GUIDANCE FOR ADDRESSING SOFTWARE COMMON CAUSE FAILURE IN HIGH SAFETY-SIGNIFICANT SAFETY-RELATED DIGITAL I&C SYSTEMS

Draft C

Prepared by the Nuclear Energy Institute
May 2021

© NEI 2021. All rights reserved.

Acknowledgements

NEI would like to thank the NEI DI&C Working Group for developing this document.

NOTICE

Neither NEI, nor any of its employees, members, supporting organizations, contractors, or consultants make any warranty, expressed or implied, or assume any legal responsibility for the accuracy or completeness of, or assume any liability for damages resulting from any use of, any information apparatus, methods, or process disclosed in this report or that such may not infringe privately owned rights.

© NEI 2021. All rights reserved.

Executive Summary

Implementation of digital technology at nuclear power stations can provide significant benefits in component and system reliability which can result in improved plant safety and availability. However, a software defect in a digital system or component can introduce a safety hazard through a potential software common cause failure (CCF).

Branch Technical Position (BTP) 7-19, Revision 8, provides three separate methods licensees can use to eliminate CCF hazards from further consideration. These three methods are (1) use of diversity within the DI&C system, (2) use of testing, or (3) use of defensive measures. NEI 20-07 is best aligned with the third method presented in BTP 7-19, Revision 8 (Section 3.1.3) – the use of defensive measures. NEI 20-07 provides objective criteria in the form of safe design objectives (SDOs) that are used to provide a defense against software CCF resulting from a software design defect. The SDOs are used for selection of platform hardware and software, and the development of application software. The approach in NEI 20-07 begins by establishing a set of first principles for the protection against software CCF in digital instrumentation and control (DI&C) systems and then subsequently the decomposition of these first principles into SDOs. This document also proposes using an assurance case method to demonstrate that the software application and the platform that hosts the application software in a high safety-significant safety-related (HSSSR) system have been adequately addressed. When a Diversity and Defense-in-Depth (D3) analysis for the HSSSR system is performed, and the assurance case demonstrates that the platform and the associated application software has adequately addressed CCF, then these parts of the system can be exempted from being postulated as a source of CCF. This does not exclude the need for an HSSSR system D3 analysis because other CCF vulnerabilities may be identified (e.g., data communications).

Numerous industries have managed to successfully implement software-based digital technology. Many of these industries manage extremely dangerous processes yet have found a way to safely and reliably operate using digital technology by capitalizing on the use of quality software design standards, such as IEC 61508. Research conducted by EPRI on the use of IEC 61508 for software development has revealed conclusive results that demonstrate IEC 61508 safety integrity level (SIL) certified digital equipment achieve their designated SIL reliability targets [4]. EPRI conducted a review of software-based platforms with over 1.6 billion hours of operation. The software used in these platforms was designed to a Safety Integrity Level (SIL) Systematic Capability of 3 as defined in IEC 61508. The research revealed no evidence that any of the platforms experienced a software CCF during the more than 1.6 billion hours of operation. Based on this research, it can be reasonably concluded that use of the guidance in IEC 61508 when developing platform software and extrapolating to application software will result in reasonable assurance that a latent software defect will not lead to a software CCF.

Development of the guidance in this document began by establishing a set of first principles for the protection against software CCF in HSSSR digital I&C (DI&C) systems. The software CCF first principles were derived and synthesized from EPRI research and industry operating experience and are used to establish a framework for industry consensus on the fundamental principles upon which an approach to adequately address software CCF can be developed. From these software CCF first principles a set of safe design objectives (SDOs) were established for application software, synthesized from IEC 61508 and other industry standards, that address the software CCF first principles. The SDOs are for use in selection of a platform and when developing application software in HSSSR systems and components. Documenting an assurance case based on adherence to these SDOs will facilitate the demonstration of

© NEI 2021. All rights reserved.

reasonable assurance that the target software contains no software design defects that could lead to a software CCF.

The guidance in this document is intended to be applied on digital upgrades to HSSSR systems and equipment that require a license amendment to implement. The Assurance Case developed through use of this document will be part of the license amendment request documentation package.

NEI 20-07 applies to all holders of operating licenses under Title 10 of the Code of Federal Regulations (10 CFR) Part 50, "Domestic Licensing of Production Facilities" and all holders of combined licenses under 10 CFR Part 52, "Licenses, Certifications, and Approvals for Nuclear Power Plants." Although the guidance in NEI 20-07 primarily focuses on power reactors, other licensees may also use the guidance in NEI 20-07 when selecting platforms and developing application software in HSSSR systems. However, certain aspects of NEI 20-07 guidance discuss regulatory requirements that may not fully apply to these licensees (e.g., Appendix B, "Quality Assurance Criteria for Nuclear Power Plants and Fuel Reprocessing Plants").

This document was developed by the NEI Digital I&C Working Group, in support of the industry response to Modernization Plan #1 (MP#1) Protection Against Common Cause Failure in the NRC's Integrated Strategy to Modernize the Nuclear Regulatory Commission's Digital Instrumentation and Control Regulatory Infrastructure (SECY-16-0070, ADAMS Accession No. ML16126A140). MP#1, contained in Enclosure 1 of SECY-16-0070, is identified as a high priority within the NRC Action Plan.

© NEI 2021. All rights reserved.

TABLE OF CONTENTS

© NEI 2021. All rights reserved.

1 Introduction

Digital instrumentation and control (DI&C) systems can be vulnerable to a software common cause failure (CCF) as a result of a latent defect in the software or software developed logic, which could defeat the redundancy achieved by the system architecture. When identical digital equipment is applied across multiple trains of a safety related system, an undetected software defect could be triggered by certain plant and/or system conditions and cause a simultaneous failure of multiple safety related trains. Similarly, when previously separate control functions are combined within the same digital component or system, a latent software defect that is triggered by an untested condition can result in simultaneous failure of multiple functions.

These types of common cause systematic failures may not have been considered in the plant safety analyses while random failures (e.g., hardware failures due to a degradation mechanism) are better understood. This document focuses on systematic failures due to a latent defect in software, and an approach to providing reasonable assurance through a quality software development process that the common cause systematic failure of an application is adequately addressed.

This approach begins by establishing a set of first principles for the protection against software CCF in high safety-significant safety-related (HSSSR) digital I&C (DI&C) systems. Appendix A provides a mapping between these first principles and NRC regulation. These CCF first principles, derived and synthesized from EPRI research and industry operating experience, provides a framework for industry consensus on the fundamental principles upon which an approach to adequately address CCF can be developed. From these software CCF first principles a set of safe design objectives (SDOs) are established, synthesized from IEC 61508 and other industry standards, that address the software CCF first principles. Ultimately the licensee would demonstrate, using an assurance case demonstrating compliance to the SDOs, providing reasonable assurance that the HSSSR DI&C system does not have a latent software design defect that could lead to a software CCF, by demonstrating compliance to the SDOs.

2 Background

Compared to their analog counterparts, properly designed digital systems are generally more robust, reliable, and more capable of preventing malfunctions of multiple controlled systems or components using redundancy, logic, and other design attributes. In addition, digital technology can be provided with the ability to select a preferred state on a controlled system in the event of a DI&C failure, thus affording the designer some alternatives that can improve plant safety and reliability. Digital technology can also provide immediate annunciation of problems with associated diagnostic capabilities not available in their analog counterparts.

Software CCFs are the result of latent defects in the software triggered by an untested condition. Once triggered, a software defect can lead to misbehavior of a system or component. The same software defect in multiple trains of a safety-related system can be simultaneously triggered and lead to a software CCF. The greater the likelihood of a software defect, the greater the likelihood of experiencing a software CCF. The inverse is also true – decreasing the likelihood of a software defect will decrease the likelihood of experiencing a software CCF. This document provides an approach to demonstrate that a software CCF is adequately addressed for a HSSSR DI&C system. The approach is based on mature industry standards, primarily IEC 61508, used worldwide in the development of high-quality software used in high safety-significant systems.

© NEI 2021. All rights reserved.

3 Definitions

Application Software - Software logic added to a platform for a particular situation or purpose to satisfy a set of end-user requirements (e.g., post-accident monitoring, reactor trip).

Common Cause Failure (CCF) – Loss of function to multiple structures, systems, or components due to a shared root cause [IEEE 603-2018].

Concurrent Trigger – A triggering condition on multiple segments/elements that occur at or about the same time.

D3 – Diversity and Defense-in-Depth

Defensive Measures – Design attributes to prevent, limit, or reduce the likelihood of a software CCF.

Design Attributes – Hardware and software design features that contribute to high dependability. Such features include built-in fault detection and failure management schemes, internal redundancy and diagnostics, and use of software and hardware architectures designed to minimize failure consequences and facilitate problem diagnosis.

Design Control Measures (DCMs) – The application of a formal methodology to the conduct of product development activities.

Latent Software Defect – Undetected errors in functional requirements, software design, or software implementation.

Platform – Software and hardware that is integrated to provide basic generic functionality for use by various applications (e.g., programmable logic controller).

Process Discipline – Strict adherence to approved and documented methodologies and processes.

Random Failure – A failure occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware [IEC 61508-4, Section 3.6.5].

Safe Design Objective (SDO) – Objective criteria for addressing the potential for a software defect being introduced during the software development and integration processes.

Safety Classification (Classes) – An assignment based on functionality and safety significance. Different safety classifications (classes) require different levels of requirements (e.g., Class 1E versus non-Class 1E, or safety-related and non-safety-related).

