INL/EXT-21-64039

Light Water Reactor Sustainability Program

Quantitative Risk Analysis of High Safety-significant Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants using IRADIC Technology

August 2021

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-21-64039

Light Water Reactor Sustainability Program

Quantitative Risk Analysis of High Safety-significant Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants using IRADIC Technology

Han Bao[1], Tate Shorthill[2], Edward Chen[3], Hongbin Zhang[1]

August 2021

[1]Idaho National Laboratory
Idaho Falls, Idaho 83415

[2]University of Pittsburgh
Pittsburgh, PA 152601

[3]North Carolina State University
Raleigh, NC 27695

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

EXECUTIVE SUMMARY

This report documents the activities performed by Idaho National Laboratory (INL) during Fiscal Year (FY) 2021 for the U.S. Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program, Risk Informed Systems Analysis (RISA) Pathway, digital instrumentation and control (DI&C) risk assessment project. In FY-2019, the RISA Pathway initiated a project to develop a risk assessment strategy for delivering a strong technical basis to support effective, licensable, and secure DI&C technologies for digital upgrades/designs. An integrated risk assessment technology for the DI&C systems (IRADIC technology) was proposed for this strategy, which aims to (1) provide a best-estimate, risk-informed capability to quantitatively and accurately estimate the safety margin obtained from plant modernization, especially for the High Safety-significant Safety-related (HSSSR) DI&C systems, (2) develop an advanced risk assessment technology to support transition from analog to DI&C technologies for nuclear industry, (3) assure the long-term safety and reliability of vital HSSSR DI&C systems, (4) reduce uncertainty in costs and support integration of DI&C systems in the plant.

To achieve these technical goals and deal with the expensive licensing justifications from regulatory insights, the IRADIC technology is instructive for nuclear vendors and utilities on how to effectively lower the costs associated with digital compliance and speed industry advances by: (1) defining an integrated risk-informed analysis process for DI&C upgrade, including hazard analysis, reliability analysis, and consequence analysis, (2) applying systematic and risk-informed tools to address common cause failures (CCFs) and quantify responding failure probabilities for DI&C technologies, particularly software CCFs, (3) evaluating the impact of digital failures at the individual level, system level, and plant level, (4) providing insights and suggestions on designs to manage the risks, thus to support the development, licensing, and deployment of advanced DI&C technologies on nuclear power plant (NPPs).

Many efforts from regulatory, industrial, and academic communities have been made for qualitatively addressing CCFs in DI&C systems, especially software CCFs, given the increased designs and deployment of HSSSR DI&C systems in NPPs. These efforts provide a technical basis for dealing with potential software CCF in the HSSSR DI&C systems of NPPs; however, some technical challenges remain:

(1) Is qualitative evaluation sufficient for addressing software CCFs in HSSSR DI&C systems?

(2) How to quantitatively evaluate CCF-related impacts to HSSSR DI&C systems and entire plant response?

(3) How to efficiently identify the most significant CCFs, especially software CCFs?

(4) How to perform a complete reliability analysis for large-scale HSSSR DI&C systems with small-scale software/digital units?

(5) Lastly, a need clearly exists to develop a risk assessment strategy to support quantitative defense-in-depth and diversity (D3) analyses for assuring the long-term safety and reliability of vital digital systems and reducing uncertainties in costs, time, and support the integration of digital systems during the plant's design stage.

To deal with the technical issues in addressing potential software CCF in HSSSR DI&C systems of NPPs, the IRADIC technology provides:

(1) An integrated and best-estimate, risk-informed capability to address new technical digital issues quantitatively, accurately, and efficiently in plan modernization progress, such as software CCFs in HSSSR DI&C systems of NPPs.

(2) A common and modularized platform for DI&C designers, software developers, plant engineers and risk analysts to efficiently prevent and mitigate risk by identifying crucial failure modes and system

vulnerabilities, quantifying DI&C system reliability, and evaluating the consequences of digital failures on the plant responses.

(3) A technical basis and risk-informed insights to assist Nuclear Regulatory Commission (NRC) and industry in formalizing relevant licensing processes relevant to CCF issues in HSSSR DI&C systems.

(4) An integrated risk-informed tool for vendors and utilities to meet the regulatory requirements and optimize the D3 applications in the design of HSSSR DI&C systems.

In this report, an approach for performing software CCF analysis, given limited data, is developed and demonstrated using a case study of a highly redundant digital reactor trip system. Consequence analysis is also performed based on different accident scenarios. Results indicate plant modernization including the improvement of HSSSR DI&C systems will make great benefits to plant safety by providing more safety margins to accident management. In addition, a novel approach is proposed in this report for the quantification of software hazards when sufficient operational and testing data available. The method incorporates software development quality as well as strong analysis techniques to identify and link software defects to potential failure modes. The approach includes both semantic and test-based analysis to detect failures that can exist in different stages of the software development life cycle. This method is applied to an advanced human-system interface relevant to reactor trip safety developed from the design of Advanced Power Reactor 1400 MW (APR-1400).

ACKNOWLEDGEMENT

The research activities and achievements documented in this report were funded by the United States Department of Energy's Light Water Reactor Sustainability Program, Risk Informed Systems Analysis (RISA) Pathway. The authors are grateful to Alison (Krager) Hahn of the United States Department of Energy, and Bruce P. Hallbert, Svetlana (Lana) Lawrence and Curtis L. Smith at Idaho National Laboratory for supporting this effort. The authors thank Kenneth D. Thomas at Idaho National Laboratory and Edward Quinn at Technology Resources for their informative technical suggestions. The authors would like to recognize the technical supports from Zhegang Ma, Ronald L. Boring, Sai Zhang, Thomas A. Ulrich, Jeffrey C. Joe at Idaho National Laboratory. The authors also thank Rebecca N. Ritter at Idaho National Laboratory for technical editing and formatting of this report.

The authors would also like to acknowledge our collaborators from universities: Nam Dinh at North Carolina State University, Heng Ban at University of Pittsburgh, Hyun Gook Kang at Rensselaer Polytechnic Institute for their valuable comments in methodology development and demonstration.

CONTENTS

FIGURES

TABLES

ACRONYMS

AFP	auxiliary feedwater pump
AFW	auxiliary feedwater
APR-1400	Advanced Power Reactor 1400 MW
ATF	accident tolerant fuel
ATWS	anticipated transient without scram
BAHAMAS	Bayesian and HRA-Aided Method for the Reliability Analysis of Software
BBN	Bayesian Belief Network
BFM	beta-factor model
BP	bistable processor
CCCG	common cause component group
CCF	common cause failure
CDF	core damage frequency
CET	core exit thermocouple
CF	coverage factor
CIM	component interface module
CPU	central processing unit
D3	defense-in-depth and diversity
DD	defect density
DiD	defense-in-depth
DI&C	digital instrumentation and control
DIS	diverse indication system
DOE	U.S. Department of Energy
DOE–NE	U.S. Department of Energy–Office of Nuclear Energy
DOM	digital output module
DPS	diverse protection system
EPRI	Electric Power Research Institute
ERP	Enhanced Resilient Plant
ESF	engineered safety feature
ESFAS	Engineered Safety Features Actuation System
ET	event tree
ETA	event tree analysis
FLEX	Flexible Coping Strategy
FMEA	failure mode effect analysis

FT	fault tree
FTA	fault tree analysis
FY	Fiscal Year
GC	group-controller
HAZCADS	Hazards and Consequence Analysis for Digital Systems
HJTC	heated-junction thermocouple
HPI	high-pressure injection
HRA	human reliability analysis
HSSSR	High Safety-significant Safety-related
IAP	integrated action plan
ICC	inadequate core cooling
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INL	Idaho National Laboratory
IPS	information processing system
IRADIC	Integrated risk assessment for digital instrumentation and control systems
KLOC	thousand lines of code
LC	loop-controller
LCL	local coincidence logic
LOCA	loss-of-coolant accident
LOSC	loss of seal cooling
LP	logic processor
LPCI	low-pressure core injection
LPI	low-pressure injection
LWR	light water reactor
LWRS	Light Water Reactor Sustainability
MCR	main control room
MDAFP	motor-driven auxiliary feedwater pumps
MDP	motor-driven pump
MGL	multiple Greek letter model
NEI	Nuclear Energy Institute
NHPP	non-homogeneous Poisson process
NPP	nuclear power plant
NRC	U.S. Nuclear Regulatory Commission
ODC	orthogonal defect classification

ORCAS	Orthogonal-defect Classification for Assessing Software reliability
PBF	partial beta factor
PWR	pressurized water reactor
PRA	probabilistic risk assessment
QIAS-P	qualified indication and alarm system – safety
QIAS-N	qualified indication and alarm system – non-safety
R&D	research and development
RCCA	rod cluster control assembly
RCS	reactor control system
RePS	Reliability Prediction System
RESHA	Redundancy-guided Systems-theoretic Hazard Analysis
RHR	residual heat removal
RISA	Risk-Informed Systems Analysis
RMS	radiation monitoring system
RPS	reactor protection system
RRO	Risk Reduction Objective
RSR	reserve shutdown room
RT	requirements traceability
RTB	reactor trip breaker
RTS	reactor trip system
RTSS	reactor trip switchgear system
SDLC	software development life cycle
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SBO	station black-out
SDD	software design description
SDN	safety data network
SP	selective processor
SRGM	software reliability growth method
SRP	Standard Review Plan
SRS	software requirements specifications
SSC	system, structure, and component
ST	shunt
STD	systems test document
STPA	systems-theoretic process analysis
TC	test coverage

TDAFP	turbine-driven AFP
THERP	Technique for Human Error Rate Prediction
UCA	unsafe control action
UIF	unsafe information flow
UPM	Unified Partial Method
U.S.	United States
UV	undervoltage

QUANTITATIVE RISK ANALYSIS OF HIGH SAFETY-SIGNIFICANT SAFETY-RELATED DIGITAL INSTRUMENTATION AND CONTROL SYSTEMS IN NUCLEAR POWER PLANTS USING IRADIC TECHNOLOGY

1. INTRODUCTION

1.1 INL-IRADIC Technology

This report documents the activities performed by Idaho National Laboratory (INL) during fiscal year (FY) 2021 for the U.S. Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program, Risk Informed Systems Analysis (RISA) Pathway, digital instrumentation and control (DI&C) risk assessment project [1] [2]. The LWRS program, sponsored by the U.S. DOE and coordinated through a variety of mechanisms and interactions with industry, vendors, suppliers, regulatory agencies, and other industry research and development (R&D) organizations, conducts research to develop technologies and other solutions to improve economics and reliability, sustain safety, and extend the operation of nation's fleet of nuclear power plants (NPPs). The LWRS program has two objectives to maintain the long-term operations of the existing fleet: (1) to provide science- and technology-based solutions to industry to implement technology to exceed the performance of the current business model and (2) to manage the aging of systems, structures, and components (SSCs) so NPP lifetimes can be extended, and the plants can continue to operate safely, efficiently, and economically.

As one of the LWRS program's R&D pathways, RISA Pathway aims to support decision-making related to economics, reliability, and safety providing integrated plant systems analysis solutions through collaborative demonstrations to enhance economic competitiveness of the operating fleet. The RISA Pathway R&D's purpose is to support plant owner-operator decisions with the aim to improve the economics and reliability and maintain the high levels of current NPPs' safety over periods of extended plant operations. The goal of the RISA Pathway is to conduct R&D to optimize safety margins and minimize uncertainties to achieve economic efficiencies while maintaining high levels of safety. This is accomplished in two ways: (1) by providing scientific basis to better represent safety margins and factors that contribute to cost and safety; and (2) by developing new technologies that reduce operating costs.

In FY 2019, the RISA Pathway initiated a project to develop a risk assessment strategy for delivering a strong technical basis to support effective, licensable, and secure DI&C technologies for digital upgrades/designs [1]. An integrated risk assessment technology for the DI&C systems (IRADIC technology) was proposed for this strategy which aims to:

- Provide a best-estimate, risk-informed capability to quantitatively and accurately estimate the safety margin obtained from plant modernization, especially for the high safety-significant safety-related (HSSSR) DI&C systems
- Develop an advanced risk assessment technology to support transition from analog to DI&C technologies for nuclear industry
- Assure the long-term safety and reliability of vital HSSSR DI&C systems
- Reduce uncertainty in costs and support integration of DI&C systems in the plant.

