Software System Safety, Software Criticality, and Software Hazard Control Categories for Information Systems

Mike D. Pessoney: APT-Research: Huntsville, AL

Shannon Stump; The Boeing Company; Huntsville, AL

Keywords: Information Systems, Software System Safety, Software Control Categories, Level of Rigor

Abstract

Faced with the task of assessing software safety in a large military information family of systems, the authors found the sample Software Control Categories matrices described in MIL-STD-882E and in the Joint Software System Safety Engineering Handbook (JSSSEH) inadequate for controlling and evaluating the software safety of the system. For an information system, both the restricted span of control and the lack of control level independence posed problems to the evaluators. Substitute Software Control Categories (renamed Software Hazard Control Categories) were postulated and refined until sufficient fidelity was reached for the Software Hazard Control Categories. These categories were used for the evaluation of software criticality and mapping to a level of rigor plan for software hazard control. This paper describes the methods and the substitute Software Hazard Control Categories developed, and then summarizes the results obtained.

An Information System, as used here, is a system that does not directly control any safety critical hardware or subsystems. All control is performed by an operator or associated system, based in whole or in part, on data provided by the information system.

Introduction

The Software Control Categories (SCC) table is used in software system safety along with the hazard severity table to determine a Software Criticality Index (SwCI). The SwCI is used to assign a Level of Rigor (LOR) or set of assurance task requirements to be applied to a Computer Software Configuration Item (CSCI). Completing the LOR requirements provides the required software safety confidence. See Figure1.

For systems controlled by software, the SCC establishes a relationship between the level of software control and the activities required to build confidence in safety assurance. The relationship between SCC and LOR intensity is direct such that the more control software has over a safety-significant function, the more stringent requirements, design, code, and test/verification activities are necessary for developing the software and assessing the safety of the system.

While this reasoning is valid for systems controlled by software, the reasoning becomes ambiguous for information systems that do not control systems except through the actions of an operator or the actions of an associated system.