Software – The programs used to direct operations of a programmable digital device. Examples include computer programs and logic for programmable hardware devices, and data pertaining to its operation [IEEE 7-4.3.2-2016].

Software CCF – The result of a latent software defect on multiple segments/elements due to a concurrent trigger.

© NEI 2021. All rights reserved.

Software Module – Construct that consists of procedures and/or data declarations and that can also interact with other such constructs [IEC 61508-4, Definition 3.3.5]

Software Tools - A sequence of instructions and commands used in the design, development, testing, review, analysis, or maintenance of a programmable digital device or its documentation. Examples include compilers, assemblers, linkers, comparators, cross-reference generators, de-compilers, editors, flow charters, monitors, test case generators, integrated development environments, and timing analyzers. (Adapted from IEEE Std 610™-1990].

Software Module – Construct that consists of procedures and/or data declarations and that can also interact with other such constructs [61508-4 Clause 3.3.5].

System – Defined as either protection, control or monitoring and comprised of one or more programmable electronic devices, including integrated and supporting elements such as power supplies, sensors and other input devices, data highways and other communication paths, and actuators and other output devices [Adapted from IEC 61508-4].

Systematic Capability – Measure (expressed on a scale of SC 1 to SC 4) of the confidence that the systematic safety integrity of an element meets the requirements of the specified SIL, in respect of the specified element safety function, when the element is applied in accordance with the instructions specified in the compliant item safety manual for the element [61508-4].

Systematic Failure – Related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operation procedures, documentation, or other relevant factors. [IEC 61508-4, Section 3.6.6].

Triggering Condition – System states (conditions) that can manifest a latent software defect and create the potential for a software CCF.

Design Attributes – Design or design process attributes within the target digital equipment to prevent, limit or reduce the likelihood of a CCF. Design or design process attributes can be assembled together to establish a preventive, limiting, or likelihood reduction measure. Defensive measures are distinguished from coping or mitigating measures, which are external to the target digital equipment and credited to maintain the plant in a safe condition after the CCF occurs.

Validation – Confirmation by examination and provision of objective evidence that the requirements for a specific intended use are fulfilled. [61508-4 Clause 3.8.2].

Verification – Confirmation by examination and provision of objective evidence that the requirements have been fulfilled [61508-4].

4 Purpose

The purpose of this document is to:

1. Establish a set of DI&C software CCF first principles to provide a framework for industry consensus on the fundamental principles upon which an approach to adequately address software CCF can be developed. Appendix A provides a mapping between these first principles and existing NRC regulatory framework.

© NEI 2021. All rights reserved.

2. Provide a set of SDOs, representing a decomposition of the first principles, that can be used to demonstrate that a software CCF is adequately addressed.

3. Explain the use of an assurance case to demonstrate that the SDOs are adequately addressed to reach the conclusion that a software CCF is adequately addressed for an HSSSR DI&C system.

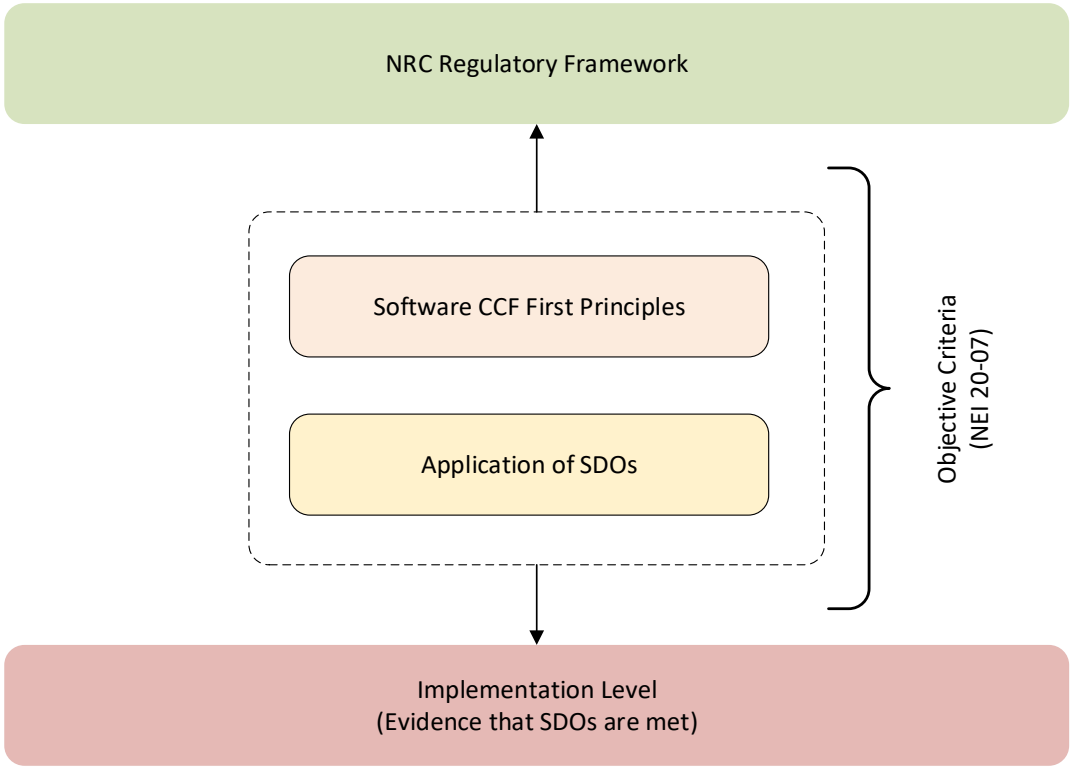
5 NRC Regulatory Framework Versus Implementation Level Activities to Address Software CCF

NEI 20-07 is intended to fill the gap between the NRC regulatory framework and implementation level activities associated with development of HSSSR software. This gap is filled by the establishment of a consensus set of software CCF first principles and the detailed SDOs addressing those first principles.

In contrast, the nuclear industry has developed, and the NRC has endorsed, objective criteria for complying with the regulatory requirements associated with cyber security, electromagnetic compatibility (EMC) and human factors engineering (HFE) related activities. NEI 20-07 will provide objective criteria for evaluation of HSSSR software to adequately address software CCF.

This process of developing objective criteria to demonstrate that further consideration of CCF of HSSSR software due to a software design defect can be eliminated, involves establishing a set of first principles for software CCF for industry consensus, and their relationship to the NRC regulatory framework. These first principles are then decomposed into SDOs that serve to provide criteria for establishing that software CCF is adequately addressed for a HSSSR DI&C system.

Figure 1 below illustrates how NEI 20-07 bridges the gap between the NRC regulatory framework and implementation level activities associated with development of HSSSR software.



© NEI 2021. All rights reserved.

Figure 1
Connection Between NRC Regulatory Framework and Implementation Level Activities

6 First Principles of Protection Against Software CCF

The first principles against software CCF represent a synthesis of EPRI research and industry best software design practices. The first principles listed in this section are considered complete and represent the starting point for decomposition of SDOs. They include the role software design defects play in the initiation of a systematic failure as well as first principle techniques to adequately address the effects of latent software design defects. The first principles of protection against software CCF will be achieved by executing the SDOs.

6.1 Software quality depends on complete and correct requirements, design, review, implementation, and testing

Software quality depends on complete and correct requirements, design, review, implementation, and test. A software defect in an I&C system is an error of commission or omission that results in the related plant systems or components to not function or perform as required by the plant design.

6.1.1 Software design quality depends on requirements quality

Software design depends largely on a complete and correct understanding of the functional and performance requirements of the affected plant systems and components. There is no method or combination of methods that can guarantee 100% complete and correct requirements for a digital-based system. However, requirements engineering methods may be applied with the appropriate rigor depending on the risks due to a requirements error.

6.1.2 Implementation quality depends on design quality and process rigor

It is important to differentiate design quality from implementation quality because design is about decisions based on requirements and architecture while implementation is about realization of software elements based on the design. Design quality is also a function of how completely and correctly the design is expressed and reviewed. While implementation and test quality can be no greater than design quality, inadequate implementation and test quality can result in an incomplete or incorrect realization of the design.

There is no method or combination of methods than can guarantee 100% complete and correct software design and implementation. However, engineering methods can provide some measure of protection against an incomplete or incorrect design and such methods may be scaled and applied with appropriate rigor depending on the risk significance of the affected system elements.

6.2 Concurrent triggering conditions are required to activate a latent software defect

Failures due to a latent defect in software are systematic failures in that a requirements error or omission, an incomplete or incorrect design, or an incomplete or incorrect implementation is a necessary ingredient, as well as the plant or system states that can reveal incomplete or incorrect requirements, design, or implementation. Undetected errors in requirements, design and implementation are called latent defects, and the plant or system states that manifest them (and result in failures) are called triggering conditions.

© NEI 2021. All rights reserved.

When defective DI&C equipment is running in multiple segments of a system and the system does not function or perform correctly due to the latent defect when the system encounters the same plant or system conditions in multiple segments (i.e., a concurrent trigger), the result is a software CCF.

6.2.1 A common defect depends on the quality and commonality of the equipment

A common software defect is a single requirements, design or implementation error that is present in two or more system elements (e.g., subsystems, controllers, control segments, divisions, etc.). If the defect is discovered during system design, test or operation, then it should be corrected. If the defect remains undiscovered (or uncorrected), then it is a latent defect.