To achieve these technical goals and deal with the expensive licensing justifications from regulatory insights, the IRADIC technology is instructive for nuclear vendors and utilities to effectively lower the costs associated with digital compliance and speed industry advances by:

(1) Defining an integrated risk-informed analysis process for DI&C upgrade, including hazard analysis, reliability analysis, and consequence analysis

(2) Applying systematic and risk-informed tools to address common cause failures (CCFs) and quantify corresponding failure probabilities for DI&C technologies, particularly software CCFs

(3) Evaluating the impact of digital failures at the component level, system level, and plant level

(4) Providing insights and suggestions on designs to manage the risks, thus, to support the development, licensing, and deployment of advanced DI&C technologies on NPPs.

The IRADIC technology includes two phases: risk analysis and risk evaluation. Risk analysis aims to identify hazards of digital-based SSCs, estimate their failure probabilities, and analyze relevant consequences by performing hazard analysis, reliability analysis, and consequence analysis. The results from the risk analysis are compared with the specific risk acceptance criteria in the risk evaluation. Figure 1 displays the schematic of the risk assessment strategy for DI&C systems. The task is to evaluate whether the risk from digital failures can be accepted at the component, system, and plant levels, then provide suggestions to reduce these potential risks in both design stage and operation and maintenance stage.

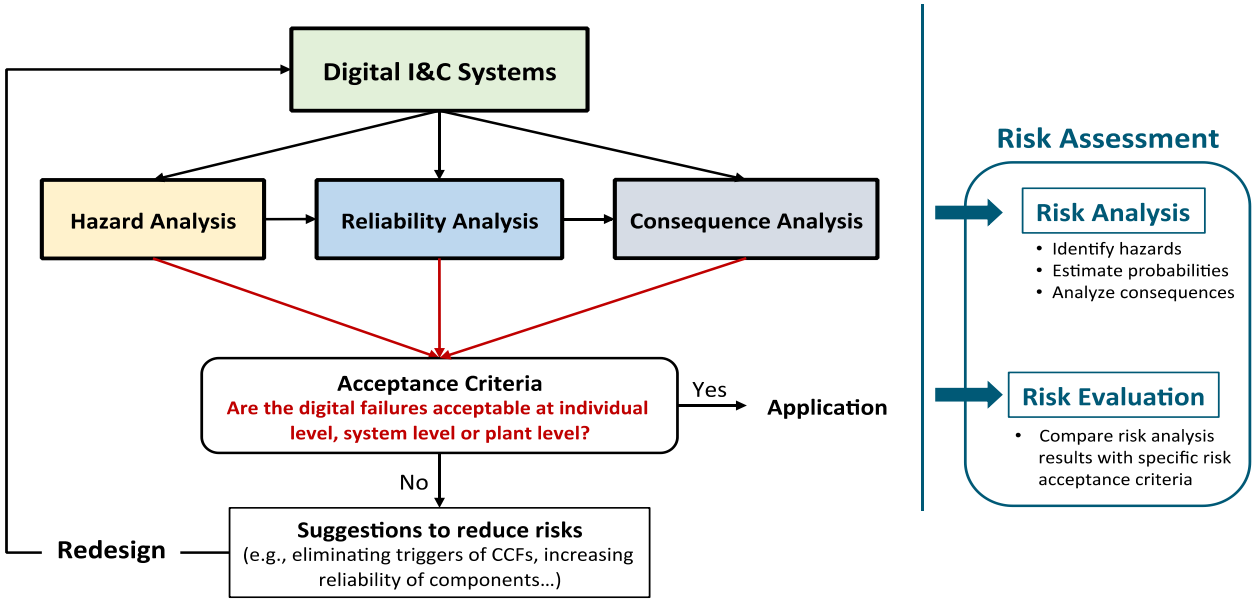

Figure 1. Schematic of proposed risk assessment strategy for DI&C systems.

1.2 Challenges in Addressing CCF in HSSSR DI&C Systems

The system aspects of DI&C involve issues that extend beyond individual components and even beyond the function of the system itself. The challenge with using these system aspects is discussed in NUREG/CR-6901. Digital systems exhibit two types of interactions—Type 1: the interactions of a DI&C system (and/or its components) with a controlled process (e.g., NPP); and Type 2: the interactions of a DI&C system (and/or its components) with itself and/or other digital systems and components [3]. Kirschenbaum et al. provide a useful summary of these concerns in their own work on the investigation of digital systems [4]. Common or redundant components are often utilized as a backup to ensure system reliability. However, the improper application of redundant features can leave a system vulnerable to CCFs, which arise from the malfunction of two or more components, or functions, due to a single failure source [5] [6]. To make redundancy designs effective, diversity is employed, providing an alternative technology, method, technique, or means to achieve a desired result [7]. The diverse protection helps to eliminate the common features necessary for a CCF. Some general observations on the consistencies and

inconsistencies in how defense-in-depth (DiD) has been defined and used were included in NUREG/KM-0009, "Historical Review and Observations of Defense-in-Depth" [8]. In 2016, the U.S. Nuclear Regulatory Commission (NRC) revised the standard review plan (SRP) to fully adapt it and associated regulatory guides to DI&C systems [9]. Chapter 7 of the SRP provided guidance for the review of the I&C portions of: (1) applications for nuclear reactor licenses or permits and (2) amendments to existing licenses.

DiD and diversity (D3) analyses are proposed and performed using deterministic approaches while the NRC probabilistic risk assessment (PRA) policy statement encourages the use of risk information in all regulatory activities supported by the state-of-the-art and data [10]. Activities to develop digital system models have been in process for some time; however, no approaches have been generally accepted for digital system modeling in current NPP PRA efforts. Currently, the NRC continues to perform research that supports the development of licensing criteria to evaluate new DI&C systems. According to guiding principles in SECY-18-0090 [11], published in 2018, a D3 analysis for reactor trip systems (RTSs), and engineered safety features should be performed to demonstrate that vulnerabilities to a CCF have been identified and adequately addressed, either by a design-basis deterministic approach or best-estimate approach. Recently, in January 2019, NRC staff developed the integrated action plan (IAP) [12], and it updates the plan as a living document. One goal of the IAP is to assist NRC staff in performing regulatory reviews and I&C system inspections in more-efficient, effective, consistent, and risk-informed ways. In addition, industry is seeking a more risk-informed, consequence-based regulatory infrastructure that removes uncertainty in requirements and enables technical consistency [12].

Many efforts from regulatory, industrial, and academic communities have been made for qualitatively addressing CCFs in DI&C Systems, especially software CCFs, given the increased designs and deployment of HSSSR DI&C systems in NPPs. To successfully model DI&C systems, the need exists to model both the hardware and software interactions of the system. Traditional methods, such as failure modes and effects analysis (FMEA) and fault tree analysis (FTA), have been used to extensively model analog systems. However, interactions between digital systems and controlled processes (i.e., Type 1 interactions) and the interactions between digital systems and their own components or other systems (i.e., Type 2 interactions) can result in failure modes or hazards that are difficult to discover using traditional methods [13]. Lessons learned from the NRC's investigation of multiple analysis methods indicate there "may not be one preferable method for modeling all digital systems" [13]. Combining methods may prove beneficial. A recent advancement in hazards analysis, developed jointly by Electric Power Research Institute (EPRI) and Sandia National Laboratory, combines FTA and the systems-theoretic process analysis (STPA) as a portion of their methodology for Hazard and Consequence Analysis for Digital Systems (HAZCADS) [14]. Though STPA may be applied at any stage of system design and review, it is ideally suited for early application in the design process before safety features have been incorporated into the design [15]. Then, as more details are incorporated, the STPA method is applied iteratively to further improve the design. However, even when fine detail about a system is known, the analysis may remain at a high level, relying on causal factor investigations to provide the detail of subcomponent failures and interactions. In other words, details such as redundant subsystems or components are often ignored in all but the final part of STPA.

In July 2021, Nuclear Energy Institute (NEI) published the pre-submittal activities for NEI 20-07, "Guidance for Addressing Software Common Cause Failure in High Safety-Significant Safety-Related Digital I&C Systems." [16] A two-step process was proposed to address HSSSR systematic CCF based on STPA: Step 1 is to perform a systematic hazards analysis based on STPA that creates a model of the system control structure, identifies unsafe control actions (UCAs) as software failures, and establish a risk reduction objective (RRO); Step 2 is to develop STPA loss scenarios and eliminate or mitigate them in an efficient way. A bounding assessment is proposed to calculate the risk change when entire HSSSR systems fail due to software CCFs (assuming system failure probability = 1). The risk change (e.g., Δ core damage frequency [CDF]) is then mapped to the regions described in RG 1.174 [17] and used to

determine the RRO. This process qualitatively addresses potential failures in DI&C based on a bounding assessment, consequently, the real safety margin gained by plant digitalization on HSSSR DI&C systems could be underestimated in this conservative way.

The described above efforts provide a technical basis for dealing with potential software CCF in the HSSSR DI&C systems of NPPs; however, some technical challenges remain:

(1) Is qualitative evaluation sufficient for addressing software CCFs in HSSSR DI&C systems? Most of the STPA-based approaches mentioned above focus on the identification of software failures but not the quantification of their probabilities. Although these software failures are added into an integrated fault tree, their probabilities are not calculated. Instead, a conservative bounding assessment is performed to evaluate their impacts to plant safety (e.g., Δ CDF), which may lead to an underestimation of safety margins gained by plant digitalization.

(2) How to quantitatively evaluate CCF-related impacts to HSSSR DI&C systems and entire plant response? This proposes a need in developing an integrated strategy to include both qualitative hazard analysis and quantitative reliability and consequence analysis for addressing software CCFs issues in HSSSR DI&C systems of NPPs. Inputs and outputs of each analysis process should be consistently connected.

(3) How to efficiently identify the most significant CCFs, especially software CCFs? Existing STPA-based approaches represents good performance in capturing systematic failures in digital interactions; however, there is not a clear representation of how to create a control structure for a complicated system containing multiple layers of redundancy and diversity.

(4) How to perform a complete reliability analysis for large-scale HSSSR DI&C systems with small-scale software/digital units? Currently, there is no consensus method for the software reliability modeling of digital systems in NPPs. A reliability analysis approach is needed, especially for the quantification of UCAs from STPA analysis.

(5) Lastly, a need clearly exists to develop a risk assessment strategy to support quantitative D3 analyses for assuring the long-term safety and reliability of vital digital systems and reducing uncertainties in costs, time, and supporting integration of digital systems during the design stage of the plant.

1.3 Value Proposition for INL-IRADIC Technology

To deal with the technical issues in addressing potential software CCF issues in HSSSR DI&C systems of NPPs, the IRADIC technology is expected to provide:

(1) An integrated and best-estimate, risk-informed capability to address new technical digital issues quantitatively, accurately, and efficiently in plan modernization progress, such as software CCFs in HSSSR DI&C systems of NPPs.

Existing qualitative approach for addressing CCFs in HSSSR DI&C systems may significantly underestimate the real safety margin introduced by plant digitalization. The IRADIC technology is developed and demonstrated in an integrated way including both qualitative hazard analysis and quantitative reliability and consequence analysis. IRADIC aims to provide a best-estimate, risk-informed capability to accurately estimate the safety margin increase obtained from plant modernization, especially for the digital HSSSR I&C systems.

In IRADIC, a redundancy-guided systems-theoretic method for hazard analysis (RESHA) was developed HSSSR DI&C systems for supporting I&C designers and engineers to address both hardware and software CCFs and qualitatively analyze their effects on system availability [18] [19]. It also provides a technical basis for implementing, following reliability and consequence analyses of unexpected software failures and supporting the optimization of D3 analyses in a cost-efficient way. Targeting the complexity of redundant designs in HSSSR DI&C systems integrates STPA [15], FTA and HAZCADS

[14] to effectively identify software CCFs by reframing STPA in a redundancy-guided way: (1) framing the complexity of the redundancy problem in a detailed representation; (2) clarifying the redundancy level using FTA before applying STPA; (3) building a redundancy-guided multilayer control structure; and (4) locating software CCFs for different levels of redundancy. This approach has been demonstrated and applied for the hazard analysis of a four-division digital RTS [18] and a four-division, digital, engineered safety features actuation system (ESFAS) [19]. These efforts have been included in the FY-20 milestone report [2].