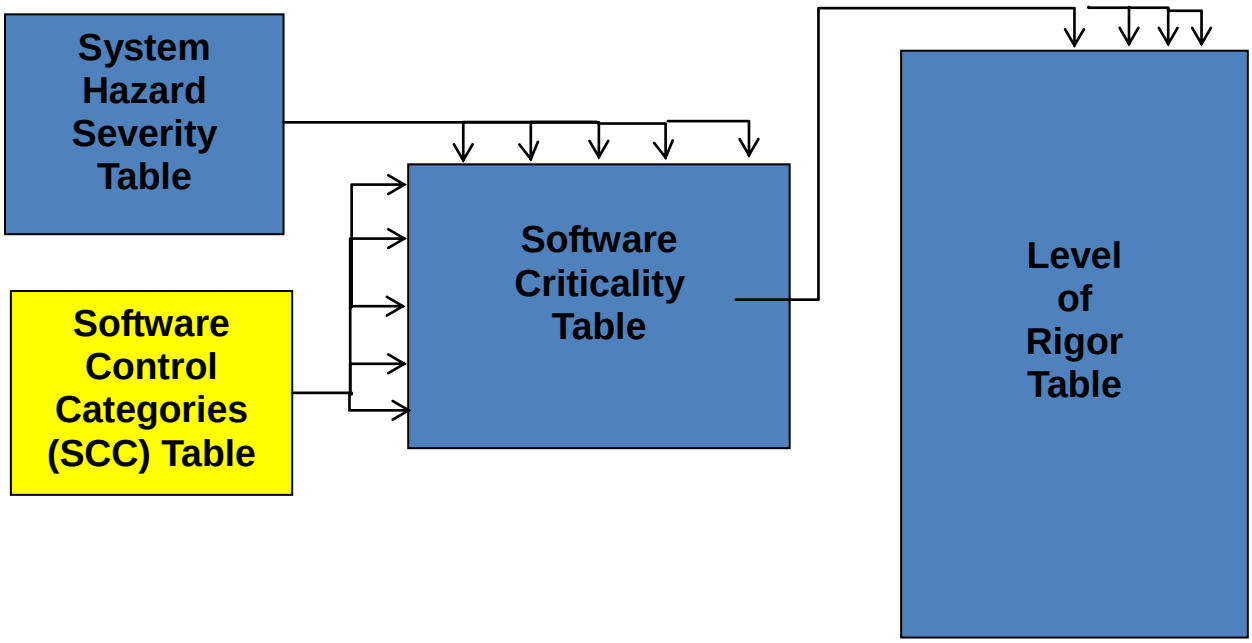


Figure 1 Relation of Software Safety Tables

Discussion

The Software Control Categories example tables available provided sufficient guidance to assess the contributions of software for systems controlling safety critical hardware or subsystems and for some mixed control and information systems, but lacked sufficient fidelity to assess an information-only system. For an information system, MIL-STD-882E (ref 1) and the Joint Software System Safety Engineering Handbook (JSSSEH) (ref 2) tables allow control of only 4 out of 5 levels and neither allows level 1 control from an information system since level 1 control is limited to autonomous operations. See Table 1**Error! Reference source not found.**. Additionally, when assessing the control exercised by a safety-significant software function or safety-significant software requirement, the function or requirement may fit one level, more than one level, or no level at all in existing tables.

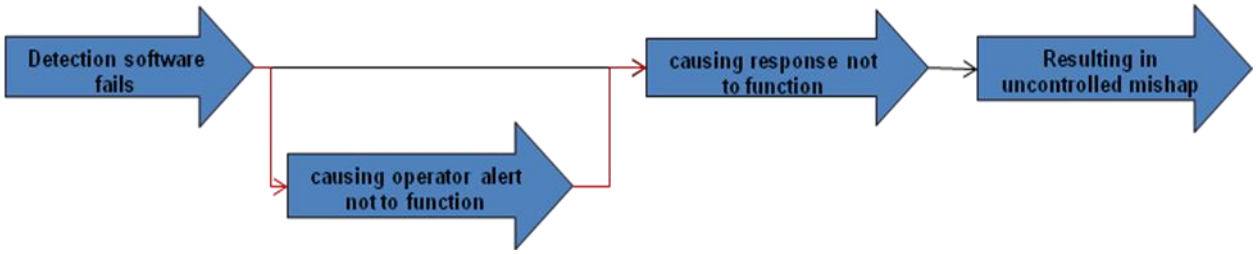
The Software Hazard Control Categories (SHCC) table was developed to bring more fidelity to the software control evaluation for an information system. The intent was to provide more meaningful software control categories for information systems that describe the control levels so they could be more accurately assigned by the analyst. These control categories would cover the same span of control that the control system software covered. The descriptions of system control were not included in the SHCC for this paper since no system control was exercised by the system under study. See Table 2.

Consider two aircraft systems where one has an engine fire control system that detects and responds to an engine fire while the other detects the engine fire condition and alerts the pilot to respond. The engine fire detection software is of equal safety significance in both systems, but is assigned level 1 control in the first scenario and level 2 control in the second scenario. A failure of the detection software has equal chances for a mishap in both systems. When a single software exception, failure, fault, or delay may lead directly to a mishap the software logically should be considered to have the highest level of hazard control. Whether an operator is involved is incidental to the level of control.

Consider two patient oxygen monitors where one has an automated system to increase the oxygen supplied while the other detects a low oxygen condition and alerts hospital personnel to respond. The low oxygen detection software is of equal safety significance in both systems, but is assigned level 1 control in the first scenario and level 2 control in

the second scenario. A failure of the detection software has equal chances for a mishap in both systems. When a single software exception, failure, fault, or delay may lead directly to a mishap the software logically should be considered to have the highest level of hazard control.

Hazard descriptions for the two cases in each example are very similar as seen in the following diagram:



Notice that whether the control is autonomous or operator controlled does not change the initiating cause or the mishap result of the hazard. Conclusion drawn from these examples and the system under study was:

Operator or external system involvement is incidental to the level of control.

Level-of-software-control to assess software criticality is an accurate attribute to use for control systems but a misleading attribute to use for information systems. Certainly the detection software in the hazard example could be identified as level 1 control in both cases, but it may be rarely identified as autonomous in the case an operator is involved to control the hazard.