6.2.2 A triggering condition depends on system conditions

A latent defect is a requirements, design or implementation error that remains undiscovered because the actual system states or conditions applied or encountered during inspection, test and operations did not reveal it. System states and conditions can range from the plant process states (fluid, electrical, etc.) to faulted conditions (and how they are managed) in the platform or application software.

When in service and system conditions arrive at a state when the latent defect causes an incorrect or incomplete functional response, or the defect causes the system to fail to meet performance requirements, then the defect is considered “triggered”. If actual system conditions are constrained to the same conditions applied or encountered during inspection, test and operations, and all defects discovered during those conditions are corrected, then any remaining latent defects will not be triggered.

6.2.3 A concurrent triggering condition depends on timing and commonality of system conditions

If a latent defect is present in two or more system elements but each element is encountering different conditions, then the likelihood of it being triggered at the same time depends on how much difference there is in the conditions encountered by each element or how much time it takes for each element to encounter the same condition.

For example, a defect may be triggered in one element and detected/corrected in time before the same defect is triggered in another element that encounters the same conditions, provided there is enough time. In this case, the result is not a software CCF.

Note that two or more system elements that have the same latent defect and always encounter the same conditions at the same time will trigger the defect in all elements at the same time if the triggering conditions are encountered. In this case, the result is a software CCF.

6.3 The effects of a software CCF can be reduced by design

First Principles 6.1 and 6.2 are focused on the concept of prevention (albeit without a 100% guarantee) as a means for protection against a software CCF. The principles of limitation, detection and response/recovery also provides means for protection against software CCF with an emphasis on reducing its effects.

© NEI 2021. All rights reserved.

6.3.1 The plant systems or components affected by a software CCF can be limited by design

The principle of limiting the number of plant systems or components that can be physically controlled or affected by a system or subsystem where a software CCF is not adequately prevented will, by design, limit the effects of the software CCF to just those systems or components.

For example, consider a system that applies the elements of one platform, and the system is composed of many control segments where each segment is provided with redundant elements, such as a main/backup pair of controllers. A software CCF of all control segments due to a latent defect in a platform element common to all segments is adequately prevented. However, a pair of controllers in an individual control segment do not encounter sufficiently different conditions such that a software CCF is not prevented within that segment. In this case, limiting the number of plant components per segment will limit the effects of a software CCF in one segment to just those components that are controlled by that segment.

6.3.2 An I&C system can be designed to force a preferred state in the event of a software CCF

Software diagnostic features not subject to the software CCF can provide a means to detect and respond by forcing an I&C system to a preferred state in the event of software CCF. A preferred state may be fail-as-is, fail-off, shutdown, etc., with an attendant notification or alarm.

6.3.3 Detection of an event or condition due to a software CCF provides an opportunity for response and recovery

Detection of a software CCF provides an opportunity to respond and recover from the event. If the software CCF occurs in a system that can initiate a plant event, or it occurs in a mitigating system that is required to respond to an initiating event, then independent means for detection and response via automation and/or manual action can terminate the sequence of events within acceptable limits.

6.4 Operating history can provide evidence of software quality

Operating history can provide evidence of adequate software quality. The depth and rigor of acceptable operating history (e.g., relevant, successful, substantial, available errata, etc.) from all safety industries can also be scaled and matched to the risk of a software CCF in various system elements.

7 Scope and Applicability

Although the technical guidance in this document may be applied to any system or component that contains software, the primary focus is on HSSSR DI&C systems. Risk insights from site-specific probabilistic risk assessments (PRAs) can be used to support the safety-significance determination in categorizing the DI&C system or component. Use of such risk insights should be an input to an integrated decision-making process for categorizing the proposed DI&C system or component. The two criteria below are additional inputs to consider in determining the high safety-significant categorization:

1. Safety-related SSCs relied upon to initiate and complete control actions essential to maintain plant parameters within acceptable limits established for a DBE; or

© NEI 2021. All rights reserved.

2. Safety-related systems and equipment whose failure could directly lead to accident conditions that may cause unacceptable consequences (i.e., exceeds acceptable limits for a DBE) and no other safety-related systems or components are able to provide the safety function.

8 Software CCF Evaluation Process

The software CCF evaluation process is illustrated in Figure 2.

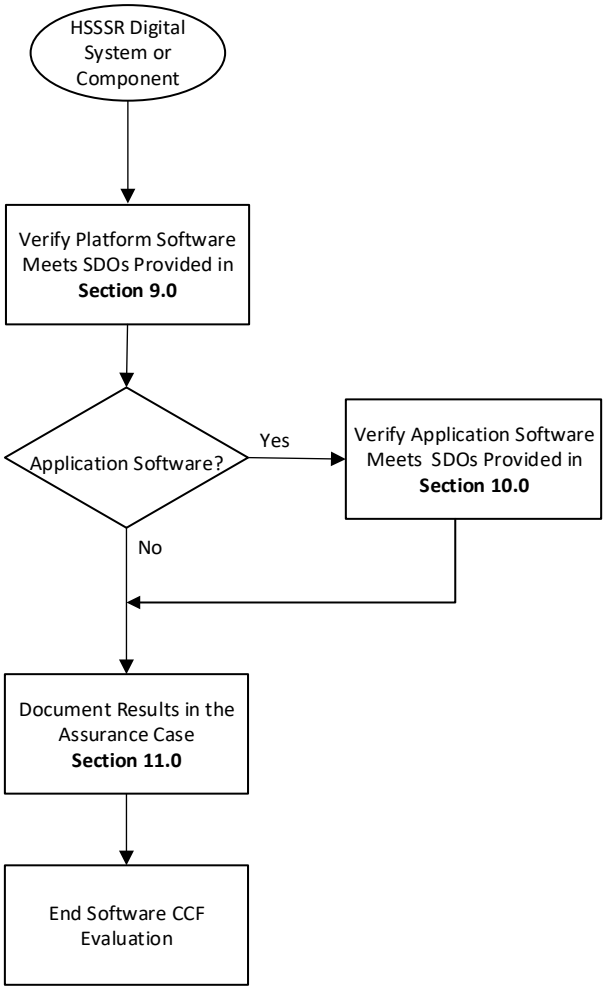


Figure 2
HSSSR Software CCF Evaluation Process

Section 9 below provides goals and SDOs for evaluating HSSSR platform software. Section 10 provides goals and SDOs for evaluating HSSSR application software. Section 11 describes the elements of an assurance case to clearly document adherence to the SDOs as well as any exceptions taken to the guidance in this document.

© NEI 2021. All rights reserved.

9 Software at the Platform and Platform Integration Levels

9.1 Platform Software Systematic Capability

EPRI report 3002011817, Safety Integrity Level (SIL) Certification Efficacy for Nuclear Power [4] reviewed failure data associated with nine operating platforms with SIL 3 certification based on IEC 61508 criteria. The platforms reviewed had a cumulative operating history of over 1.6 billion hours (after SIL3 certification). The researchers found that for "those cases where systematic failures caused the estimated field failure rate to exceed the predicted failure rate, the systematic failures typically resulted from manufacturing process issues, and in no cases did they result from software faults, including software common cause failures.". The report concluded that SIL certifications appear to be an accurate indicator of platform level reliability. If high reliability is achieved via the process of SIL 3 certification then the converse can be assumed: low failure rates can be achieved. A software CCF needs a latent defect and will manifest itself when coincident with a trigger mechanism. A platform certified with a significant low failure rate is also certified in terms of systematic capability, and often the two are comparable. The IEC 61508 standard provides means for assuring a quantifiable failure rate as well as a corresponding means for assuring confidence in systematic capability.

When determining the achieved random capability (SIL) and systematic capability (SC) of a platform, the certifier determines the safe and dangerous failure modes of its functions. When a platform is applied, the designer is responsible for assuring any constraints in the platform certification are implemented in the applied functions in order to achieve the requisite random and systematic capability at the application level. IEC 61508 defines 4 safety integrity levels, labeled from SIL 1 to SIL 4. SIL 3 is the highest safety integrity level that is commonly necessary for the most safety significant industrial operations. SIL 3 certification involves the requirements for high levels of resistance to systematic design failures.

The Platform OEM is responsible for determining the functional safety requirements, selecting certified equipment consistent with those requirements, and verifying the application achieves the necessary random and systematic capability for the safety functions. Platform functions that are rated for SIL 3 operation can be used for high reliability targets of ≥E-4 to <E-3 probability of failure on demand without redundancy.

SIL ratings determine the functional safety requirements that need to be fulfilled. There are different recommendations for software development and design techniques based on SILs. The SIL and SC rating is based on quantitative and qualitative factors such as the development process.

EPRI's report on SIL certification [4] concludes that, "SIL 3 (the second most rigorous of the four defined in IEC-61508) requirements, takes a deep look into the product's hardware and software, as well as the project's functional safety management processes and documentation, to demonstrate the product's safety integrity for performing safety functions. IEC-61508 provides objective and effective criteria to ensure functional safety." The EPRI report further states that, "The SIL 3 certification process is rigorous enough that many products "fail" a certification audit, at least the first time around (i.e., they do not achieve SIL 3 certification without needing some sort of design change). The most common type of design change needed is an improvement in diagnostic coverage. Superior diagnostics, along with the associated programming to ensure the equipment is placed in a safe state once the diagnostics detect a failure, drive the safe failure fraction up by converting dangerous undetected failures into safe detected failures. This is often necessary to satisfy the SIL 3 failure rate requirements, as well as the SIL 3

© NEI 2021. All rights reserved.

architectural constraints." The SIL3 certification process is not an audit but rather a systematic and methodical assessment against the criteria of IEC 61508 by an accredited third party.