The second part in risk analysis is reliability analysis with the tasks of: (1) quantifying the probabilities of basic events of the integrated fault tree (FT) from the hazard analysis; (2) determining the optimal basic component combinations for prevention and mitigation; and (3) estimating the probabilities of the consequences of digital system failures. In IRADIC, two methods have been developed for different application conditions: the Bayesian and HRA-Aided Method for the reliability Analysis of Software (BAHAMAS) [20] for limited-data conditions and Orthogonal-defect Classification for Assessing Software reliability (ORCAS) for data-rich analysis. More details can be found in Chapter 4.

As the third and final part, consequence analysis should be implemented to evaluate the impact of consequences from digital failures on plant responses. The main concern is some software failures have the potential to initiate an unanalyzed event or scenario that may not be analyzed before and, therefore, to threat reactor safety, such as by core damage or a large early release. The PRA results from the previous reliability analyses are supposed to provide different risk-significant accident scenarios for the multi-physics best-estimate plus uncertainty analysis. The capability has been built by different platforms such as the INL-developed LOTUS [21]. In FY 2021, a couple of accident scenarios have been selected for the consequence analysis, as described in Chapter 3.

(2) A common and modularized platform for I&C designers, software developers, plant engineers, and risk analysts to efficiently prevent and mitigate risk by identifying crucial failure modes and system vulnerabilities, quantifying DI&C system reliability, and evaluating the consequences of digital failures on the plant responses.

Many programs/projects were and are being created with various methods/approaches/frameworks generated either for single software reliability analysis or for quantifying the system-level interactions between digital systems or between digital systems and other systems. However, these efforts are rarely targeting software CCFs in HSSSR DI&C systems.

As shown in Figure 2, IRADIC, as a modularized platform, aims to have a good communication with various small-scale unit-level software reliability analysis methods (e.g., quantitative software reliability methods) and large-scale system-level reliability analysis frameworks (e.g., PRA). RESHA, as a top-down approach, can identify the digital or software failures in the unit-level interactions inside of a digital system, then BAHAMAS and ORCAS can be used to quantify the probability of the STPA-identified software failures based on suitable existing quantitative software reliability methods such as Bayesian networks, test-based, or metric-based methods.

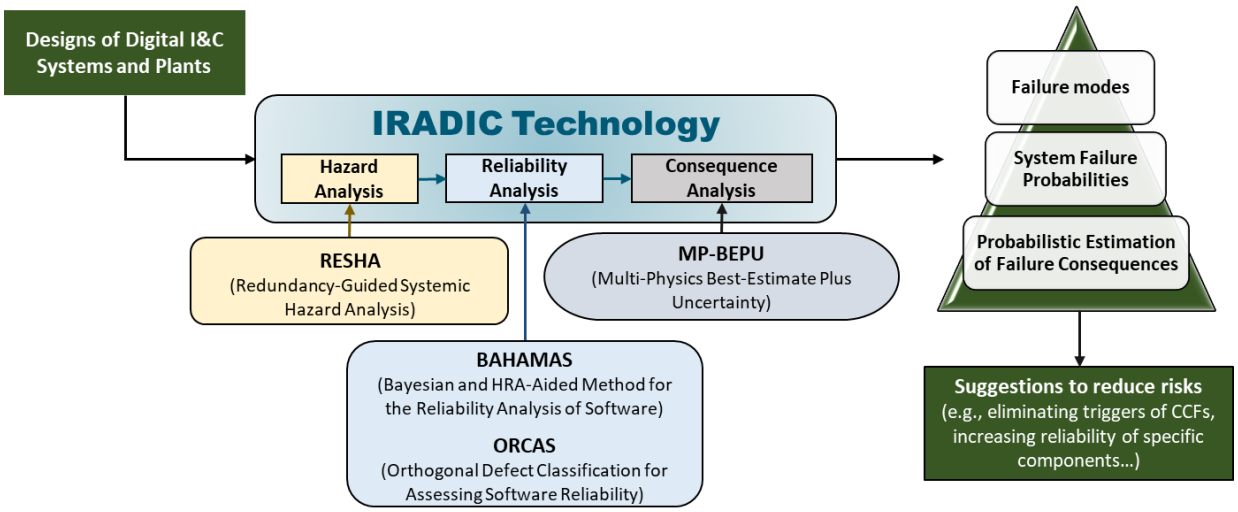

Figure 2. The flexible and modularized structure of the IRADIC technology.

(3) A technical basis and risk-informed insights to assist NRC and industry in formalizing relevant licensing processes relevant to CCF issues in HSSSR DI&C systems.

Figure 3 illustrates how the IRADIC technology can support licensing of a HSSSR DI&C design or upgrade. NRC BTP 7-19, "Guidance for Evaluation of Diversity and Defense-In-Depth in Digital Computer-Based Instrumentation and Control Systems Review Responsibilities" [22], clarifies the requirement for acceptable methods for addressing CCFs, including identifying CCFs, reducing CCF likelihood, and evaluating CCF impacts in design-basis events. The capabilities of IRADIC technology in hazard, reliability, and consequence analysis matches well with these requirements.

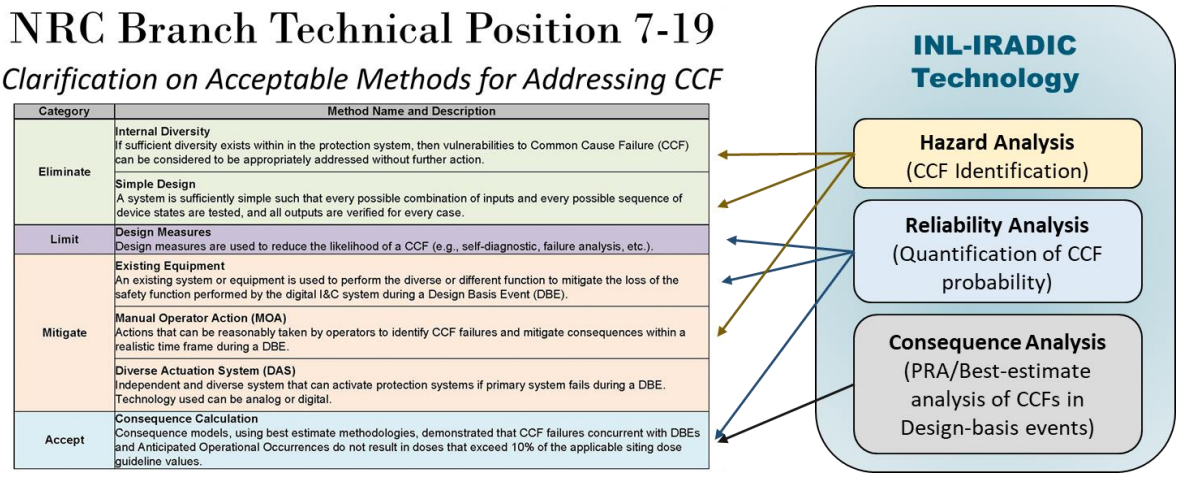

Figure 3. Clarification on acceptable methods for addressing CCF according to NRC BTP 7-19 vs. INL-IRADIC capability in CCF analysis.

(4) An integrated risk-informed tool for vendors and utilities to meet the regulatory requirements and optimize the D3 applications in the design of digital HSSSR systems.

The IRADIC technology can be beneficial for the design of digital HSSSR systems in plant modernization process: the estimated safety margin using IRADIC should be much higher and more accurate than other conservative bounding assessment approach. The safety improvements of these new digital designs are expected to be significant and can be presented more clearly.

Currently, it is thought after qualitatively addressing CCFs, all of them need to be fixed by adding diversity, which costs a lot. In fact, some of CCFs do not have significant impacts on the change of CDF or large release frequency. The IRADIC technology can evaluate the impacts of single software CCFs to the HSSSR DI&C systems and even the plant safety, based on which suggestions can be provided to optimize the D3 application in the design of HSSSR DI&C systems. By comparing the risk and cost of different redundant and diverse designs, cost can be saved if some CCFs are proved to be insignificant to plant safety. Based on current IRADIC analysis results, failure probability of HSSSR DI&C system due to software CCFs is quite low and the CDF is also significantly reduced compared with the one with traditional analog systems. More details can be found in Chapter 3.

1.4 Report Layout

Chapter 2 describes RISA efforts in software CCF modeling and estimation, considering existing CCF modeling methods were mainly developed for hardware failures, which may be inapplicable in software CCF analysis. Chapter 3 documents the consequence analysis of a generic pressurized water reactor (PWR) model with improved digital RTS and ESFAS FTs. Chapter 4 introduces the recent efforts in the improvement of IRADIC capability for software reliability analysis, including the development of ORCAS method. In Chapter 5, ORCAS is demonstrated for the risk analysis of an advanced human-system interface (HSI) relevant to reactor trip; the model of which was developed based on the APR-1400 HSI design. Chapter 6 provides conclusions and outlines future work.

2. SOFTWARE COMMON CAUSE FAILURE MODELING AND ESTIMATION

CCF modeling is an important part of a risk assessment. Previous research efforts focused on the identification and quantification of risks associated with HSSSR DI&C systems. The work described herein details the CCF modeling of the risks identified in previous efforts. Section 2.1 provides background details for CCF modeling. Section 2.2 introduces methods for modeling CCFs. Section 2.3 describes a selected approach for modeling components that are part of multiple common cause component groups (CCCG), groups of components which share coupling. Section 2.4 applies the selected CCF modeling techniques as part of a case study. Sections 2.5 and 2.6 provide the results and conclusions of this chapter.

2.1 Background of Common Cause Failure Modeling

A CCF is the occurrence of two or more failure events "simultaneously" due to a shared cause [5] [6] [23]. A CCF is the result of the existence of two main factors—a failure cause and a coupling factor (or mechanism) [24] [25] [26]. The failure cause is the condition to which failure is attributed. O'Connor indicates failure cause can be divided into a proximity cause or a root cause [27]. The proximity cause is a readily identifiable condition but may not truly represent the cause of the failure. Meanwhile, the root cause is the true source of failure which leads to the proximity cause and failure of the component [27]. The coupling factor (or coupling mechanism) creates the condition for a failure cause to affect multiple components thereby producing a CCF [24]. Example of common coupling mechanisms given in NRUEG/CR-5485:

- Design
- Hardware
- Function
- Installation
- Maintenance
- Procedures
- Interfaces
- Locations
- Environment.

The identification of coupling factors and CCCGs is an essential part of the CCF analysis. When root causes and coupling factors are known, they can be used for the explicit modeling of CCFs. However, the number and variations root causes and coupling factors can quickly become unmanageable for PRA models [28]. Implicit CCF modeling provides a way to simplify the modeling of CCFs and lead to more manageable analyses. According to [28], guesses for explicit models might be better than choosing to use implicit models exclusively. If possible, at least some of the potential CCFs should be modeled explicitly [28] while the rest may be modeled using implicit methods. The analysis of CCFs is dependent on the requirements of the analysis, available data, and the configuration components for the system of interest.

The main steps usually involved in a CCF analysis are to identify the system, identify hazards, determine CCCGs, select CCF models, define model parameters, and evaluate CCFs and their influences. Most of the preliminary steps are covered in the early portions of IRADIC. Here we will focus on the current challenges to our work, namely the quantification process starting from the identification of CCCGs.

Within a CCCG of m components, CCFs are in various combinations of k/m where $(2 \leq k \leq m)$. The components of a CCCG are generally assumed to have identical coupling factors. Table 1 provides an example system. Here, components A, B and C share installation procedure, location, and manufacturer. Assume these components are part of a system that requires 2 out of 3 (2oo3) for success of the system. Figure 4 details a portion of the FT for Example System 1.