A safety-significant subsystem is a subsystem containing identified hazards. Software systems that fail to properly control safety-significant subsystems do so because they fail to control one or more of its hazards (not because they fail to control the subsystem). Software systems that fail to provide information to an operator to control a safety significant subsystem also fail to control essentially the same hazard. In both cases, the detection software failure has the same result and should be subjected to the same level of rigor evaluation. Level of software hazard control seems a more accurate attribute to describe both control systems and information systems.

This is not practical within existing Software Control Categories tables and cannot be as long as autonomy is used as a primary discriminator. Autonomy has been used to describe software controlling a safety-significant subsystem with no possibility of intervention by a control entity. Since what is actually controlled is a hazard causing the subsystem to be safety-significant, this is the same as software for which failure provides a sole source for a hazard leading to conclusion 2:

Sole source for a hazard is a more accurate discriminator for software control than autonomous control of a subsystem.

Existing SCC tables list both control and information descriptions for level of control. This paper developed and used only the information descriptions because the information system being evaluated did not control any safety-significant subsystems. A combined table was developed and is presented as Table 3 but was not verified or used.

Existing Software Control Categories

Table 2 contains the common level numbers and level names from the MIL-STD-882E and JSSSEH software control categories tables along with the discriminators used in the tables. The discriminators used for information systems are highlighted in yellow.

		Combined MIL-STD-882E, and JSSSEH 2010 Software Control Categories (SCC)	
Level	Name	MIL-STD-882E	JSSSEH Description
1	AT Autonomous	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.)	Software functionality that exercises autonomous control authority over potentially safety-critical or safety-significant hardware systems, subsystems, and/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 systems/software functionality with multiple subsystems, interacting parallel processors, multiple interfaces, and safety critical functions that are time critical.)
2	SAT Semi-Autonomous	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 functionality that exercises control authority over potentially safety-critical or safety-significant hardware systems, subsystems, and/or components allowing time for predetermined safe detection and intervention by independent safety mechanisms to mitigate or control the 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 that 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 the mishap occurrence. (This definition assumes that the safety-critical display information may be time-critical, but	Software items that display safety-critical or safety significant information requiring immediate operator entity to execute a predetermined action for mitigation or control over a hazard. Software exception, failure fault, or delay will allow or fail to prevent the mishap

		the time available does not exceed the time required for adequate control entity response and hazard control.)	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	RFT Redundant Fault Tolerant	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 functionality that issues commands over safety-critical or safety-significant hardware systems, subsystems, and/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-critical or 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.	Software that generates information of a safety-critical or safety-significant 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.	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	NSI No Safety Impact	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.	Software functionality that does not possess command or control authority over safety-related hardware systems, subsystems, and/or components, and does not provide safety-related information. Software does not provide safety-critical or time-sensitive data or information that requires control entity interaction. Software does not transport or resolve communication of safety-critical or time-sensitive data.
Note: All SCC categories should be re-evaluated if legacy software functions are included in a System-of-System 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.			

Table 1: Combined MIL-STD-882E, and JSSSEH Software Control Categories

Example Software Hazard Control Categories (SHCC) Table for Information Systems

This Software Hazard Control Categories Table was the result of the author's attempts to create a more accurate and usable table for assigning control to information system software. The discriminator "Immediate" can be read as "Now" and describes control of requirements or functions where failure can result in an immediate mishap. The discriminator "Eventual" can be read as "Later" and describes control of requirements or functions where failure can result in an eventual mishap. See examples to clarify.