IEC 61508 describes Systematic Capability (SC) as a "measure (expressed on a scale of SC 1 to SC 4) of the confidence that the systematic safety integrity of a device meets the requirements of the specified SIL, in respect of the specified safety function, when the device is applied in accordance with the instructions specified in the device safety manual." Systematic capability is determined with reference to the requirements for the avoidance and control of systematic faults. For a device comprised of hardware and software, the interactions between hardware and software failure mechanisms are considered.

The NRC Regulatory Guides and endorsed IEEE standards are intended to meet 10 CFR 50 Appendix B quality requirements. They are process criteria that when applied, allows the treatment of software CCF as a beyond design basis condition. The IEC 61508 standard not only provides quality process criteria, but also includes software design techniques and measures to assure sufficient systematic capability. The quality process and the design techniques and measures are both applied at the platform level and application level. Applying this additional level of design techniques and measures in IEC 61508 can allow for the conclusion to be reached that a CCF sourced by a SIL3/SC3 certified platform does not need to be postulated.

A CCF is a type of systematic failure, and therefore, CCF is directly addressed by addressing systematic capability. Based on the rigor of IEC 61508 SIL/SC3 certification to achieve sufficient systematic capability and the results of the EPRI report, use of a platform certified to the requirements of SIL 3 and SC3 provides reasonable assurance that software CCF in the platform has been adequately addressed and need not be considered a source.

Note that certification to the requirements of SIL 3 and SC3 is not enough to conclude a platform can be used in a safety related application. The platform must meet applicable 10 CFR 50 Appendix B quality requirements as well as any specific environmental qualification requirements. Appendix B quality and environmental qualification requirements are beyond the scope of this document.

9.1.1 Goals

The safe design objectives for platform software systematic capability are intended to achieve the objectives or properties provided in the following clauses of IEC Std. 61508-3:

- 7.4.1 – Objectives
- 7.4.2 – General Requirements
- 7.4.3 – Requirements for Software Architecture Design
- 7.4.4 – Requirements for Support Tools, Including Programming Languages
- 7.4.5 – Requirements for Detailed Design and Development – Software System Design
- 7.4.6 – Requirements for Code Implementation
- 7.4.7 – Requirements for Software Module Testing

© NEI 2021. All rights reserved.

- 7.4.8 – Requirements for Software Integration Testing
- 7.5.2 – Requirements for Programmable Electronics Integration (Hardware and Software)
- 7.7.2 – Requirements for Software Aspects of System Safety Validation
- 7.9.2 – Requirement for Software Verification

9.1.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.4 - Operating history can provide evidence of software quality

9.1.3 Safe Design Objectives

Safe design objectives for achieving platform software requirements quality are listed below:

9.1.3.1 The platform software, including user programmable integrated circuits (such as FPGA, CPLD, ASIC, etc.), meets or exceed a systematic capability of SC3 (as for a SIL 3 system) as described in IEC Std. 61508-3. If a platform does not have SC3 certification, the assurance case should demonstrate how the platform meets the SIL 3 criteria in IEC 61508-3.

9.1.3.2 The platform is used in the HSSSR system consistent with the SIL3/SC3 safety function certification and its safety manual.

9.2 Platform Software Integration within a System Architecture

9.2.1 Goals

- Platform software elements are described to the extent necessary to enable integration into a system, subsystem, or element
- When a platform software element is re-used or is intended to be re-used in other systems, information about the element is sufficiently precise and complete to support an assessment of the integrity of any safety functions that depend on the re-used element
- Platform software element attributes are defined, including hardware constraints or other software that must be accounted for during integration and application
- Platform software element properties are described in terms of what the element is designed for, including its intended behavior and characteristics

9.2.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

© NEI 2021. All rights reserved.

- First Principle 6.2 - Concurrent triggering conditions are required to activate a latent software defect

- First Principle 6.3 - The effects of a software CCF can be reduced by design

9.2.3 Safe Design Objectives

9.2.3.1 When platform software elements are integrated at the system level, subsystem level, or among other elements, they are integrated in accordance with a safety manual that complies with IEC 61508-2 Annex D or 61508-3 Annex D (for pre-existing platform software elements).

10 Software at the Application and Plant Integration Levels

Similar to the arguments for why the IEC 61508 criteria applied to a platform can allow for the conclusion that a CCF sourced from the SIL3/SC3 certified platform is adequately addressed, the same IEC 61508 criteria is the source for the SDOs for application software, assuming the application software is executed on the certified platform. The SDOs are synthesized from the same IEC 61508-3 software requirements that are applied to SIL/SC certified platforms.

The process of synthesizing the IEC 61508-3 SIL3 requirements into SDOs involved evaluating the standard criteria and converting them into objectives. The SDOs are primarily synthesized from IEC 61508-3, Clause 7. Some clauses are not exactly objective criteria. For example, Clause 7.2.2.1 states, "The first objective of the requirements of this subclause is to specify the requirements for safety-related software in terms of the requirements for software safety functions and the requirements for software systematic capability." As a result, not all clauses are adaptable to an SDO.

The synthesizing process adapted the criteria to application software hosted on a platform. Some IEC 61508-3 clauses may be combined into one SDO as part of the synthesis process.

There are some normative sections of IEC 61508-3 that were not synthesized into SDOs. One example is Clause 6. The reason Clause 6 was not synthesized into SDOs is because the underlying configuration management (CM) principles are not unique to software or digital in general.

There are two normative annexes in IEC 61508-3. Annex A is a set of techniques and measures related to criteria in Clause 7 and are considered implementation approaches. The techniques in Annex A are indicated as either "Not Recommended", Recommended", "Highly Recommended", or silent on any recommendation at all, depending on the SIL. It is expected that the assurance case will demonstrate the use of the techniques, or variations with adequate justification, that are Highly Recommended for SIL3 when providing the argument and evidence to meet the applicable SDO associated with them.

Annex D establishes the requirements for a Safety Manual. Its perspective is commodity software such as a PLC that would be used in multiple applications. Application software development is not about creating reusable elements that would require a safety manual for proper use. SDOs for a safety manual for application software would essentially be all the other SDOs in NEI 20-07.

© NEI 2021. All rights reserved.

Demonstrating that the SDOs for the application software, hosted by the SIL3/SC3 hosted platform, have been met, can provide reasonable assurance that the application software will not be a source of CCF in a system.

10.1 Requirements Quality

10.1.1 Goals

The safe design objectives for application software requirements quality are intended to achieve the following goals:

- Requirements correctly express system functions allocated to application software
- Requirements completely express system functions allocated to application software
- Application software requirements are unambiguous
- Application software requirements are understandable
- Application software requirements provide a basis for verification and validation
- When application software functions of different safety classifications are required in one system, independence between such software functions is expressly required (e.g., software functions within different safety classes do not interact or share data)

10.1.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.2 - Concurrent triggering conditions are required to activate a latent software defect
- First Principle 6.3 - The effects of a software CCF can be reduced by design

10.1.3 Safe Design Objectives

Safe design objectives for achieving application software requirements quality are listed below:

10.1.3.1 Application software requirements are derived from, and backward traceable to, the functional and performance requirements of the affected plant systems and their design and licensing bases.

10.1.3.2 A hazard analysis method is used to identify hazardous control actions that can lead to an accident or loss, and application software requirements and constraints are derived from the identified hazardous control actions. The hazard analysis is focused on the system (not just the application software) and should consider plant-level and system-level functions and processes. The hazard analysis should include faults and failures as well as misbehaviors in the absence of any faults or failures.

© NEI 2021. All rights reserved.

10.1.3.3 The application software requirements resulting from activities performed under SDOs 10.2.3.1 and 10.2.3.2 are sufficiently detailed to support an assessment of functional safety.

10.1.3.4 Hardware constraints on the application software are specified and complete.

10.1.3.5 Application software functional and performance requirements are decomposed from I&C system requirements, the I&C system architecture, and any constraints imposed by the I&C system design.

10.1.3.6 If application software requirements are expressed or implemented via configuration parameters, the specified parameters and their values are consistent and compatible with the I&C platform and the I&C system requirements.

10.1.3.7 If data communications are required between application software elements and/or between application software elements and external systems, data requirements are specified, including best- and worst-case performance requirements. Best-case performance is based on ideal conditions. Worst-case performance is based on conservative assumptions of conditions (e.g., communication retries).

10.2 Application Software General Quality

10.2.1 Goals

The safe design objectives for application software general quality are intended to achieve the following goals:

- The application software design fulfills the specified requirements
- Application software requirements imposed by the hardware architecture are fulfilled, including hardware/software interactions that influence the safety of the equipment under control
- The tools, languages, compilers, run-time system interfaces, user interfaces, and data formats are suitable, and assist in verification and validation activities
- The application software is analyzable and verifiable, and is capable of being safely modified
- The required safety functions designed and implemented via application software are achieved and verified

10.2.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.4 - Operating history can provide evidence of software quality

10.2.3 Safe Design Objectives

© NEI 2021. All rights reserved.

Safe design objectives for achieving application software general quality are listed below:

10.2.3.1 When the application software can include or affect a number and/or variety of system elements, and responsibilities for application software design of such elements are split among two or more organizational entities[1], then a clear division of responsibility (DOR) is developed and agreed upon by all entities, and the DOR is maintained throughout the course of application software development activities.