Table 1. Coupling factors for Example System 1.

Component	**Installation Procedure**	**Location**	**Manufacturer**
A	Procedure A	Room A	Company A
B	Procedure A	Room A	Company A
C	Procedure A	Room A	Company A

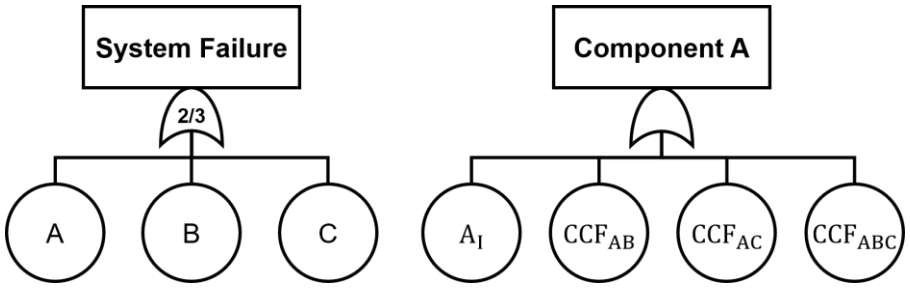

Figure 4. Example System showing the relationship of independent and dependent failures in the context of a FT.

Consider a scenario where the components in Example System 1 are arranged in the 2/3 criteria for failure shown in Figure 4. The probability of failure for the system is described in NUREG/CR-5485 [24] and is shown below:

$$\begin{aligned} P(F) = {} & P(A_I)P(B_I) + P(B_I)P(C_I) + P(A_I)P(C_I) \ldots \\ & + P(CCF_{AB}) + P(CCF_{BC}) + P(CCF_{AC}) \ldots \\ & + P(CCF_{ABC}) \end{aligned} \tag{1}$$

It is a common practice in reliability modeling to assume the failure probabilities (or rates) of similar components are the same [24]. This symmetry assumption results in the following:

$$Q_1^3 = P(A_I) = P(B_I) = P(C_I) = Q_1 \tag{2}$$

$$Q_2^3 = P(CCF_{AB}) = P(CCF_{BC}) = P(CCF_{AC}) = Q_2 \tag{3}$$

$$Q_3^3 = P(ABC) = Q_3 \tag{4}$$

Q_k^m represents the failure rate or probability of an event involving of k/m components in a CCCG of size m. Substitutions are made resulting in a simplified expression for the system failure probability.

$$P(F) = 3{Q_1}^2 + 3Q_2 + Q_3 \tag{5}$$

The symmetry assumption is most often applied to simplify the model and nearly all conventional CCF modeling techniques use it [27]. The symmetry assumption assumes all identical components with identical coupling factors can be placed in a single CCCG, as is the case for Example System 1. However,

there are cases where the symmetry assumption may not be appropriate and can make modeling difficult. Consider Example System 2 where the components share some but not all coupling factors. In this scenario, there are only two options: either ignore the differences or attempt to account for them. Table 2 shows a new scenario in which the coupling factors are not the same for each identical component. In this scenario, components A, B, and C are coupled by procedures, while A and B are coupled by location.

Table 2. Coupling factors for Example System 2.

Component	**Installation Procedure**	**Location**
A	Procedure A	Room A
B	Procedure A	Room A
C	Procedure A	Room B

Placing all identical components in one CCCG is the option most often selected because it does not cause issues with conventional CCF modeling techniques. Thus, the symmetry assumption is employed resulting a case were $P(CCF_{AB})$ is assumed approximately identical to $P(CCF_{BC})$ and $P(CCF_{AC})$. This assumption may be appropriate depending on the situation. Conversely, it may be of greater importance to account for variations in the coupling factors.

The second option is to allow components to be part of different CCCGs (components A, B, &C for CCCG1 and A&B for CCCG2). Allowing components to be part of multiple groups creates additional challenges for the analysis because conventional methods may provide two different probabilities for the same CCF event. For example, using conventional methods to determine $P(CCF_{AB})$ from CCCG1 and CCCG2 will lead to two different values for $P(CCF_{AB})$. This is because conventional methods incorporate the group size as part of their evaluation process. If CCF modeling is performed using a program such as Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) [29], having multiple CCCGs may lead to double counting of the $P(CCF_{AB})$. Ma et al. further address CCCG issues in their work from 2020. To avoid double counting, they suggest using the largest CCCG that is reasonable [30]. However, this solution requires the analyst to ignore asymmetry. A second option may be to select which of the two values for $P(CCF_{AB})$ is appropriate. Examples of components being part of multiple CCCGs do exist but require very specific conditions [30]. To be part of multiple CCCGs, each CCCG represents a unique failure mode or is part of a unique FT scenario. Conventional methods will work for these scenarios because the coupling factors and CCCGs are associated with different events (each the failure is different even though they are still CCFs). Some examples from [30]:

- Unique failure modes (rotating components vs. drive mechanism of pumps): “Redundant motor-driven auxiliary feedwater pumps (MDAFPs) A and B are in a CCCG with the same design, manufacture, capacity, and environment (CCCG size = 2), while they are in another CCCG along with the turbine-driven AFP (TDAFP) due to the similarity in function/design of the rotating components in MDAFPs and TDAFP (CCCG size = 3).”
- Unique FTs (low-pressure injection vs. residual heat removal): “Redundant motor-driven pumps (MDPs) A and B are in a CCCG with the residual heat removal (RHR) function (CCCG size = 2), while the same components of A and B are in another CCCG along with MDP C for the low-pressure core injection (LPCI) function (CCCG size = 3).”

2.2 Methods for Common Cause Failure Modeling

The examples of multiple CCCGs in the preceding section are outside of the current scope. This work explores CCF modeling where various coupling factors and associated root causes result in the same CCF event. For this case, when identical components exhibit coupling factors that are not necessarily shared by all components of the CCCG, there are two options: (1) group all identical components together in one CCCG and justify the assumption of symmetry; (2) place components into unique, coupling-factor-based

CCCGs and account for the inconsistency between the groups. The second option is a large motivation of this work as it pertains to several CCFs to be modeled in our case studies. This current work explores the second option by selecting methods which will allow for multiple CCCGs without introducing logical inconsistencies. The next sections discuss conventional approaches, some of their attributes, and their applicability for multiple CCCGs.

Once the CCCG's have been identified the next step usually is to determine how to model CCFs of the components within the CCCG. A number of CCF modeling techniques have been developed for these purposes. Some have placed these techniques into categories such as: direct assessment methods, ratio models, shock models, and other models [27].

2.2.1 Direct Assessment Methods

The primary means of explicit modeling is through direct assessment via data analysis of operations or testing experience. This method depends on available data. For example, the number of failures observed given demands, and how many failures involved two or more components. This is perhaps the simplest method for assessing CCFs. These methods are limited by their dependence on testing or performance data to determine CCFs. Data may be limited and not show any CCF occurrences. In addition, results are system specific. As performance data and test data can often be black box in nature, this method may actually provide very little information regarding how the component or systems failed. Some advantages are that method is simple, and if the direct assessment can be performed, it provides a groundwork for many other approaches.

The basic parameter model is an example of direct estimate methods. The basic parameter model groups CCFs according to the number of components that fail together. Given a CCCG of size 3 components (e.g., A, B, and C), there can be CCFs for 2/3 or 3/3 components. The model employs a symmetry assumption in which each combination of 2/3 components within the CCCG are assumed the same failure probability—for example, $P(CCF_{AB}) = P(CCF_{AB}) = P(CCF_{AB}) = Q2$. This allows for a simple model of the system failure probability when each combination of failure can be expressed with a value, such as Q2, Q3, etc. The basic parameter model is simple but limited by its assumption of symmetry; therefore, it is not directly applicable for our case study.

2.2.2 Ratio Models

Ratio models include those models which rely on the assumption that a ratio exists between total failure and dependent failures. This category includes the beta-factor model (BFM), alpha factor model (AFM), and multiple Greek letter (MGL) models [27]. Each relies on a ratio with only subtle differences in their formulations. The MGL and AFM account for combinations of CCFs within CCCGs (i.e., k/m for CCCG of size m). The BFM only considers CCFs for which all m/m components fail. Because each method in the category is very similar, there is an overlap between them. For example, the BFM is a unique case of the MGL model [24]. Also, CCFs for each combination of k/m in the AFM can be expressed in the MGL method. In fact, there are tables showing their equivalence in NUREG/CR-5485 [24]. The primary difference in the methods comes from how they are formulated. Alpha factor is designed to work with parameters that are more easily measured than compared with MGL [24]. Details for their formulations can be found in NUREG/CR-5485 among other sources. O'Conner introduced a variation of the AFM called the partial AFM the goal of which was to account for coupling factors and root causes in the assessment of CCF. The partial AFM relies on failure event data for the allocation and scoring of the model parameters [27].

Because of their similarity to each other, ratio models have the similar advantages and limitations. Data remains the limiting factor. When data is limited, the parameters must be approximated via expert elicitation or numerous assumptions [24]. When data is available, each method will have some approach for quantifying their parameters based on the available data.

2.2.3 Shock Models

According to NUREG/CR-5485, "shock is an event that occurs at a random point in time and acts on the system, i.e., all the components in the system simultaneously." The shock model considers a component may fail either due to random independent causes or by shocks that impact all components of the system at a certain frequency [24]. O'Connor indicates shock models follow a Poisson process where for each component of a CCCG the shock is a Bernoulli trial that will fail the component with some given probability. Shock models attempt to model the actual physical phenomena that lead to a CCF [27]. Additionally, O'Connor provided some of the advantages and disadvantages given in literature and based on his review [27]. Some advantages are shock models are applicable for systems with high levels of redundancy and are easy to adjust for different sized CCCGs. Shock models can be limited by their dependence on data, shock models require data to find parameters, and parameters are difficult to measure since they assume symmetry and do not account for defenses in a system's architecture that may prevent CCFs.

2.2.4 Other Models

Influence diagram/Bayesian belief networks (BBNs) have been of interest for some time for reliability modeling. Even our most recent effort relies on Bayesian techniques using BAHAMAS. The BBN depict the influences of events on each other by accounting for their conditional and independent relationships. BBNs are used largely for prediction or for diagnosis. Prediction is defined as given some prior knowledge of a root cause, the probability of an event can be determined. Diagnosis analysis proceeds in reverse; given evidence, the probabilities of potential causes can be determined [31]. BBNs can incorporate disparate information, which is advantageous for performing analyses [32]. These methods provide insight that allows consideration of defenses mechanisms of coupling factors of CCFs [33]. However, there can be significant uncertainty when dealing with limited data. BAHAMAS works in a very specified way to limit uncertainty associated with its application of BBN; however, such formulations may not be applicable to a network designed for coupling factors and defense mechanisms.

Interference models predict the failure of components by assuming random variables for strength and load [27]. The load is imposed upon a component or a group of components, and strength (or resistance) is how components react to a load. Failures occur when strength is overcome by the load [33]. The more intense a load is, the higher the probability of component failure. One example of the interference model is the common load model or the extended common load model. Some advantages of these models are they can be used to model high levels of redundancy and may be capable of modeling asymmetry by removing the assumption of identically distributed components [27] [33]. Disadvantages include complexity and the reliance on the symmetry assumption under normal configurations [33]. In the absence of data, interference models require assumed probability distributions for load and resistance intensities [27].

2.3 Selected Method for Modeling Multiple CCCGs

The review of existing methods provided a common theme. Nearly all methods are limited by an assumption of symmetry (the most notable exception being the BBN-based approaches) [27], and for those method which do rely on symmetry, allowing component to be part of multiple groups may lead to double counting. In addition, most methods are designed to incorporate some form of operational data. A major challenge for our current work is the issue of limited data which has a direct influence on all methods, without specific failure data the ratio models must be quantified using elicitation techniques. Shock models require assumptions about the shock parameters which are normally intended to model natural phenomena. Without certain design details, it would be challenging to approximate shocks correctly. The BBN and influence models, such as Zitrou's [33], are limited by the uncertainty associated with elicitation. Finally, interference models require assumptions that would be challenging to make for a system not yet fully defined. Therefore, rather than making special exceptions for conventional methods,

other options were investigated. One method happened to be specifically developed for the multiple CCCG scenario. This method from 2012 is a modified version of the BFM [34].

