements for a fault tolerant design [Safety and Requirements and Design] • Create requirements to ensure that all interfaces are validated and controlled at all times [Safety and Requirements and Design]	• All LOR-2 and -3 Tasks • Functionally partition all implementations of LOR-3 requirements from lower levels of rigor in the design [Design] • Update design to be stress tolerant [Design] • Update design for SCF (and only SCF) philosophy [Safety and Design] • Update design to control functional, physical, and human interfaces [Design]	• All LOR-2 and -3 Tasks • Create a unit test plan defining the criteria for unit testing of safety-critical code [Safety and Development] • Review unit test results and verify that the unit tests provide the required unit test coverage and were executed in compliance with the unit test plan [Safety and Test] • Perform detailed inspections of code for compliance with safety-critical coding standards and guidelines [Test] • Perform detailed code inspections for fault contributions [Safety, Test, and Development] • Create unit tests with goal of approaching 100% source code branch-point unit testing [Development]	• All LOR-2 and -3 Tasks • Add safety-critical integration test cases to the formal test plan(s) [Safety and someone independent of the developer] • Execute safety-critical integration and test cases [Someone independent of the developer] • Add fault injection safety-critical test cases to the formal test plan(s) [Safety and Test] • Execute fault injection testing [Test] • Add test cases to the Regression Test Plan to support 100% regression testing on all LOR-3 software [Safety, Test, and Development] • Perform 100% regression testing on all LOR-3 software that is changed [Test] • Test to verified and validated interfaces [Test] • Perform code walkthroughs and review all LOR-3 code for safety issues [Safety and Development] • Add safety test cases to verify that all functional, physical, and human interfaces are under continuous control [Safety and Test] • Execute complete decision coverage of the code [Test] • Execute complete modified condition/decision coverage of the code	• All LOR-2 and -3 Tasks

LOR-2 Med Risk	• All LOR-3 Tasks • Review safety-critical requirements for completeness [Safety]	• All LOR-3 Tasks • Perform a Sub-System Hazard Analysis [Safety] • Functionally partition all implementations of LOR-2 requirements from lower levels of rigor in the design [Design] • Incorporate fault isolation, annunciation, and tolerance into the design [Design]	• All LOR-3 Tasks • Perform high-level reviews of code for compliance with safety-critical coding standards and guidelines [Safety and Development] • Independently witness the execution of unit tests [Safety and Test] • Review unit test plan [Safety]	• All LOR-3 Tasks • Create test cases for safety-critical code to test for [Safety, Test, and Development]: - Stress testing - Stability testing - Disaster testing • Review each LOR-2+ test case [Safety] • Participate in test anomaly resolution [Safety] • Plan, perform, and review failure modes and effects testing (FMET) plans and procedures. • Plan, perform, and review functional and FMET regression test plans and procedures.	• All LOR-3 Tasks • Review defects for safety impact [CM and Safety] • Review and give signature approval on safety-critical Change Requests (CRs) [Safety] • Independently review and check in code changes to CM [someone other than the author of the changes]
LOR-3 Low Risk	• Perform a System Hazard Analysis [Safety] • Create a traceability matrix from safety-critical requirements (contributing or mitigating) to identified hazards (of initial RAC Medium or High) [Safety] • Review safety-critical requirements and prioritize for future builds [Safety and Requirements] • Create requirements to ensure that safety-critical interfaces are validated and controlled at all times [Safety, Requirements, and Design] • Map safety requirements to functions and into views of the system and software architectures, labeling COTS and NDI as they become "make-or-buy" outcomes.	• Continue System Hazard Analysis [Safety] • Follow design guidelines for safety-critical design [Design] • Analyze the design (including functional systems and software architectures and interfaces) for failure modes and hazard contributions [Safety and Design] • Review the design for compliance with the design guidelines and for safety issues [Safety and Design] • Review of the User Interface design for safety issues [Safety and Design] • Create traceability from design components to safety-critical requirements [Requirements and Design]	• Continue System Hazard Analysis [Safety] • Mark safety-critical code with the appropriate LOR [Development] • Follow coding guidelines and comply with coding standards for safety-critical code [Development] • Create traceability from code to safety-critical design requirements [Design and Development] • Execute unit tests [Development] • Participate in acceptance review of safety-critical code [Safety] • Create a safety-critical test report documenting the safety-critical unit testing compliance and execution results [Safety and Test]	• Continue System Hazard Analysis [Safety] • Create test cases for safety-critical code [Safety, Test, and Development] - Exception handling correctness - Fault handling correctness - Interface correctness - Boundary handling correctness • Review safety-critical test results and verify that the safety-critical test cases provide the required test coverage and were executed in compliance with the formal test plans [Safety and Test] • Create traceability between safety-critical test cases and safety-critical requirements [Safety and Test] • Mark safety-critical test cases with the appropriate LOR [Safety and Test] • Create a safety-critical test report documenting the safety-critical formal testing compliance and execution results [Safety and Test] • Calculate and document the residual safety risk (after mitigation) [Safety] • Review all traceability matrices for coverage and completeness [Safety and Design]	• Review proposed CRs for safety impact [Safety and Requirements] • Mark safety-critical items in CM with the appropriate LOR [Development and CM] • Document the results of any Safety Reviews [Safety] • Review problem reporting/defect tracking, change control, and change review activities for safety impact and compliance [CM and Safety] • Perform an Operations and Support Hazard Analysis [Safety]