10.2.3.2 Abstraction and modularity are used to control complexity in the application software design.

10.2.3.3 The application software design method aids the expression of functions; information flow; time and sequencing information; timing constraints; data structures and properties; design assumptions and dependencies; exception handling; comments; ability to represent structural and behavioral views; comprehension by organizational entities who need to understand the design; and verification and validation.

10.2.3.4 Testability and modifiability in the operations and maintenance phase of the system lifecycle is considered during application software design.

10.2.3.5 The application software design method has features that support software modification, such as modularity, information hiding, and encapsulation.

10.2.3.6 Application software design notations are clearly and unambiguously defined.

10.2.3.7 The application software design elements are simple to the extent practicable.

10.2.3.8 The application software design includes self-monitoring of control flow and data flow, and on failure detection, appropriate actions are taken.

10.2.3.9 Application software elements of varying safety classifications shall all be treated as the highest safety classification unless adequate independence between elements of different safety classifications is justified.

10.2.3.10 When a pre-existing application software element is used to implement a system function, it meets the SDOs in Section 10.

10.2.3.11 When the digital equipment consists of pre-existing functionality that is configured via data to meet application-specific requirements, the applied configuration design is consistent with the design of the equipment. Methods are used to prevent errors during design and implementation of the configuration data using specified configuration data structures.

10.3 Application Software Architecture Design Quality

10.3.1 Goals

[1] An organizational entity is either a group of people within a corporate structure or a separate corporate entity.

© NEI 2021. All rights reserved.

The safe design objectives for application software architecture design quality are intended to achieve the following goals:

- The application software architecture design is complete and correct with respect to application software requirements
- The application software architecture design supports freedom from intrinsic design faults
- The method of expressing the application software architecture design promotes simplicity and understandability
- The application software architecture design promotes predictable behavior
- The application software architecture design promotes verifiable and testable design
- The application software architecture design promotes fault tolerance
- The application software architecture design provides defense against common cause failure from external events

10.3.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.2 - Concurrent triggering conditions are required to activate a latent software defect
- First Principle 6.3 - The effects of a software CCF can be reduced by design

10.3.3 Safe Design Objectives

Safe design objectives for achieving application software architecture design quality are listed below:

10.3.3.1 The application software architecture design uses an integrated set of techniques necessary to meet the functional and performance requirements developed via the SDOs in Section 0.

10.3.3.2 Application software architecture design is partitioned into elements or subsystems, and information about each element or subsystem provides verification status and associated conditions.

10.3.3.3 Application software architecture design determines hardware/software interactions unless already specified by the system architecture.

10.3.3.4 Application software architecture design uses a notation that is unambiguously defined or constrained to unambiguously defined features.

© NEI 2021. All rights reserved.

10.3.3.5 Application software architecture design determines the features needed for maintaining the integrity of safety significant data, including data at rest and data in transit.

10.3.3.6 Appropriate software architecture integration tests are specified.

10.4 Application Software Support Tool and Programming Language Quality

10.4.1 Goals

The safe design objectives for application software support tool and programming language quality are intended to achieve the following goals:

- Tools support the production of the application software and its required characteristics
- Tool operation and functionality is clear
- Tool output is correct and repeatable

10.4.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.4 - Operating history can provide evidence of software quality

10.4.3 Safe Design Objectives

Safe design objectives for achieving application software tool and programming language quality are listed below:

10.4.3.1 Application software is supported by on-line and off-line support tools. Off-line support tools are classified in terms of their direct or indirect potential impacts to the application software executable code.

10.4.3.2 An application software on-line support tool is an element of the system under design.

10.4.3.3 Application software off-line support tools are an element of development activities and are used to reduce the likelihood of errors, and to reduce the likelihood of not detecting errors. When off-line tools can be integrated, the outputs from one tool are suitable for automatic input to a subsequent tool to minimize the likelihood of human error.

10.4.3.4 Offline tools have specified behaviors, instructions, and any specified constraints when 1) they can directly or indirectly contribute to the executable code, or 2) they are used to support the test or verification of the design or executable code where errors in the tool can fail to reveal defects.

10.4.3.5 Offline tools are assessed for the reliance placed on them and their potential failure mechanisms that may affect the executable application software when 1) they directly or indirectly contribute to the executable code, or 2) they are used to support the test or

© NEI 2021. All rights reserved.

verification of the design or executable code where errors in the tool can fail to reveal defects.

10.4.3.6 Offline tool conformance to its documentation may be based on a combination of history of successful use (in similar environments and for similar applications) and its validation.

10.4.3.7 Tools are validated with a record of their versions, validation activities, test cases, results, and any anomalies.

10.4.3.8 When a set of tools can function by using the output from one tool as input to another tool then the set is regarded as integrated and they are verified to ensure compatibility.

10.4.3.9 The application software design representation or programming language uses a translator that is assessed for suitability at the point when development support tools are selected. The suitability assessment evaluates qualities such as the use of defined language features, support for detection of mistakes, and support for the design method for the project.

10.4.3.10 If SDO 10.4.3.9 is not fully demonstrated, then the fitness of the language and any measures to address identified shortcomings is justified.

10.4.3.11 Programming languages for developing application software are used per a suitable set of rules which specify good practice, prohibit unsafe features, promote understandability, facilitate verification and validation, and specify code documentation requirements.

10.4.3.12 When offline tools are used, the application software configuration baseline information includes tool identification and version, traceability to the application software configuration items produced or affected by the tool, and the manner in which the tool was used, when 1) the tool can directly or indirectly contribute to the executable code, or 2) the tool is used to support the test or verification of the design or executable code where errors in the tool can fail to reveal defects.

10.4.3.13 Offline tools are under configuration management to ensure compatibility with each other and the system under design, and only qualified versions are used, when 1) the tool can directly or indirectly contribute to the executable code, or 2) the tool is used to support the test or verification of the design or executable code where errors in the tool can fail to reveal defects.

10.4.3.14 Qualification of each new version of an offline tool may be demonstrated by qualification of an earlier version if the functional differences will not affect compatibility with other tools, and evidence shows that the new version is unlikely to contain significant faults. The evaluation that the new version is unlikely to contain significant faults includes identifying the changes made for the revision, a review of the verification and validation activities performed on the revision, and review of any relevant operating experience with the revised version.

10.5 Application Software Detailed Design and Development Quality

10.5.1 Goals

© NEI 2021. All rights reserved.

The safe design objectives for application software detailed design and development quality are intended to achieve the following goals:

- The application software detailed design and development is complete and correct with respect to application software requirements developed per Section 0
- The application software detailed design and development demonstrates freedom from intrinsic design errors
- The method of expressing application software detailed design promotes understandability
- The application software detailed design demonstrates predictable behavior
- The application software detailed design is verifiable and testable
- The application software detailed design demonstrates fault tolerance / fault detection

10.5.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.2 - Concurrent triggering conditions are required to activate a latent software defect
- First Principle 6.3 - The effects of a software CCF can be reduced by design

10.5.3 Safe Design Objectives

Safe design objectives for achieving application software detailed design and development quality are listed below:

10.5.3.1 Information items that describe application software requirements, architecture design, and validation planning are completed prior to application software detailed design and implementation activities and are used to inform the detailed design and its implementation.

10.5.3.2 The application software is modular, testable, and modifiable.

10.5.3.3 For each major element or subsystem identified in the application software architecture design produced via the SDOs provided in Section 10.2.3, further refinement into application software modules is based on partitioning, and modules are designed in sets suitable for integration and integration testing at the software and system levels.

10.5.3.4 Application software integration tests and software/hardware integration tests ensure conformance to the requirements produced under the SDOs in Section 0.

© NEI 2021. All rights reserved.

10.6 Application Software Implementation Quality

10.6.1 Goals

The goals for application software implementation quality are as follows:

- The method of expressing the application software implementation is readable, understandable, and testable.
- The application software implementation is performed using the results of SDO 10.4.3.11.
- The application software implementation satisfies the design resulting from the SDOs provided in Section 10.5

10.6.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

10.6.3 Safe Design Objectives

Safe design objectives for achieving application software implementation quality are listed below:

10.6.3.1 Each application software module is reviewed against the goals listed above.

10.6.3.2 When an application software module is produced by an automatic tool, the SDOs provided in Section 10.4 are demonstrated.

10.6.3.3 When an application software module consists of reused pre-existing software, SDO 10.2.3.10 is demonstrated.

10.7 Application Software Module Test Quality

10.7.1 Goals

The goals for application software module test quality are as follows:

- Completeness of module testing with respect to the application software design
- Correctness of module testing with respect to the application software design specification
- Module testing is repeatable
- The module testing configuration is precisely defined

10.7.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

© NEI 2021. All rights reserved.

10.7.3 Safe Design Objectives

Safe design objectives for achieving application software module test quality are listed below:

10.7.3.1 Each application software module is verified (as specified via SDO 10.4.3.5) to perform its intended function and does not perform unintended functions.

10.7.3.2 Application software module testing results are documented.

10.7.3.3 If an application software module test is not successful, corrective actions are specified.

10.8 Application Software Integration Test Quality

10.8.1 Goals

The goals for application software integration test quality are as follows:

- Completeness of integration testing with respect to the application software design
- Correctness of integration testing with respect to the application software design specification
- Integration testing is repeatable
- The integration testing configuration is precisely defined

10.8.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

10.8.3 Safe Design Objectives

Safe design objectives for achieving application software integration test quality are listed below:

10.8.3.1 Using the results of activities performed under SDO 10.5.3.4, application software integration testing is performed using specified test cases, and test data; in a specified and suitable environment; with specified acceptance criteria.