2.3.1 Modified Beta-Factor Model

The modified BFM from 2012 is designed specifically to allow for components to be members of multiple CCCGs [34]. The model assumes the total failure probability/rate (Q_t) of a component is the summation of independent (Q_I) and dependent failures (Q_D). Equation (7) shows the total dependent failure consists of the contribution of each CCCG failure, where each CCCG is assigned a group beta (β_w). Each group beta represents the contribution of a single CCCG to the total failure probability. Equation (11) shows the independent failure probability in terms of each CCCG beta and total failure probability.

$$Total\ failure\ probability = Q_t = [independent\ failure] + [dependent\ failures] \tag{6}$$

$$Q_D = P(CCCG_1) + P(CCCG_2) + \cdots P(CCCG_w) \tag{7}$$

$$P(CCCG_w) = (\beta_w)Q_t \tag{8}$$

$$\beta_t = \sum_{1}^{w}(\beta_w) \tag{9}$$

$$Q_D = Q_t \sum_{1}^{w}(\beta_w) \tag{10}$$

$$Q_I = (1-\beta_t)Q_t = \left[1 - \sum_{1}^{w}(\beta_w)\right] Q_t \tag{11}$$

Some advantages of this method include its ability to account for multiple CCCGs directly. It also avoids the potential for double counting that is present in conventional methods. Double counting is avoided because it is based on the beta-factor assumptions; the CCFs represent the failure of all the components with the CCCG. Using an earlier example, given CCCGs (components A, B, &C for CCCG1 and A&B for CCCG2), there will be no chance of counting $P(CCF_{AB})$ twice because $P(CCF_{AB})$ is only found for CCCG2. Additional advantages of this method are its ease of application and its ability to consider the coupling factors unique for each CCCG.

This modified BFM, like most methods, require reference data to determine each CCCG failure probability/rate. Like other ratio-based methods, the quantification of its parameters can be challenging for the limited-data scenario. The modified BFM is limited to identical components with identical total failure probabilities. If the Q_t for the components within a CCCG are not identical, then depending on the Q_t selected for Equation (8), there will be differing values for the same CCFs. An additional limitation can occur if the total beta, shown by Equation (9), exceeds unity. If this happens, then the dependent failures will exceed the total failure probability. To account for this issue, the original authors indicate a possible solution which is to normalize the CCCG beta factors such that they sum to unity while maintaining their relative magnitudes. Other options include normalizing by the largest CCCG beta or using weight factors for each CCCG [34].

Despite its known limitations, this work will employ the modified BFM for the quantification of CCFs because it works directly for the multiple CCCG scenario. The next challenge is defining the model parameters. Some approaches use direct data [24]. Others attempt expert elicitation. The emphasis of the current work is the limited-data scenario which naturally requires some form of elicitation. For elicitation,

it is desirable to consider qualitative defenses against CCFs. There are at least two methods presented in literature which express the elicitation of the beta parameter without the use or dependence on operational data. These two methods, both of which are called "partial beta methods," develop betas from a combination of partial attributes. The chief difference among them is how they find an overall beta; one employs an additive scheme, while the other a multiplicative scheme [33]. The following is a discussion of these two methods for defining an overall beta.

2.3.2 Partial Beta Factor-1

Developed around 1987, this method's goal was to provide an improvement over the traditional BFM. The claim was the dependent failure rate for a system could not be determined without an engineering assessment of that system's defenses for such failures. The resultant method provides a means of determining quantitative results informed through a structured qualitative analysis [35]. The system is assessed according to 19 defenses (see Table 3). Each defense receives a beta value (between 0–1), the product of which is the beta to be assigned for the system. This method is focused on finding a beta for use in the standard BFM. Again, this is a component failure ratio model in which the total failure rate is related to the independent and dependent failures by means of the ratio beta [27].

Table 3. Defenses for the assessment of a system beta.

Defense categories	**Partial betas**
Design control	β_1
Design review	β_2
Functional diversity	β_3
Fail-safe design	…
Operational interfaces	…
Protection & segregation	…
Redundancy & voting	…
Proven design & standardization	…
Derating & simplicity	…
Construction control	…
Testing & commissioning	…
Inspection	…
Construction standards	…
Operational control	…
Reliability monitoring	…
Maintenance	…
Proof test	…
Operations	β_{19}
$\beta = \prod \beta_i$	

- Limitations: There is very limited documentation for this method. The use of a product means there may be a tendency for the system beta to approach a small value when this may not be appropriate. For example, if 18/19 defense were given $\beta_i = .99$, there should be a high likelihood of CCF for the system. However, the remaining β can dominate the system score resulting in an improper score for the system beta (e.g., $\beta_{19} = .1$ will result with $\beta = .083$). Should additional defense categories be

needed, their product would approach a small value and may result in underpredicted dependent failures.

- Advantages: The method provides a more rigorous estimation of beta compared simple guesses. The method is simple.

2.3.3 Partial Beta Factor-2

This second method has been developed and modified by several individuals starting with R. A. Humphreys and Rolls-Royce and Associates [36] and later modified to become part of what is called the Unified Partial Method (UPM) [33]. The UPM has a systems level and a component level focus. UPM's component level approach is called the partial beta-factor method and is based on Humphreys' work, though Humphreys never called it a partial beta-factor method. In fact, the method doesn't use partial betas, rather the method uses a collection of sub-factors which contribute to the overall beta score. So, this method really should be called the beta-from-sub-factors method; however, since this is a mouthful, we will keep with continue to call this a partial beta-factor method. To keep it distinguished for remainder of the work, it will be called Partial Beta Factor-2 (PBF-2).

The PBF-2 is a method used to define a beta for use in CCF modeling. The method was founded in the question, "What attributes of a system reduce common cause failures?" [36] A collection attributes, called sub-factors, were selected which are known to contributed to the prevention of CCFs. The sub-factors are shown in Table 4. Each sub-factor was weighted by reliability engineers for their importance. The methodology requires the analyst to assign a score (A, B, C, etc.) for each sub-factor. An "E" indicates a component is well defended against CCFs (i.e., A= worst, E = best). Beta is then determined as a function of the assigned scores using Equation (12). The model was arranged such that the upper and lower limits for beta correspond with values reported in literature [36]. This is ensured by the sub-factor weighting and the denominator given in Equation (12). The beta value determined by this method is intended to be used with the BFM.

Table 4. Sub-factor estimation table [37].

Sub-factors	**A**	**A+**	**B**	**B+**	**C**	**D**	**E**
Redundancy (& Diversity)[1]	1750	875	425	213	100	25	6
Separation	2400		580		140	35	8
Understanding[2]	1750		425		100	25	6
Analysis	1750		425		100	25	6
MMI[3]	3000		720		175	40	10
Safety Culture[4]	1500		360		90	20	5
Control	1750		425		100	25	6
Tests	1200		290		70	15	4

[1]This sub-factor category was only called Similarity in Humphreys' original version [36]. Also, there was no A+ or B+ score for any category.
[2]Complexity in [36].
[3]Procedures in [36].
[4]Training in [36].

$$\beta = \frac{\sum(Sub - factors)}{50000} \tag{12}$$

The sub-factor names alone are not sufficient to describe the details for assessing each actual sub-factor; therefore, readers are advised to visit the original source material for making assessments. For

example, Table 5 provides details from Humphreys [36] and UPM [37] for defining the "Redundancy & Diversity" sub-factor.

Table 5. Redundancy and diversity sub-factor scoring criteria.

A	Minimum identical redundancy (e.g., 1oo2, 2oo3, 3oo4 for success)
	H: Identical units
A+	Enhanced identical redundancy (e.g., 1oo3, 2oo4 for success)
	H: n/a
B	Robust identical redundancy (e.g., 1oo4, 1oo5, 2oo5 etc.)
	H: Similar units with only small differences in layout or circuit
B+	Unusually high-identical redundancy (1oo ≥8)
	H: n/a
C	Enhanced identical redundancy (e.g., 1oo3) with functional diversity OR Robust identical redundancy (e.g., 1oo ≥4) with operational diversity OR Unusually high-identical redundancy (e.g., 1oo ≥8) in a passive system
	H: Different layout or circuit to achieve same purpose
D	Robust identical redundancy (1oo≥4) with functional diversity
	H: Units with different function but with common factors (e.g., a number of identical components)
E	Two entirely diverse independent redundant subsystems.
	H: Diverse units, quite different both in function and construction.
Note: "H" indicates guidance from Humphreys [36]	

- Limitations: the PBF-2 is limited by the range of possible beta it can produce. Zitrou indicates the use of expert judgment to define the weighting of the sub-factors, and the fact that the model is point based means there is inherent uncertainty for the model which remains to be captured [33]. Additionally, because the possible outcomes of beta are limited, the model may be considered inflexible.
- Advantages: The model is simple to apply and allows for a more structured determination of beta than simple judgments. The provided model allows for qualitative features to be considered in the quantification of CCFs. Unlike PBF-1, a weighting scheme prevents any particular sub-factor score from illogically and drastically offsetting the total system beta.

Despite its limitations, perhaps the most valuable aspect of PBF-2 is its ability to provide a beta value with very little system-specific detail. Other methods require specific operational data (e.g., Cause-Defense Matrices [38]) or system configuration details (e.g., Betaplus [39]). Betaplus requires information about wiring configurations and possibly testing information [39] [40]. Such detail is not always available for novel designs or for preliminary risk assessments; consequently, PBF-2 provides the best option for our requirements. Future work may lead to improvements of PBF-2 to better account for its known limitations.

This section has discussed the development of an approach for performing CCF analysis given the limited data and multiple CCCGs. The approach relies on the modified BFM to account for multiple CCCGs and PBF-2 to define beta factors for each CCCG. Together these two methods will provide a means to overcome the limitations of conventional methods. A formalized process that relies on the modified BFM and PBF-2 is shown in Figure 5. The primary sections' headings of Figure 5 come from

descriptions of CCF modeling processes found in existing sources [24] [27]. The subsequent section will demonstrate this process as part of a case study.

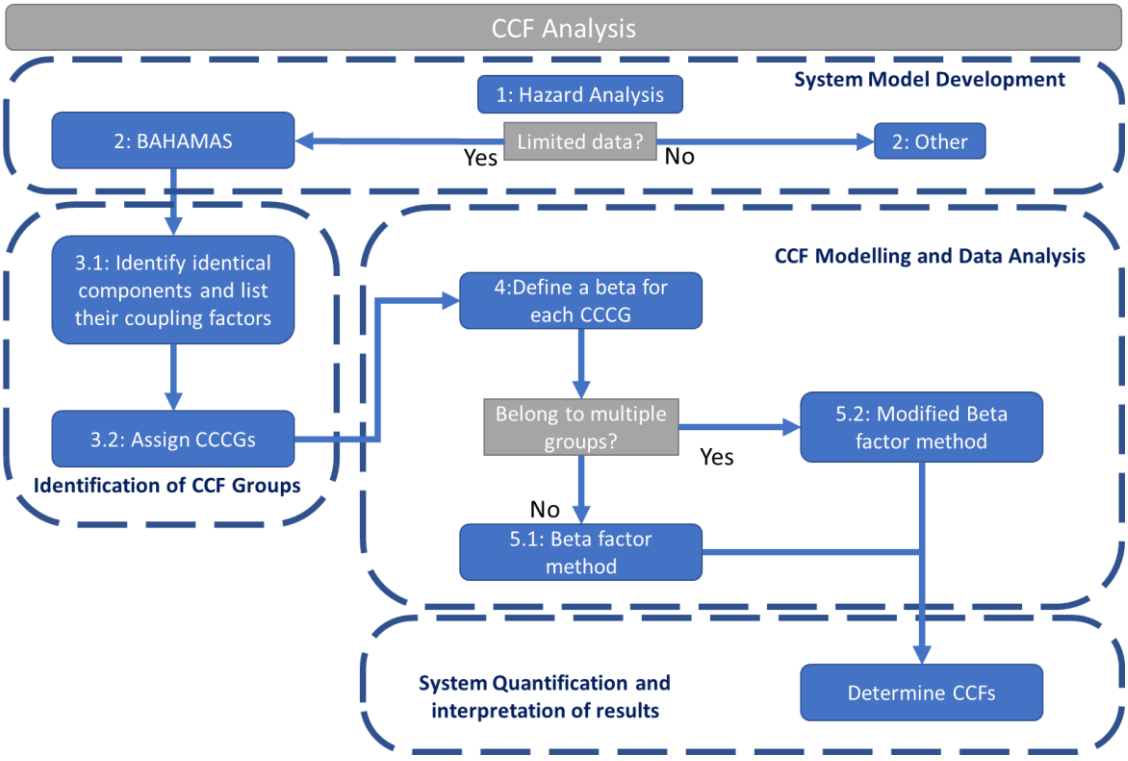

Figure 5. CCF modeling flowgraph.