Example Software Hazard Control Categories (SHCC) Table for Information Systems			
Level	Name	Description	Examples
1	Sole Source Immediate (Now)	A software function or requirement that necessitates immediate response from an operator or external system based on data provided for mitigation or control over a hazard and potential immediate mishap. The software collection, distribution, display, or warning function or requirement provides the only information source.	Single source: Entering a minefield, stall warning, low oil pressure, red traffic light, engine overheat, fire alarm, tornado warning, impending collision, medical evacuation request message, no pulse, breathing interrupted
2	Sole Source Eventual (Later)	A software function or requirement that may necessitate eventual response from an operator or external system based on data provided for mitigation or control over a hazard and potential eventual mishap. The software collection, distribution, display, or warning function or requirement provides the only information source.	Single source: Approaching a minefield, check engine, tornado watch, gas volume low, tire pressure low, communication suite status change notification
3	Redundant Source Immediate (Now)	A software function or requirement that necessitates immediate response from an operator or external system based on data provided for mitigation or control over a hazard and potential immediate mishap. The software collection, distribution, display, or warning function or requirement provides the primary source of information although independent supplemental sources are available.	Multiple independent sources (M): Stall warning(M), low oil pressure (M), red traffic light (M), engine overheat (M), fire alarm (M), tornado warning (M), impending collision (M), medical evacuation request message(M), no pulse(M), breathing interrupted(M)
4	Redundant Source Eventual (Later)	A software function or requirement that may necessitate eventual response from an operator or external system based on data provided for mitigation or control over a hazard and potential eventual mishap. The software collection, distribution, display, or warning function or requirement provides the primary source of information although independent supplemental sources are available.	Multiple independent sources (M): Check engine(M), tornado watch(M), communication suite status change notification (M)
5	NSI No Safety Impact	A software function or requirement that does not necessitate immediate or eventual response from an operator based on data provided to prevent the occurrence of a mishap. It does not provide safety-related or time-sensitive data or information that requires control entity interaction.	Time of day, indoor temperature, phase of moon, current fashions, most internet data

Table 2: Example Software Hazard Control Categories for Information Systems

Experience of use of SHCC table

In reviewing requirements at the Software Requirements Specification (SRS) level in an information system, the software safety team reviewed several thousand requirements yielding nearly a thousand safety-significant requirements, but had great trouble identifying the software criticality of these requirements using the available SCC tables.

The system used for testing this SHCC table was a battlefield information family of systems where many forms of information were provided for Situational Awareness (SA) and Command and Control (C2). The hazards were restricted to the accuracy, timeliness, and completeness of the information provided along with some non-interference hazards associated with co-resident functions. The collection, distribution, display, and warning functions and requirements that affected hazards were designated as safety-significant and traced to the hazards affected. A SHCC value from the table was chosen for each requirement to determine the level of rigor needed to provide confidence in the software. The use of the SHCC table provided consistent SwCI values from the Software Criticality Table for multiple evaluators while the SCC table did not. The SwCI results were the same using the SCC and SHCC tables in this system but were reached with substantially less bias, negotiation, argument, and rework. (This was important in a system involving several analysts and several thousand requirements).

The Software Safety team conducted continuous analyses to "living" (continuously updated) software requirement specifications. The team was able to compare results from utilizing the existing Software Control Categories with results from utilizing the proposed Software Hazard Control Categories. This methodology minimized variability in hazard impact conclusions made by multiple project team members. It is a much easier technique to use for software safety analyses involving multiple analysts due to its straightforward, Boolean nature. In this implementation example, two analysts performed the categorization analysis on the same requirement subsets and results were compared. Results were identical, validating the ease of result replication. In this test, the assignment process proved to be more user-friendly and applicable to the type of system, thus yielding more accurate results and an overall better understanding of the safety impact of the information system. This methodology also helped eliminate potential safety-significant requirements from being incorrectly allocated into the "No Safety Impact" level due to lack of adherence to the descriptions in the remaining levels.

Benefits of Using SHCC Table

The example SHCC table allows information systems software hazard control at all the same levels as hardware control, an approach which seems appropriate. If properly assessed for response time needed and source type using the example SHCC table, a safety-significant software function or requirement will fit exactly one hazard impact level, eliminating variable conclusions. In the event that a safety-significant requirement fails to fit one of the SHCC levels, the designation as safety-significant should be questioned.

Drawbacks of Using SHCC Table

Allowing information system software to be classed as Level 1 may cause additional effort to be expended by software safety. Indeed, if the system provides single source data requiring immediate operator action to control a hazard and avoid an immediate mishap, then additional effort should be required! Compartmentalizing such software to a single CSCI can limit the effort as can providing multiple sources for the data.

Combining the SCC and SHCC Table

Combining the software control categories and software hazard control categories tables may present some difficulties for software safety of a system containing both control and information. Therefore, an example combined table was formed as a guide and is presented as Table 3 below. The control descriptions were not used on the current project.