10.8.3.2 Application software integration tests demonstrate correct interaction between all application software modules and/or application software elements/subsystems.

10.8.3.3 Application software integration testing information includes whether test acceptance criteria have been met, and if not, the reasons why such that corrective actions are specified.

10.8.3.4 During application software integration, any module changes are analyzed for extent of 1) impact to other modules and 2) rework of activities performed under prior SDOs.

© NEI 2021. All rights reserved.

10.9 System Integration Quality

10.9.1 Goals

The goals for I&C system integration and test quality are as follows:

- Application software and system hardware are combined in a mutually compatible manner
- System integration is complete and correct with respect to design specifications
- System integration is repeatable
- The integrated system configuration is precisely defined

10.9.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation
- First Principle 6.2 - Concurrent triggering conditions are required to activate a latent software defect
- First Principle 6.3 - The effects of a software CCF can be reduced by design

10.9.3 Safe Design Objectives

Safe design objectives for achieving system integration and test quality are listed below:

10.9.3.1 Application software is integrated with the system hardware in accordance with SDO 10.9.3.2.

10.9.3.2 Using the results of activities performed under SDO 10.5.3.4, system integration testing is performed using specified test types, test cases, and test data; in a specified facility with a suitable environment; using specified software and hardware integration instructions; and with specified acceptance criteria.

10.9.3.3 System integration testing information includes whether test acceptance criteria have been met, and if not, the reasons why such that corrective actions are specified. During application software integration, any module changes are analyzed for extent of 1) impact to other modules and 2) rework of activities performed under prior SDOs.

10.10 System Validation Quality

10.10.1 Goals

The goals for system validation quality in the context of application software functions are as follows:

- The integrated system complies with the requirements developed via activities under the SDOs provided in Section 0

© NEI 2021. All rights reserved.

- System validation is complete and correct with respect to design specifications
- System validation is repeatable
- The validation configuration is precisely defined

10.10.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

10.10.3 Safe Design Objectives

Safe design objectives for achieving system validation quality in the context of application software functions are listed below:

10.10.3.1 System validation procedural and technical steps are specified in order to demonstrate the application software meets the requirements produced via activities performed under the SDOs in Section 0.

10.10.3.2 System validation information includes a chronological record of activities; the validated functions; tools and equipment used; results; and any anomalies - including the reasons why so that corrective actions are specified.

10.10.3.3 For application software, system testing is the primary method of validation, and the system is tested by exercising inputs; exercising expected conditions (both normal and abnormal); and exercising hazards that require system action (as identified via activities performed under SDO 10.1.3.2). Analysis, modeling, and simulation may supplement system testing.

10.10.3.4 Tools used for system validation meet the SDOs provided in Section 10.4.

10.10.3.5 System validation results demonstrate 1) all application software functions required via activities performed under the SDOS in Section 2.1 are met correctly, 2) the application software does not perform unintended functions, 3) test case results information for later analysis or assessment, and 4) successful validation, or if not, the reasons why.

10.11 Application Software Verification Quality

10.11.1 Goals

The goals for application software verification quality are as follows:

- Verification is complete and correct with respect to the results of activities performed under the SDOs in Sections 0 and 10.3 through 10.9, unless such results are already demonstrated via validation activities under the SDOs in Section 10.10
- Verification is repeatable
- The verification configuration is precisely defined

© NEI 2021. All rights reserved.

10.11.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

10.11.3 Safe Design Objectives

Safe design objectives for achieving application software verification quality are listed below:

10.11.3.1 Application software verification activities are specified: selection of strategies and techniques; selection and utilization of tools; evaluation of results; and corrective action controls.

10.11.3.2 Evidence of application software verification activities is recorded, including verified application software configuration items; information used during verification; and the adequacy of results from activities conducted under prior SDOs, including compatibilities between prior activities.

10.11.3.3 Application software functional and performance requirements produced via activities under the SDOs in Section 0 are verified against the I&C system requirements that are identified via SDO 10.1.3.

10.11.3.4 The results of activities performed under the SDOs in Sections 10.2 through 10.6 are verified to ensure conformance to the requirements produced via activities performed under the SDOs in Section 0, as well as completeness, consistency, and compatibility between the results of the activities performed under the SDOs within each Section, and the feasibility, readability, and modifiability of the results produced under the activities of SDOs in each section.

10.12 Protection Against Concurrent, Untested Triggering Conditions

10.12.1 Goals

The goals for protection against concurrent, untested triggering conditions in the context of application software are as follows:

- The number of latent defects in the application software are minimal via preceding SDOs
- Plant and/or plant system conditions that can trigger potentially hazardous behavior in an application software element are identified, then mitigated in the I&C system design
- Concurrent, untested triggering conditions among I&C system elements that have identical application software elements have no impact on those system elements

10.12.2 Associated First Principles of Protection Against Software CCF

- First Principle 6.1 - Software quality depends on complete and correct requirements, design and implementation

© NEI 2021. All rights reserved.

- First Principle 6.2 - Concurrent triggering conditions are required to activate a latent software defect

10.12.3 Safe Design Objectives

Safe design objectives for achieving protection against concurrent, untested triggering conditions in the context of application software are as follows:

10.12.3.1 For each potentially hazardous control action identified via activities performed under SDO 10.1.3.2, causal factor scenarios related to the application software are identified and mitigated.

10.12.3.2 Analysis demonstrates that untested combinations of external and internal I&C system states have no impact on achieving the application software functional and performance requirements resulting from the SDOs provided in Section 0.

10.12.3.3 When equipment under the control (EUC) of the I&C system is normally in the state needed to perform a safety function, the I&C system design has no inputs that will change state when the EUC is in its normal state, and non-normal states in the EUC are readily detectable via means independent of the application software controlling the EUC. Administrative controls limit the duration of non-normal EUC states and limit the EUC in a non-normal state to one channel or division.

11 Diversity

Existing nuclear power plants have a defense-in-depth design. For those plants that do not have an HSSSR digital system, the defense-in-depth design did not consider the potential systematic failure of the HSSSR digital system. When an HSSSR system is replaced with digital technology, a method for analyzing potential systematic failures of the digital HSSSR system is required. The HSSSR system hazard analysis required by SDO 10.1.3.2 involves analyzing the HSSSR potential systematic failures. An important aspect of the HSSSR system hazard analysis is identifying HSSSR systematic misbehaviors in the absence of any HSSSR faults and failures. The results of such a systematic approach to a hazard analysis identifies HSSSR systematic failures that can rise to a CCF.

For each systematic failure identified in the hazard analysis under SDO 10.1.3.2, control methods are determined to avoid the hazard. While implementing the SDOs in Section 10.12.3 or control methods identified to address HSSSR systematic failures in the hazard analysis under SDO 10.1.3.2, diversity may be selected as one attribute of the control method.

12 Assurance Case Development

The Assurance Case is used to document adherence to platform and application software SDOs such that an auditor or inspector can clearly discern how each SDO was applied and how the software development complies with the first principles of protection against software CCF. Any exceptions taken to application of SDOs should be clearly documented with an explanation of why the excluded SDO was not applicable or essential to software development quality. Appendix B provides a suggested roadmap for developing the assurance case.

© NEI 2021. All rights reserved.

IEC 61508-3 Annex A is a set of techniques related to the SDOs and are considered implementation approaches. The techniques in Annex A are indicated as either “Not Recommended”, Recommended”, “Highly Recommended”, or silent on any recommendation at all, depending on the SIL. It is expected that the assurance case will demonstrate the use of the techniques, or variations with adequate justification, that are Highly Recommended for SIL3 when providing the argument and evidence to meet the applicable SDO associated with them.

13 Summary and Conclusion

Creating an HSSSR system using a SIL3/SC3 platform with application software developed in accordance with the SDOs provides reasonable assurance that this aspect of the HSSSR system will not be a source of software CCF. An HSSSR system D3 analysis is still required to assess the system as a whole and identify other CCF vulnerabilities. Examples of other CCF vulnerabilities include the use of data communications and any external hazards that could impact multiple redundancies of the HSSSR system.

14 References

1. 10 CFR Part 50, Appendix A, “General Design Criteria for Nuclear Power Plants”
2. 10 CFR Part 50, Appendix B, “Quality Assurance Criteria for Nuclear Power Plants and Fuel Reprocessing Plants”
3. IEC 61508, Edition 2.0, 2010-04, Functional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems
4. EPRI Report 30020011817, Final Report Dated July 2019; Safety Integrity Level (SIL) Certification Efficacy for Nuclear Power
5. EPRI 2018 Technical Report 3002011816, Digital Engineering Guide, Decision Making Using Systems Engineering
6. IEC 61513, Edition 2.0, 2011-08, Nuclear Power Plants - Instrumentation and Control Important to Safety - General Requirements for Systems
7. STPA Handbook, Nancy G. Leveson and John P. Thomas, March 2018
8. ISO/PAS 21448:2019, Road Vehicles - Safety of The Intended Functionality”, International Organization for Standardization
9. EPRI Report 1019183, Effects of Digital Instrumentation and Control Defense-in-Depth and Diversity on Risk in Nuclear Power Plants
10. EPRI Report 3002005326, Methods for Assuring Safety and Dependability when Applying Digital Instrumentation and Control Systems

© NEI 2021. All rights reserved.