2.4 Case Study

This section describes the quantification of CCFs associated with the four-division digital RTS. The structure shown in Figure 6 is based on state-of-the-art digital systems in existing NPPs [41]. The RTS consists of many redundant features including, bistable processors (BPs), logic processors (LPs), digital output modules (DOMs), selective processors (SPs), undervoltage (UV) trip devices, shunt trip (ST) devices, and reactor trip breakers (RTBs). The BPs detect a plant trip scenario from sensor information. Each BP sends a plant trip signal to the local coincidence logic (LCL) racks for each division of the RTS. The LPs within the LCL racks require at least one-out-of-two (1oo2) BP signals per division, and at least 2oo4 divisions in order to transmit a reactor trip signal. The LPs transmit trip signals through the DOMs to the SPs. To transmit a trip signal, the SPs require 2oo2 signals, one from Rack-1 (1oo2 from DOM-1 or DOM-3), and one from Rack-2 (1oo2 from DOM-2 or DOM-4). When each of these criteria have been met, the SPs transmit the trip signal to the undervoltage trip devices which proceed to activate the RTBs. There RTBs for each division actuate from the either the UV trip device or from the ST devices. The diverse protection system (DPS) activates the ST devices. Manual trip activates the RTBs manually. The control rods will insert and trip the reactor upon activation of 2oo4 divisions.

This case study describes the quantification of the CCFs associated with the BPs. Previous efforts provided a deterministic value for the total software failure probability for the BPs. The objective here is to define the dependent and independent portions of the total software failure probability previously found. The following are assumptions of the study:

- There is no diversity in the software.
- RTBs hardware is diverse.

- All hardware components are identical (unless otherwise specified).
- Installation teams and maintenance teams are assumed identical for each CCCG.
- Each set of identical components that are part of the same CCCGs have the same total failure probabilities.
- Software probability for the BP and LCL processors were quantified by BAHAMAS. They are assumed identical given limited details available to distinguish them.

The goal of the case study is to quantify the CCFs of the hardware and software components of the digital RTS. Table 6 provides the list of components for which failure rates need to be quantified. The initial demonstration of BAHAMAS assumed a generic layout for these components, consisting of four parts: an input, output, central processing unit (CPU), and memory; each part was assumed to have software. For our current work, the BP and LCL are both PLC platforms, and software is only housed in the memory of each PLC processor. Thus, the evaluation with BAHAMAS followed the same format as given in the original publications [20] but tailored to the assumptions of the current case study.

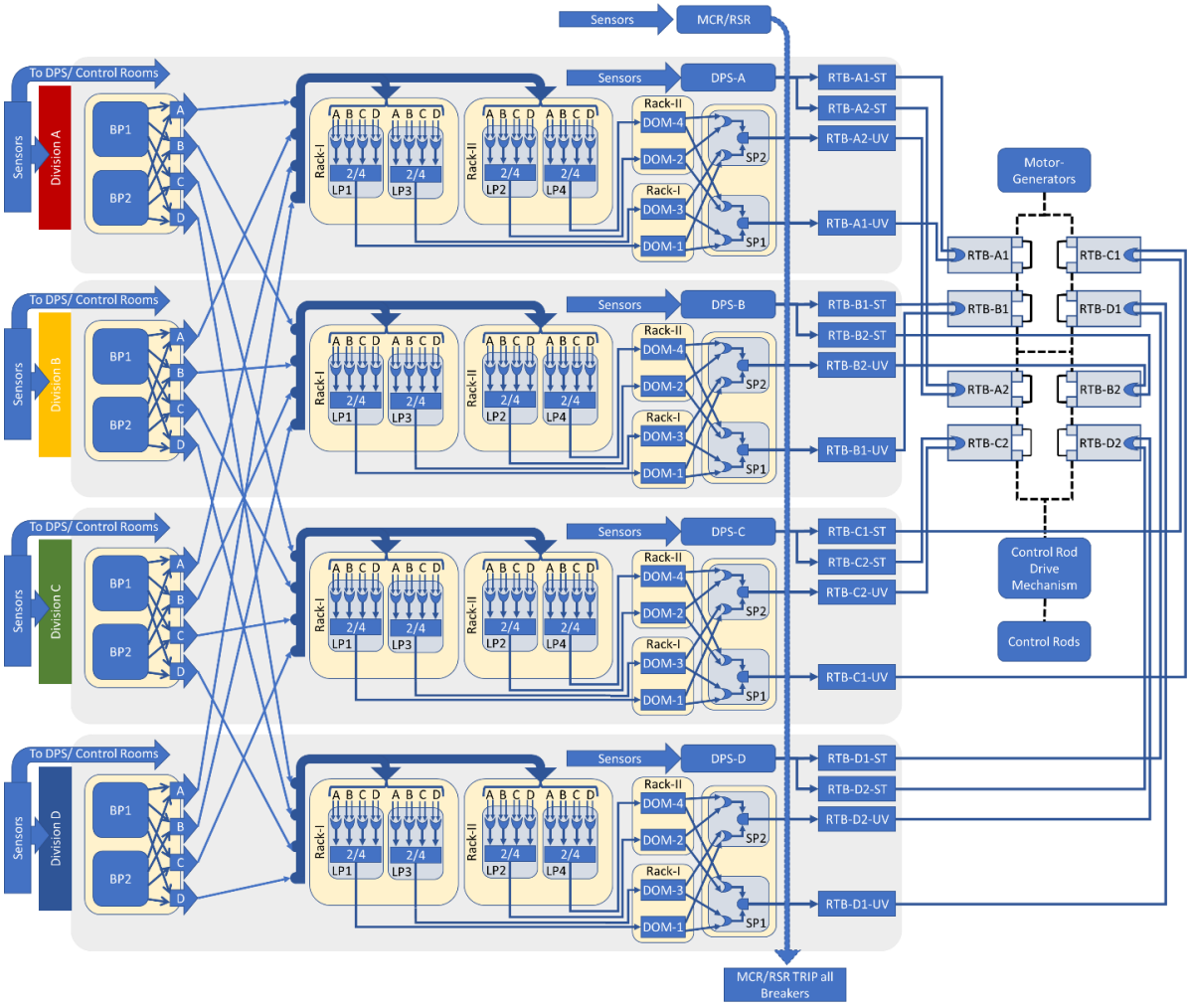

Figure 6. Detailed representation of the digital RTS.

Table 6. Total hardware and software failure probabilities for CCF analysis.

Components	Hardware failure	Total Hardware failure probability	Software failure	Total Software failure probability
BPs	YES	4.00E-5	YES	1.871E-4
LCL Processors	YES	6.48E-5	YES	1.871E-4
Digital Output Modules	YES	1.64E-5	N/A	N/A
Selective Relay	YES	6.20E-6	N/A	N/A
RTB-UV device	YES	1.70E-3	N/A	N/A
RTB-Shunt device	YES	1.20E-4	N/A	N/A
RTB RTSS1	YES	4.50E-5	N/A	N/A
RTB RTSS2	YES	4.50E-5	N/A	N/A
All hardware values came from [42]. Software probability for the BP and LCL processors were quantified by BAHAMAS. They are assumed identical given limited details available to distinguish them.				

Step 1: Perform a hazard analysis

The hazard analysis of the digital RTS shown in Figure 6 was completed previously. The details of the hazard analysis can be found in [18]. The key output from Step 1 is the identification of the hazard to be quantified in the reliability analysis.

Step 2: Perform a reliability analysis

The objective of the reliability analysis is to quantify the hazards identified in Step 1—the details of which can be found in [18]. The subsequent steps for CCF modeling are for the limited-data scenario. Thus, according to Figure 5, the steps follow the BAHAMAS quantification. The key output from Step 2 is the total component software failure probabilities. These will be broken down into their respective independent and dependent portions in the subsequent steps.

Step 3: Identify potential CCCGs

The objective of Step 3 is to identify potential CCCGs, for the component of interest. This process is done by first identifying all identical components within the system, then listing their coupling factors.

Step 3.1: Identify identical components and list their coupling factors

Identical components will share coupling factors. NUREG/CR-5485 [24] provides a list of common coupling factors. These factors include the same design, hardware, function, installation and maintenance procedures, interfaces, locations, environments, and others. Components that share these features should be considered as part of a potential CCCG. There are eight identical BPs in the RTS, two per division. They each have identical function and are assumed to share the same features except for location of installation. Table 7 shows a simplified list of the coupling factors.

Table 7. Coupling factors for the BPs.

Component	Coupling Factor 1	Coupling Factor 2	Coupling Factor 3	Coupling Factor 4	Coupling Factor 5
Division A: BP1	Function	Hardware X	Software Y	Manufacturer Z	Division A
Division A: BP2	Function	Hardware X	Software Y	Manufacturer Z	Division A
Division B: BP1	Function	Hardware X	Software Y	Manufacturer Z	Division B
Division B: BP2	Function	Hardware X	Software Y	Manufacturer Z	Division B
Division C: BP1	Function	Hardware X	Software Y	Manufacturer Z	Division C
Division C: BP2	Function	Hardware X	Software Y	Manufacturer Z	Division C
Division D: BP1	Function	Hardware X	Software Y	Manufacturer Z	Division D
Division D: BP2	Function	Hardware X	Software Y	Manufacturer Z	Division D

Step 3.2: Assign components to CCCGs

Once the associated components and coupling factors have been identified, the CCCGs can be assigned. Each CCCG shall be formed based on a unique set of coupling factors. The BPs shown in Table 7 are identical except for location. For this case, a CCCG should be formed to account for CCFs based each set shared features. The BPs all share function, hardware, software, and manufacturer; thus, a CCCGs should be formed based on these features. Location remains an additional coupling factor identified which can be used to form CCCGs. Table 8 shows the CCCGs for the BPs.

Table 8. CCCGs for the BPs.

CCCGs		Coupling Factors
1	All BPs	Function, Hardware, Software, & Manufacturer
2	Division A: BP1, BP2	Division A
3	Division B: BP1, BP2	Division B
4	Division C: BP1, BP2	Division C
5	Division D: BP1, BP2	Division D

Step 4: Define CCCG parameters

The objective of Step 4 is to define the beta-factor parameters to be used for CCF modeling. This process follows the PBF-2. This is done for each CCCG. Note groups 2–5 in Table 8 contain the same details and will have identical betas. PBF-2 consists of an evaluation of defense categories which pertain to the prevention of CCFs. Each CCCG receives a score for each sub-factor category. Table 4 shows the sub-factors and their weights. An "E" indicates the CCCG is well defended against CCFs (i.e., A= worst, E = best).

Table 9 shows the sub-factor scores applied to the BPs of CCCG2 and the calculation for beta based on Equation (12). Table 10 shows the beta values for each CCCG.

Table 9. Sub-factor scores for BPs CCCG 2 (Division CCF).

Sub-factors	**Scores for CCCG2**	
Redundancy (& Diversity)	A	1750
Separation	A	2400
Understanding	A	1750
Analysis	D	25
MMI	C	175
Safety Culture	E	5
Control	D	25
Tests	C	70
$\beta = \frac{\sum(1750 + 2400 + 1750 + 25 + 175 + 5 + 25 + 70)}{50000} = 0.124$		

Table 10. Beta factors for each BP CCCG.