Section 3.2 Definitions

- 3.2 Definitions are mandatory.
- Key SwSS definitions in -882E
  - 3.2.37 Software. A combination of associated computer instructions and computer data that enable a computer to perform computational or control functions. Software includes computer programs, procedures, rules, and any associated documentation pertaining to the operation of a computer system. Software includes new development, complex programmable logic devices (firmware), NDI, COTS, GOTS, re-used, GFE, and Government-developed software used in the system.
  - 3.2.10 Firmware. The combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device. The software cannot be readily modified under program control.
  - 3.2.18 Level of rigor (LOR). A specification of the depth and breadth of software analysis and verification activities necessary to provide a sufficient level of confidence that a safety-critical or safety-related software function will perform as required.
  - 3.2.24 Non-developmental item (NDI). Items (hardware, software, communications/ networks, etc.) that are used in the system development program, but are not developed as part of the program. NDIs include, but are not limited to, COTS, GOTS, GFE, re-use items, or previously developed items provided to the program “as is”.
  - 3.2.35 Safety-significant. A term applied to a condition, event, operation, process, or item that is identified as either safety-critical or safety-related.
  - 3.2.32 Safety-critical function (SCF). A function whose failure to operate or incorrect operation will directly result in a mishap of either Catastrophic or Critical severity.
  - 3.2.34 Safety-related. A term applied to a condition, event, operation, process, or item whose mishap severity consequence is either Marginal or Negligible.

SCC (cont)

- All SCC should be re-evaluated if legacy software functions are included in system-of-systems (SoS) environment.
- The legacy functions should be evaluated at both the functional and physical interfaces for potential influence or participation in top-level SoS mishap and hazard causal factors.

Re-evaluate software for legacy implication in SoS

Software Contribution to Risk

- For software, the SwCI and LOR define the requirements of mitigation efforts.
- MIL-STD-882E, Appendix B provides guidance on evaluating software's contribution to system risk and additional guidance on software safety engineering and analysis activities.
- The 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.
- If the software design does not provide sufficient evidence that it meets safety requirements, then an assessment shall be made to determine the risk associated with inadequately verified software hazard causes and controls.