Appendix A: Connection Between Software CCF First Principles and NRC Regulatory Framework

This Appendix describes the relationship between the first principles of protection against software CCF and the NRC regulatory framework. The first principles are defined in detail in Section 6. For each first principle below the associated NRC regulatory requirements are identified.

Note that the regulations listed below may not necessarily apply to all applicants and licensees. The applicability of the regulatory requirements is determined by the plant-specific licensing basis and any proposed changes to the licensing basis associated with the proposed DI&C system under evaluation.

1. Software quality depends on complete and correct requirements, design, review, implementation, and testing

a. Design quality depends on complete and correct requirements

The following regulatory requirements are related to complete and correct requirements for HSSSR systems:

- 10 CFR 50.54(jj), 10 CFR 50.55a(h) - IEEE 603-1991 or IEEE 279 -1971 as incorporated by reference requires, in part, that components and modules shall be designed, manufactured, inspected, installed, tested, operated, and maintained in accordance with a prescribed quality assurance program.

  SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet these regulatory criteria. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

- 10 CFR Part 50, Appendix A “General Design Criteria (GDC)”

  – GDC 1, “Quality Standards and Records” - states, in part, that “Structures, systems, and components important to safety shall be designed, fabricated, erected, and tested to quality standards commensurate with the importance of the safety functions to be performed.”

  Since HSSSR systems are considered of high significance regarding the importance of safety functions to be performed, this GDC applies. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

  GDC 1 also states, in part, “Where generally recognized codes and standards are used, they shall be identified and evaluated to determine their applicability, adequacy, and sufficiency and shall be supplemented or modified as necessary to assure a quality product in keeping with the required safety function.”

© NEI 2021. All rights reserved.

NEI 20-07 credits the SIL 3 certification of platforms based on IEC 61508 and establishes SDOs synthesized from IEC 61508-3 for application software to support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture. IEC 61508 is a recognized standard by non-nuclear safety related industries (e.g., oil & gas industry and petrochemical processing industry).

GDC 1 also states, in part, "A quality assurance program shall be established and implemented in order to provide adequate assurance that these structures, systems, and components will satisfactorily perform their safety functions. Appropriate records of the design, fabrication, erection, and testing of structures, systems, and components important to safety shall be maintained by or under the control of the nuclear power unit licensee throughout the life of the unit."

SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

– GDC 13, "Instrumentation and Control" states, "Instrumentation shall be provided to monitor variables and systems over their anticipated ranges for normal operation, for anticipated operational occurrences, and for accident conditions as appropriate to assure adequate safety, including those variables and systems that can affect the fission process, the integrity of the reactor core, the reactor coolant pressure boundary, and the containment and its associated systems. Appropriate controls shall be provided to maintain these variables and systems within prescribed operating ranges."

  The HSSSR system requirements development needs to address the functional requirements stated in this GDC. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

– GDC 19, "Control Room" states, in part, "Equipment at appropriate locations outside the control room shall be provided (1) with a design capability for prompt hot shutdown of the reactor, including necessary instrumentation and controls to maintain the unit in a safe condition during hot shutdown, and (2) with a potential capability for subsequent cold shutdown of the reactor through the use of suitable procedures."

  The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. The HSSSR system requirements development needs to address the functional requirements stated in this GDC. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall

© NEI 2021. All rights reserved.

safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

– GDC 20, “Protection System Functions” states, “The protection system shall be designed (1) to initiate automatically the operation of appropriate systems including the reactivity control systems, to assure that specified acceptable fuel design limits are not exceeded as a result of anticipated operational occurrences and (2) to sense accident conditions and to initiate the operation of systems and components important to safety.”

  The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. The HSSSR system requirements development needs to address the functional requirements stated in this GDC. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

– GDC 21, “Protection System Reliability and Testability” states, “The protection system shall be designed for high functional reliability and inservice testability commensurate with the safety functions to be performed. Redundancy and independence designed into the protection system shall be sufficient to assure that (1) no single failure results in loss of the protection function and (2) removal from service of any component or channel does not result in loss of the required minimum redundancy unless the acceptable reliability of operation of the protection system can be otherwise demonstrated. The protection system shall be designed to permit periodic testing of its functioning when the reactor is in operation, including a capability to test channels independently to determine failures and losses of redundancy that may have occurred.”

  The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. The HSSSR system requirements development needs to address the design requirements stated in this GDC. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

– GDC 25, “Protection System Requirements for Reactivity Control Malfunctions” states, “The protection system shall be designed to assure that specified acceptable fuel design limits are not exceeded for any single malfunction of the reactivity control systems, such as accidental withdrawal (not ejection or dropout) of control rods.”

  The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. The HSSSR system requirements development needs to address the functional requirements stated in this GDC. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application

© NEI 2021. All rights reserved.

of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

– GDC 28, "Reactivity Limits" states, "The reactivity control systems shall be designed to have a combined capability, in conjunction with poison addition by the emergency core cooling system, of reliably controlling reactivity changes to assure that under postulated accident conditions and with appropriate margin for stuck rods the capability to cool the core is maintained."

  The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. The HSSSR system requirements development needs to address the functional requirements stated in this GDC. SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this GDC. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

- 10 CFR Part 50, Appendix B "Quality Assurance"

  – Criterion III, "Design Control" states, in part, "Measures shall be established to assure that applicable regulatory requirements and the design basis, as defined in § 50.2 and as specified in the license application, for those structures, systems, and components to which this appendix applies are correctly translated into specifications, drawings, procedures, and instructions. These measures shall include provisions to assure that appropriate quality standards are specified and included in design documents and that deviations from such standards are controlled. Measures shall also be established for the selection and review for suitability of application of materials, parts, equipment, and processes that are essential to the safety-related functions of the structures, systems and components."

    SDOs synthesized from IEC 61508-3 for application software for complete and correct requirements (requirements quality) in NEI 20-07 augment the current set of NRC regulatory guidance for requirements development to meet this criterion. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

b. Implementation and testing quality depend on design quality and process discipline

The following regulatory requirements are related to design quality and process discipline, the elements that implementation and testing quality depend on:

- 10 CFR Part 50, Appendix A "General Design Criteria (GDC)"

  – GDC 1, "Quality Standards and Records": Similar to the disposition of this GDC for complete and correct requirements, the SDOs in Section 10.6 – 10.11 synthesized from IEC 61508-3 for application software address the design quality and process discipline for implementation and testing. These SDOs augment the current set of NRC regulatory guidance for design

© NEI 2021. All rights reserved.

quality and process discipline for implementation and testing to meet this criterion. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

- 10 CFR Appendix B, "Quality Assurance Criteria"
  - Criterion I "Organization": From a holistic perspective the quality requirements imposed on the HSSR system including this criterion from 10 CFR 50 Appendix B plus the requirements from NEI 20-07 support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.
  - Criterion II "Quality Assurance Program": The quality requirements imposed on the HSSR system including this criterion from 10 CFR 50 Appendix B plus the requirements from NEI 20-07 support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.
  - Criterion III "Design Control": Similar to the disposition of this criterion for complete and correct requirements, SDOs synthesized from IEC 61508-3 for application software for implementation and test quality (10.6 – 10.11) in NEI 20-07 augment the current set of NRC regulatory guidance for HSSSR system design quality and process discipline for implementation and testing to meet this criterion. This and the application of IEC 61508 criteria on the platform hosting the application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

2. Concurrent triggering conditions are required to activate a latent defect

a. A common defect depends on the quality and commonality of the software

Quality of the software is addressed in first principle 1. The following regulatory criteria relate to commonality of software for HSSSR systems:

- GDC 24, "Separation of Protection and Control" states, "The protection system shall be separated from control systems to the extent that failure of any single control system component or channel, or failure or removal from service of any single protection system component or channel which is common to the control and protection systems leaves intact a system satisfying all reliability, redundancy, and independence requirements of the protection system. Interconnection of the protection and control systems shall be limited so as to assure that safety is not significantly impaired."

  SDO 10.1.3.2, SDOs in 10.12.3, and Section 11 address the necessity to evaluate HSSSR system hazards at the system architecture level keeping in mind the current defense in depth strategy of the nuclear power plant that addresses this GDC. Any hazards identified that jeopardize the plant's defense in depth design would require control methods to eliminate those hazards.

- 10 CFR Appendix B, "Quality Assurance"

© NEI 2021. All rights reserved.

– Criterion III “Design Control”: The quality requirements imposed on the HSSR system from 10 CFR 50 Appendix B plus the requirements from NEI 20-07 to address platform and application software reliability (and conversely addressing common defects) by applying the criteria of IEC 61508, support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

b. A triggering condition depends on system conditions

The following regulatory criteria relate to a trigger condition depends on system conditions:

GDC 22, “Protective System Independence” states in part, “Design techniques, such as functional diversity or diversity in component design and principles of operation, shall be used to the extent practical to prevent loss of the protection function.”