CCCGs	**Group Beta**	**Coupling Factors**
All BPs	.0454	Function, Hardware, Software, & Manufacturer
Division A: BP1, BP2	.124	Division A
Division B: BP1, BP2	.124	Division B
Division C: BP1, BP2	.124	Division C
Division D: BP1, BP2	.124	Division D

Step 5: Calculate the independent and depend failures

The purpose of Step 5 is to evaluate the independent and dependent failures for each component of the CCCG. The evaluation is conducted using either the tradition BFM or the PBF2.

Step 5.1: Single CCCG: apply the traditional BFM

If a component belongs to only a single CCCG, then the dependent and independent failures are found using the traditional BFM. Originally proposed by Fleming in 1975, the BFM it is arguably one of the most well-known methods [28]. This is a component failure ratio model in which the total failure rate is related to the independent and dependent failures by means of the ratio beta [27].

The model is defined using the following:

$$Q_k^m = \begin{cases} (1-\beta)Q_t & k = 1 \\ 0 & m > k > 1 \\ \beta Q_t & k = m \end{cases} \quad (13)$$

Q_t = the total failure probability of one component.

m = number of components in CCCG.

k = number components to fail due to common cause.

β = ratio of CCFs to total failures.

Q_k^m = the failure rate or probability of an event involving of k/m components in a CCCG of size m.

None of the components shown in Table 10 belong to only one CCCG. Thus, there will be no need to apply the conventional BFM for the BPs. Details for multiple CCCGs are provide subsequently.

Step 5.2: Multiple CCCG: apply the modified BFM

If a component belongs to multiple CCCG, then the dependent and independent failures are found using the modified BFM. The details of the modified BFM are given earlier in this chapter. The modified BFM requires each CCCG for a component be identified. For example, Division A BP1 is found in CCCG1 and CCCG2 according to Table 10. The evaluation for the independent and dependent failures for Division A BP1 are shown below. Equation (8) is used to solve for the CCF probability for each CCCG of A1. The process requires the total failure probability ($Q_t = 1.871E-4$) that was found by BAHAMAS. The total independent failure is found using Equation (11). Results are shown below:

$$P(CCCG_1) = (\beta_1)Q_t = 8.494E-6 \quad (14)$$

$$P(CCCG_2) = (\beta_2)Q_t = 2.320E-5 \quad (15)$$

$$Q_D = P(CCCG_1) + P(CCCG_1) = 3.169E-5 \quad (16)$$

$$Q_I = (1-\beta_t)Q_t = \left[1 - \sum_{1}^{w}(\beta_w)\right]Q_t = 1.554E-4 \quad (17)$$

2.5 Results of CCF Analysis

The results of the CCF analysis are shown in Table 11 and Table 12. Each table provides the betas used for each CCCG. Note the categories, Individual, Rack, Division, and ALL, correspond to the CCCGs. The CCCG All contains all the identical components within the RTS. Not all components have every CCCG category; hence, there are not Rack or Division CCCGs for each component of the RTS.

Table 11. Hardware failures and beta factors for each CCCG of RTS components.

Component	Individual	Rack	Division	All	Total
BPs	3.322E-05		4.960E-06	1.816E-06	4.00E-05
(CCCG Beta)			0.124	0.0454	0.1694
LCL Processors	5.098E-05	6.901E-06	3.972E-06	2.942E-06	6.48E-05
(CCCG Beta)		0.1065	0.0613	0.0454	0.2132
DOMs	1.462E-05	1.175E-06	4.333E-07	1.729E-07	1.64E-05
(CCCG Beta)		0.07162	0.02642	0.01054	0.10858
Selective Relay	5.556E-06		5.525E-07	9.164E-08	6.20E-06
(CCCG Beta)			0.08912	0.01478	0.1039
RTB-UV device	1.674E-03			2.604E-05	1.70E-03
(CCCG Beta)				0.01532	0.01532
RTB-Shunt device	1.182E-04			1.838E-06	1.20E-04
(CCCG Beta)				0.01532	0.01532
RTB RTSS1	4.302E-05			1.976E-06	4.50E-05
(CCCG Beta)				0.04392	0.04392
RTB RTSS2	4.302E-05			1.976E-06	4.50E-05
(CCCG Beta)				0.04392	0.04392

Table 12. Software failures and beta factors for each CCCG of RTS components

	Individual	Rack	Division	All	Total

BPs	1.554E-04		2.320E-05	8.494E-06	1.871E-04
(CCCG Beta)			0.124	0.0454	0.1694
LCL Processors	1.472E-04	1.993E-05	1.147E-05	8.494E-06	1.871E-04
(CCCG Beta)		0.1065	0.0613	0.0454	0.2132

2.6 Summary and Conclusions

This chapter presented a CCF analysis of the highly redundant digital RTS. The chapter introduces CCF modeling by providing essential definitions and attributes of CCFs. A CCF is the failure of two or more components due to the same cause at the same time. For a CCF to occur, two factors must exist (1) a root failure case that is common between two or more components and (2) the existence of a coupling mechanism. It is the coupling mechanism (or factor) which creates the condition for a failure cause to affect multiple components. CCF analyses depend on the identification of these two factors. A hazard analyses identifies root failures leaving the coupling factors as a major focus area for this chapter.

When a group shares coupling factors, they form a CCCG. It is essential CCCGs be identified as part of a proper CCF analysis. For the majority of analyses, the components that belong to a CCCG do not belong to any other groups. This is because the components have no other coupling factors to share with components outside of their existing group. A challenge arises when this is no longer the case. When components can be grouped into multiple CCCGs, it becomes difficult to model their failure probabilities using conventional methods, and there is the potential for erroneous and illogical results. Because of the unique and highly redundant structure of digital RTS, there is a need to model several of its components as part of multiple CCCGs.

With the case study as clear motivation, this work proceeded to investigate possible solutions to modeling multiple CCCGs. The chapter describes a procedure for modeling the multiple CCCG scenario using a combination of existing tools. The modified BFM was selected to model components with multiple CCCGs as it was constructed for such a case. Normally CCF methods rely on historical data or experience to define model parameters. However, with limited data available concerning the novel digital RTS, a solution for quantifying model parameters had to be found. Without the parameters, the modified BFM would not work. Several elicitation methods were reviewed, and PBF-2 was selected. The novel application of PBF-2, together with the modified BFM, allows for a successful quantification process for the multiple CCCGs under a limited-data scenario.

Several aspects of CCF modeling within the IRADIC technology remain for future work. First, the goal of the PBF-2 is to define a CCF parameter (beta) whose value reflects the quality of a component's defenses against CCF. The method only considers eight sub-factors for assessing beta. The PBF-2 model can be improved and even tailored for software-related CCFs. There have been other publications indicating important factors which affect a component's CCF probability. Some factors address specific coupling mechanisms. The PBF-2 could be improved by incorporating additional sub-factors. The Cause-Defense matrix formalism provides a structure that may lend itself to an improved PBF-2. Of course, each of these improvements would remain part of the limited-data focus. Once data is available, such a method is no longer needed for an elicitation process. The modified BFM can also be improved. In its current form the method, like other ratio-based methods, it is limited to identical components with identical total failure probabilities; future work may provide guidance for when this case is no longer true.

In conclusion, the CCF modeling process developed for this work provides an effective means of quantification given the scope of the case study. Despite the known limitations, the method provided a means to account for software CCFs in PRA for a limited-data scenario. Future collaborations with industry partners may afford our team the opportunity to investigate the data-sufficient scenario. In this case, there will be many opportunities to improve our models. In any case, there remain avenues to explore in the upcoming fiscal years.

3. CONSEQUENCE ANALYSIS OF A GENERIC PWR MODEL WITH IMPROVED FAULT TREES

SAPHIRE is a probabilistic risk and reliability assessment software tool developed and maintained by INL for the U.S. NRC [29]. This chapter documents the consequence analysis of a generic PWR SAPHIRE model with improved digital RTS and ESFAS FTs. Section 3.1 describes the generic PWR SAPHIRE model. In Section 3.2, the scenario to be analyzed is introduced as well as the original event tree (ET) models for these scenarios including a FT for the failure of an analog RTS and one CCF basic event for the ESFAS failure. Section 3.3 compares the original FTs for analog RTS and ESFAS and the new FTs for digital RTS and ESFAS. Results for consequence analysis of these selected ET models are discussed in Section 3.4.

3.1 Introduction of INL Generic PWR SAPHIRE Model

In past decades, SAPHIRE has been widely used to model plant response to both internal hazards (e.g., general transients, loss of offsite power, and loss of feedwater) and external hazards (e.g., seismic, fire, external flooding, and high wind). SAPHIRE 8 is a powerful PRA software that has both the basic PRA modeling capabilities such as creating event trees and fault trees, defining and assigning basic event failure data, linking and solving event trees and fault trees, documenting and reporting the results, and the advanced capabilities such as integrated Level 1 and Level 2 PRA analysis, performing sensitivity and uncertainty analyses, and conducting specialized analyses for the NRC's Accident Sequence Precursor program and Significance Determination Process. [43]

A generic internal events PRA model was developed at INL using SAPHIRE 8 for a typical PWR plant for the accident scenario analysis with various initiating events, which has been applied for different purposes including plant-level scenario-based risk analysis for Enhanced Resilient Plant (ERP) during station black-out (SBO) and loss-of-coolant accident (LOCA) [43], risk-informed analysis for an ERP with Accident Tolerant Fuel (ATF), optimal use of Diverse and Flexible Coping Strategy (FLEX), and new passive cooling systems [44] [45]. There are 24 ETs included in this generic PWR SAPHIRE model, as listed below:

- EQK-BIN-1: Seismic Initiator (0.05 - 0.3g)
- EQK-BIN-2: Seismic Initiator (0.3 - 0.5g)
- EQK-BIN-3: Seismic Initiator (> 0.5)
- INT-ISL-HPI: ISLOCA IE 2-CKV HPI interface
- INT-ISL-LPI: ISLOCA IE 2-CKV LPI interface
- INT-ISL-RHR: RHR pipe ruptures
- INT-LLOCA: LARGE LOCA
- INT-LOACA: LOSS OF VITAL 4160V AC BUS A
- INT-LOCCW: LOSS OF CCW INITIATING EVENT
- INT-LOCHS: LOSS OF CONDENSER HEAT SINK
- INT-LODCA: LOSS OF VITAL 125 VDC BUS A
- INT-LODCB: LOSS OF VITAL 125 VDC BUS B
- INT-LOMFW: LOSS OF MAIN FEEDWATER
- INT-LONSW: LOSS OF NSW COOLING INITIATING EVENT
- INT-LOOPGR: LOSS OF OFFSITE POWER INITIATOR (GRID-RELATED)
- INT-LOOPPC: LOSS OF OFFSITE POWER INITIATOR (PLANT-CENTERED)
- INT-LOOPSC: LOSS OF OFFSITE POWER INITIATOR (SWITCHYARD-RELATED)
- INT-LOOPWR: LOSS OF OFFSITE POWER INITIATOR (WEATHER-RELATED)
- INT-MLOCA: MEDIUM LOCA
- INT-SGTR: SG TUBE RUPTURE

- INT-SLBOC: STEAM LINE BREAK OUTSIDE CONTAINMENT
- INT-SLOCA: SMALL LOCA
- INT-TRANS: GENERAL PLANT TRANSIENT
- INT-XLOCA: EXCESSIVE LOCA INITIATING EVENT.

3.2 Scenario Selections

This project applies SAPHIRE 8 for the FT development of DI&C system and combines these improved FTs with the existing generic PWR ET models. The consequence analysis of DI&C failures documented in this report covers the following accident scenarios: INT-TRANS (initiating event - general plant transient) with ATWS (anticipated transient without scram), LOSC (loss of seal cooling), SBLOCA (small-break loss-of-coolant accident) and MBLOCA (medium-break LOCA). These five accident ETs are respectively shown from Figure 7 to Figure 11. INT-TRANS is selected here for the DI&C consequence analysis because it shows relatively significant impacts of digital failures to key plant responses. RTS and ESFAS failures are treated as initiating events or important basic events included in cut sets that have significant contributions to change of CDF (ΔCDF).