Assessing Software's Contribution to Risk

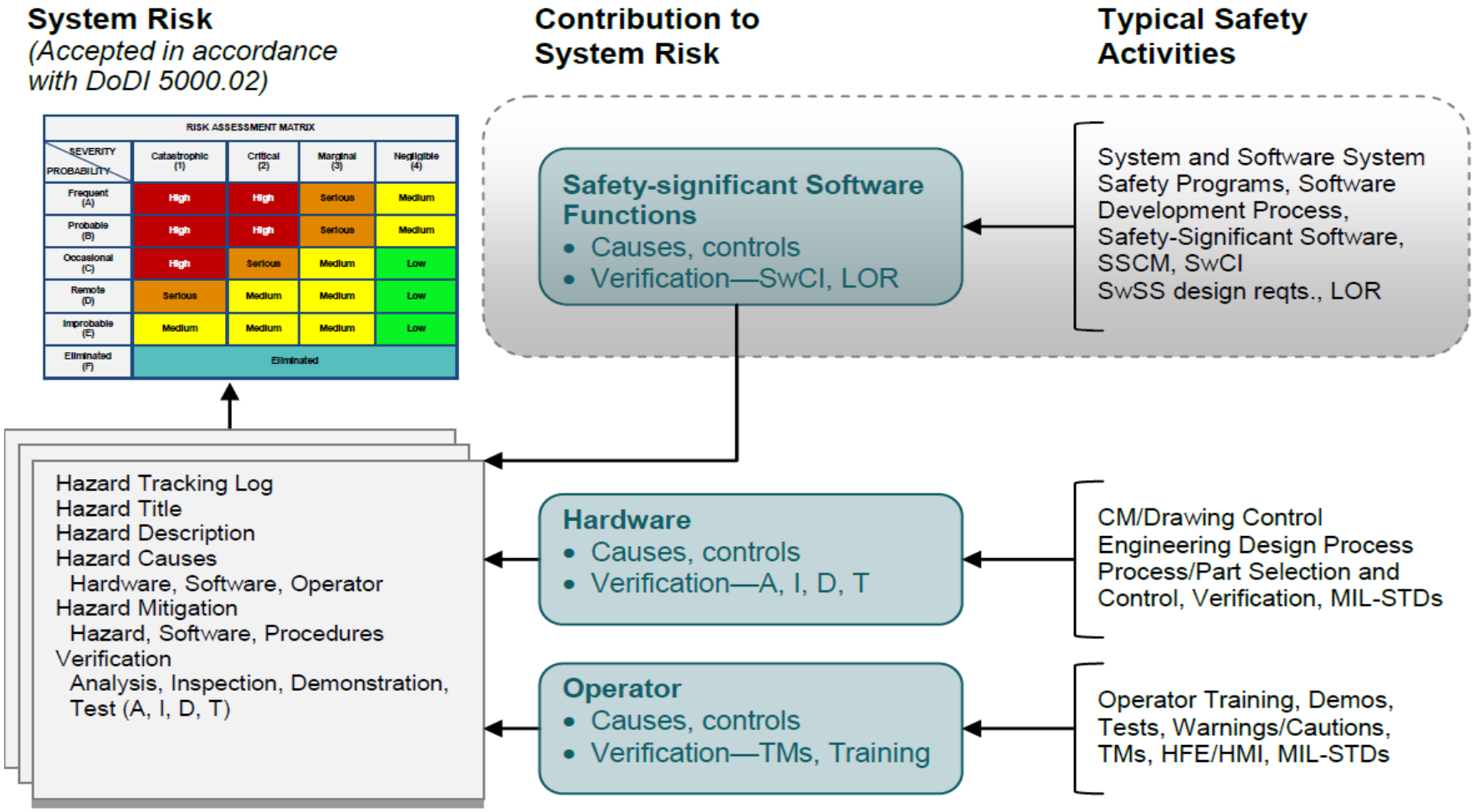

FIGURE B-1. Assessing software's contribution to risk

Table B-I. 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.

This is the table that should be used to assess Software contribution to risk