The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. The design basis for operating nuclear plants includes functional diversity for the protective functions. SDO 10.1.3.2, SDOs in 10.12.3, and Section 11 address the necessity to evaluate HSSSR system hazards at the system architecture level. As described in Section 11, the HSSSR system hazard analysis involves analyzing the HSSSR potential systematic failures. An important aspect of the HSSSR system hazard analysis is identifying HSSSR systematic misbehaviors in the absence of any HSSSR faults and failures. Implementing the SDOs in Section 10.12.3 or control methods identified to address HSSSR systematic failures in the hazard analysis under SDO 10.1.3.2 can require diversity in component design and principles of operation as an attribute of the control method. As stated in SDO 10.1.3.2, any required control methods applied to the application software are to be included in the application software requirements.

c. A concurrent triggering condition depends on timing and commonality of system conditions

The same regulatory criteria in “b. A triggering condition depends on system conditions” apply to this CCF first principle.

3. The effects of a software CCF can be reduced by design

a. The plant systems or components affected by a software CCF can be limited by design

The following regulatory criteria relate to a software CCF can be reduced by design:

GDC 23, “Protective System Failure Modes” states, “The protection system shall be designed to fail into a safe state or into a state demonstrated to be acceptable on some other defined basis if conditions such as disconnection of the system, loss of energy (e.g., electric power, instrument air), or postulated adverse environments (e.g., extreme heat or cold, fire, pressure, steam, water, and radiation) are experienced.”

The scope of NEI 20-07 is HSSSR DI&C systems and these systems need to meet this GDC. SDO 10.1.3.2, SDOs in 10.12.3, and Section 11 augment NRC regulatory guidance to address this GDC, by identifying the necessity to evaluate HSSSR system hazards at the system architecture level. As stated in SDO 10.1.3.2, any required control methods applied to the application software are to be included in the application software requirements.

© NEI 2021. All rights reserved.

Limiting the impact of a software CCF on plant systems or components must meet the regulatory requirements in:

GDC 24, “Separation of Protection and Control”: SDO 10.1.3.2, SDOs in 10.12.3, and Section 11 address the necessity to evaluate HSSSR system hazards at the system architecture level keeping in mind the current defense in depth strategy of the nuclear power plant that addresses this GDC. Any hazards identified that jeopardize the plant’s defense in depth design would require control methods to eliminate those hazards. As stated in SDO 10.1.3.2, any required control methods applied to the application software are to be included in the application software requirements.

b. An I&C system can be designed to force a preferred state in the event of a software CCF

See the disposition of GDC 23 in 3a.

c. Detection of an event or condition due to a software CCF provides an opportunity for response and recovery

See the disposition of GDC 23 in 3a.

4. Operating history can provide evidence of software quality

The following regulatory criteria related to operating history can be used as evidence of software quality.

10 CFR 21 states, in part, that basic components are “items designed and manufactured under a quality assurance program complying with appendix B to part 50 of this chapter, or commercial grade items which have successfully completed the dedication process.” It also states, “This assurance is achieved by identifying the critical characteristics of the item and verifying their acceptability by inspections, tests, or analyses performed by the purchaser or third-party dedicating entity after delivery, supplemented as necessary by one or more of the following: commercial grade surveys; product inspections or witness at holdpoints at the manufacturer's facility, and analysis of historical records for acceptable performance.”

One of the methods for commercial dedication is analyzing historical records for acceptable performance. The platform used for a DI&C HSSSR system, if not developed under a 10 CFR 50 Appendix B quality assurance program, must go through a commercial grade dedication process. The requirement for platform SIL3 certification augments the NRC regulatory guidance to address 10 CFR 21 commercial grade dedication. This and the application of SDOs for the hosted application software support an overall safety case to adequately address CCF sourced from either the platform or its hosted application software in the HSSSR DI&C system architecture.

© NEI 2021. All rights reserved.

Appendix B: Assurance Case Development
(Informative)

The assurance case structure provided in this appendix was adopted from IEEE 15016-2. The assurance case starts with a top-level claim for the system and uses a structured argument and evidence to support the claim. Through multiple levels of subordinate claims, the structured argument connects the top-level claim to the evidence.

The assurance case is constructed by connecting key elements, which include:

- *Claims* which are assertions about a property of the system. Claims that are asserted as true without justification become assumptions and claims supporting the argument are called sub-claims.

- *Arguments* which link the evidence to the claim, which can be deterministic, probabilistic or qualitative.

- *Evidence* which provides the basis for the justification of the claim. Some sources of evidence may include the design, the development process, testing, and inspections.

A simplified diagram of an assurance case is shown in Figure B-1.

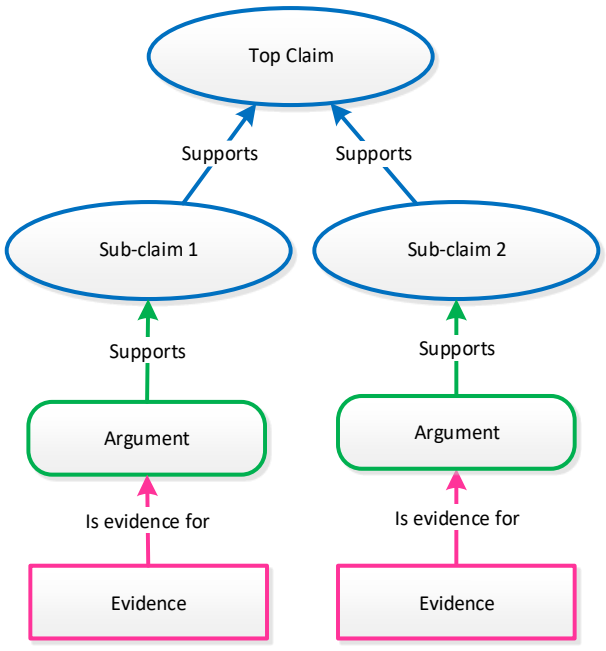


Figure B-1 - Simplified Assurance Case Structure

© NEI 2021. All rights reserved.

B.1 Assurance Case Claim Structure to Ensure Software CCF is Adequately Addressed

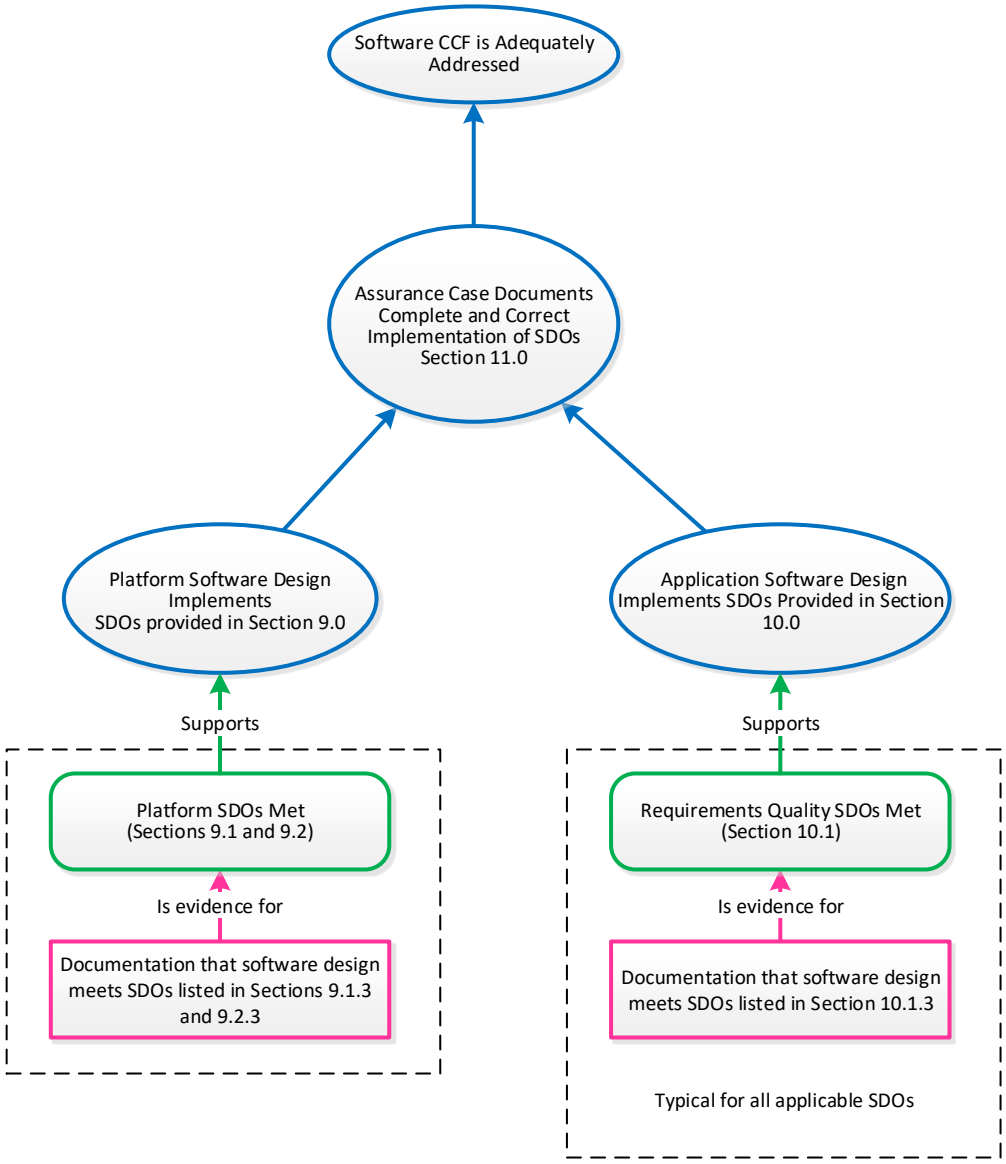


Figure B-2

© NEI 2021. All rights reserved.