Combined Table Control-(MIL-STD-882E), Information-(SHCC)		
Level	Name	Description
1	Sole Source Immediate	Software functionality that exercises sole source control over hardware systems, subsystems, or component hazards without the possibility intervention by an independent control entity to preclude the occurrence of a hazard and potential immediate mishap. (This definition includes complex system/software functionality with multiple subsystems, interacting parallel processors, multiple interfaces, and safety-critical functions that are time critical.)
		A software function or requirement that necessitates immediate response from an operator or external system based on data provided for mitigation or control over a hazard and potential immediate mishap. The software collection, distribution, display, or warning function or requirement provides the only information source.
2	Sole Source Eventual	Software functionality that exercises sole source control over hardware systems, subsystems, or component hazards, without the possibility of intervention by an independent control entity to preclude the occurrence of a hazard and potential eventual mishap.. (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.)
		A software function or requirement that may necessitate eventual response from an operator or external system based on data provided for mitigation or control over a hazard and potential eventual mishap. The software collection, distribution, display, or warning function or requirement provides the only information source.
3	Redundant Source Immediate	Software functionality that exercises control over systems, subsystems, or component hazards requiring an independent redundant control entity to complete the control function and control the hazard and potential immediate mishap. (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.)
		A software function or requirement that necessitates immediate response from an operator or external system based on data provided for mitigation or control over a hazard and potential immediate mishap. The software collection, distribution, display, or warning function or requirement provides the primary source of information although independent supplemental sources are available.
4	Redundant	A software function or requirement that may necessitate eventual

	Source Eventual	response from an operator or external system based on data provided for mitigation or control over a hazard and potential eventual mishap. The software collection, distribution, display, or warning function or requirement provides the primary source of information although independent supplemental sources are available.
5	NSI No Safety Impact	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.
Note: All SCC categories should be re-evaluated if legacy software functions are included in a System-of-System 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.		

Table 3: Combined Table SCC (MIL-STD-882E) and, Information (SHCC)

Conclusions

Use of the example SHCC matrix is recommended for more accurately determining the software hazard control for system requirements and/or functions within an information system. This methodology has thus far yielded successful results for its charter execution by the software safety team supporting an Army information family of systems program. The methodology developers believe that its applicability and beneficial contributions are universal to not only systems similar to the example explained in this paper, but to most if not all information systems. For application to a combined control and information system, the SHCC table would have to be expanded to address software control of safety-significant subsystems using Table 3 as a guide. This combined table was not needed or used in the system under review. Using the SHCC involves a young methodology implying the existence of some potential drawbacks, but the developers believe this is a favorable solution to the challenge presented in the abstract.

References

1. U.S. Department of Defense. MIL-STD-882E, *Standard Practice for System Safety*. Rev. E, 2012.
2. U.S. Department of Defense. JOINT SOFTWARE SYSTEMS SAFETY ENGINEERING HANDBOOK, Version 1.0, August 27, 2010

Biography

Mr. Pessoney is currently a Software System Safety Engineer working in software system safety supporting the U.S. Army's Aviation and Missile Research, Development, and Engineering Center (AMRDEC) Software Engineering Directorate (SED). Mr. Pessoney has worked in the Military and Space Software Industry for 45 years as a Software Engineer, Software Development Manager, and System Software Safety Engineer. Major programs supported include software development for the Apollo program, Site Defense Ballistic Missile Defense, P3-B/P-3C communications programs, Grizzly Remote I/O Modules, and Abrams and Bradley Diagnostics, as well as software system safety for the Apache, Gladiator, NLOS, and JBC-P systems. Mr. Pessoney received his BS and MA in Mathematics from Sam Houston State University in Huntsville, Texas. Mr. Pessoney has been an active member of the System Safety Society (SSS) for 7 years and has served as VP and President of the TVC Chapter.

Ms. Stump is currently a System Safety Engineer supporting National Aeronautics and Space Administration (NASA). Ms. Stump previously supported the U.S. Army's Aviation and Missile Research, Development, and Engineering Center (AMRDEC) Software Engineering Directorate (SED). Ms. Stump supported the Joint Battle Command Platform (JBC-P) program as a Junior Software Safety Engineer. Ms. Stump received her BS in Industrial and Systems Engineering from Auburn University in Auburn, Alabama.