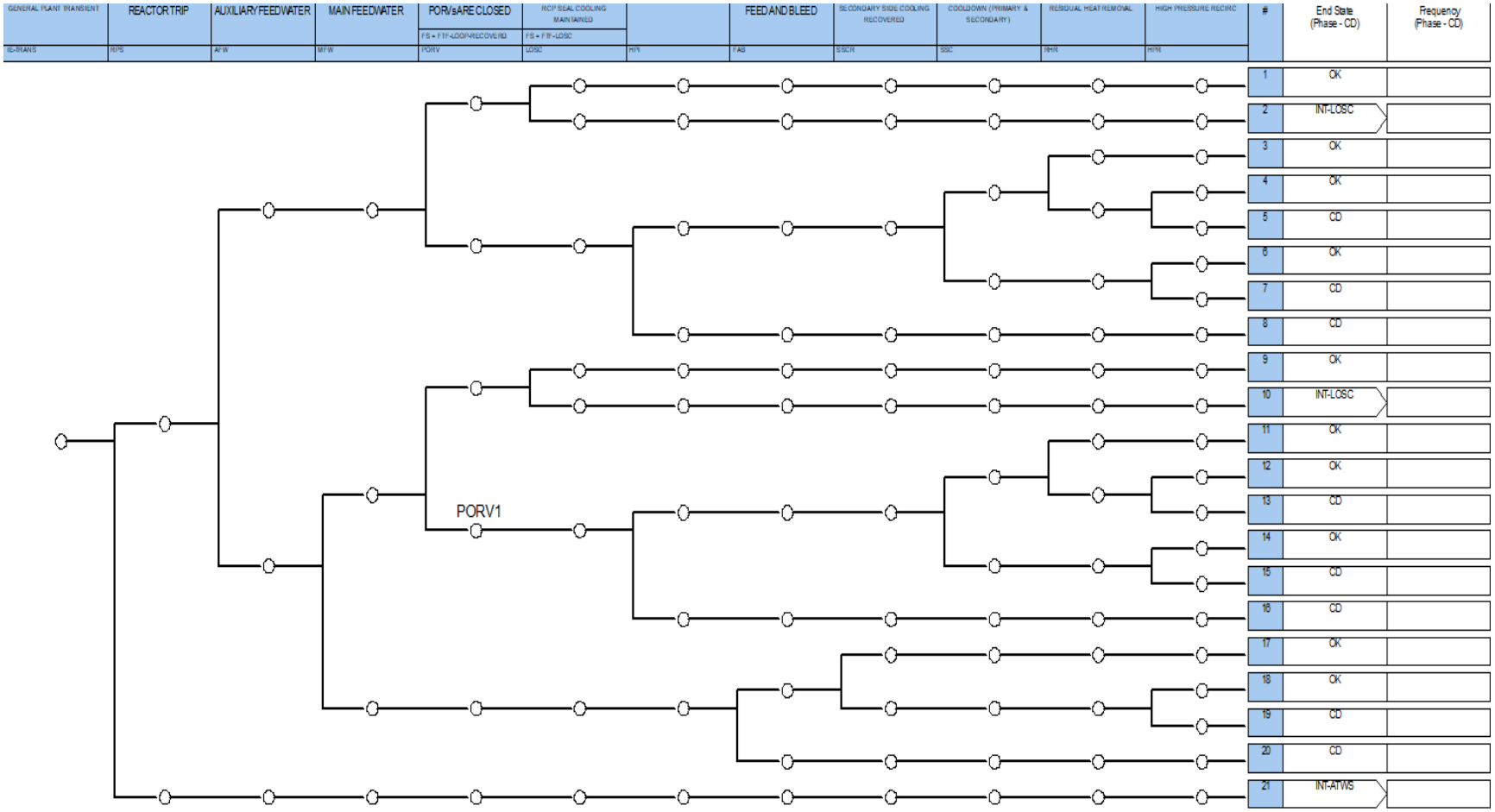

Figure 7. Generic PWR ET for general plant transient (INT-TRANS).

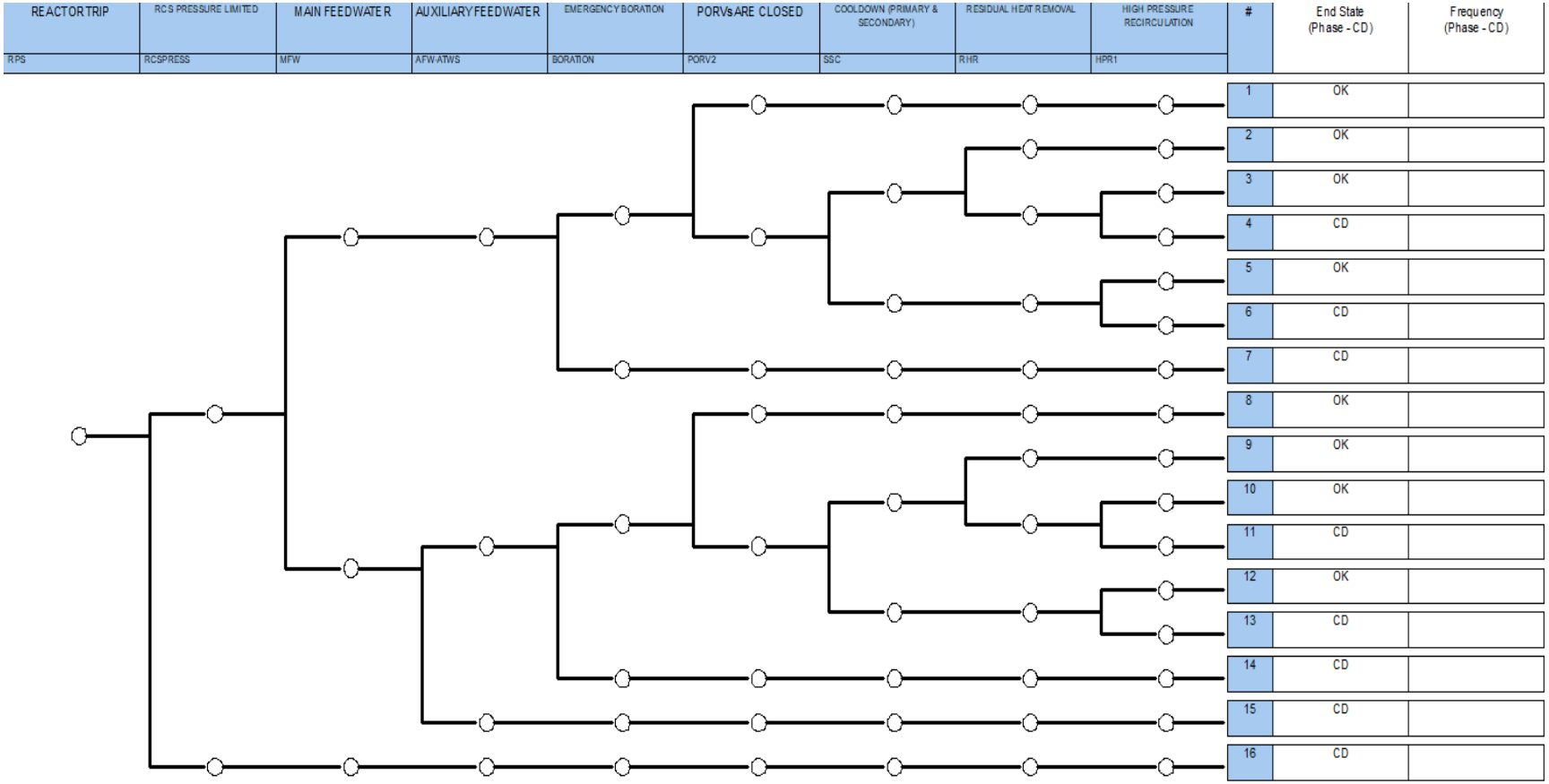

Figure 8. Generic PWR ET for INT-ATWS.

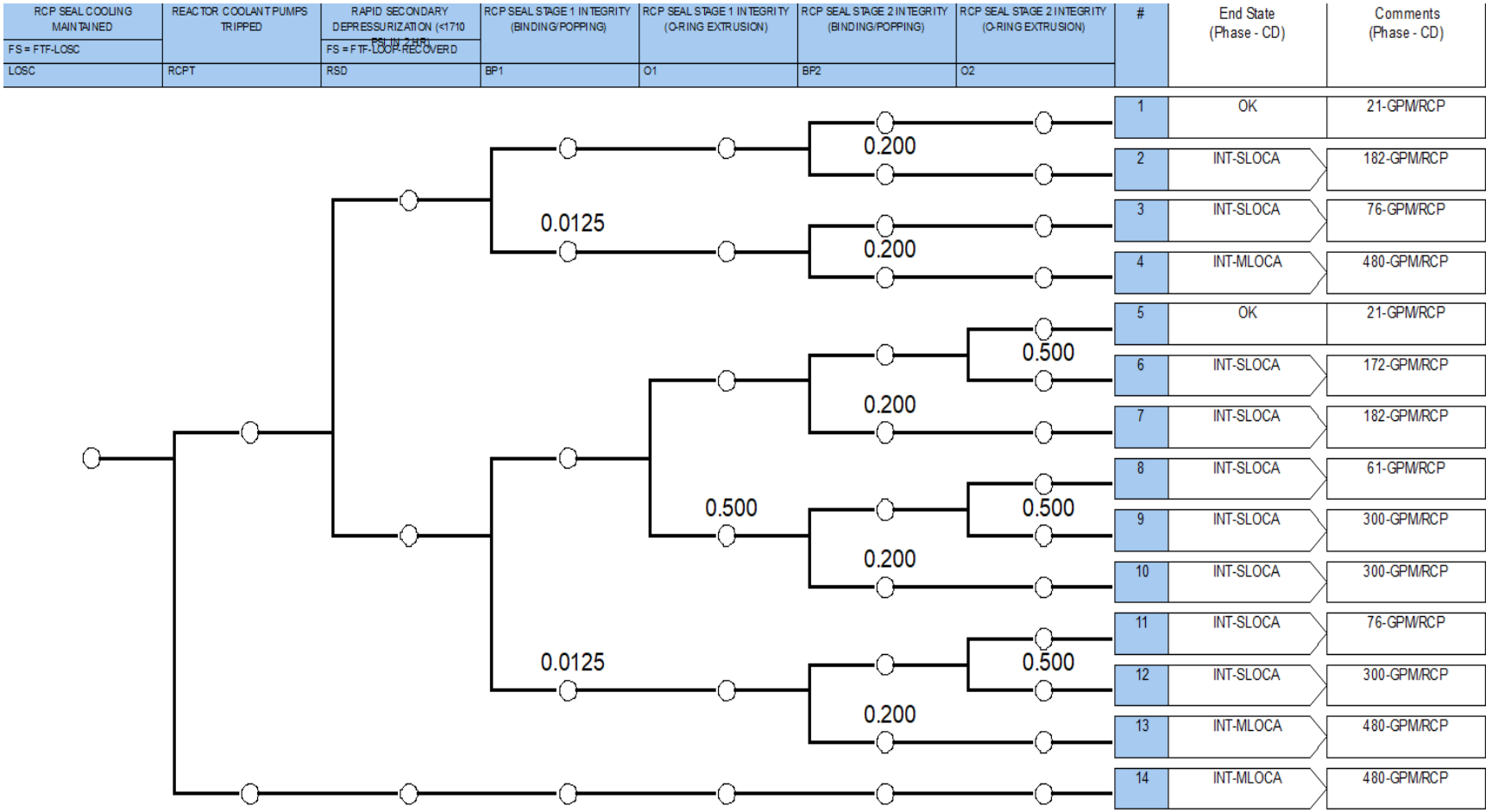

Figure 9. Generic PWR ET for loss of seal cooling (INT-LOSC).

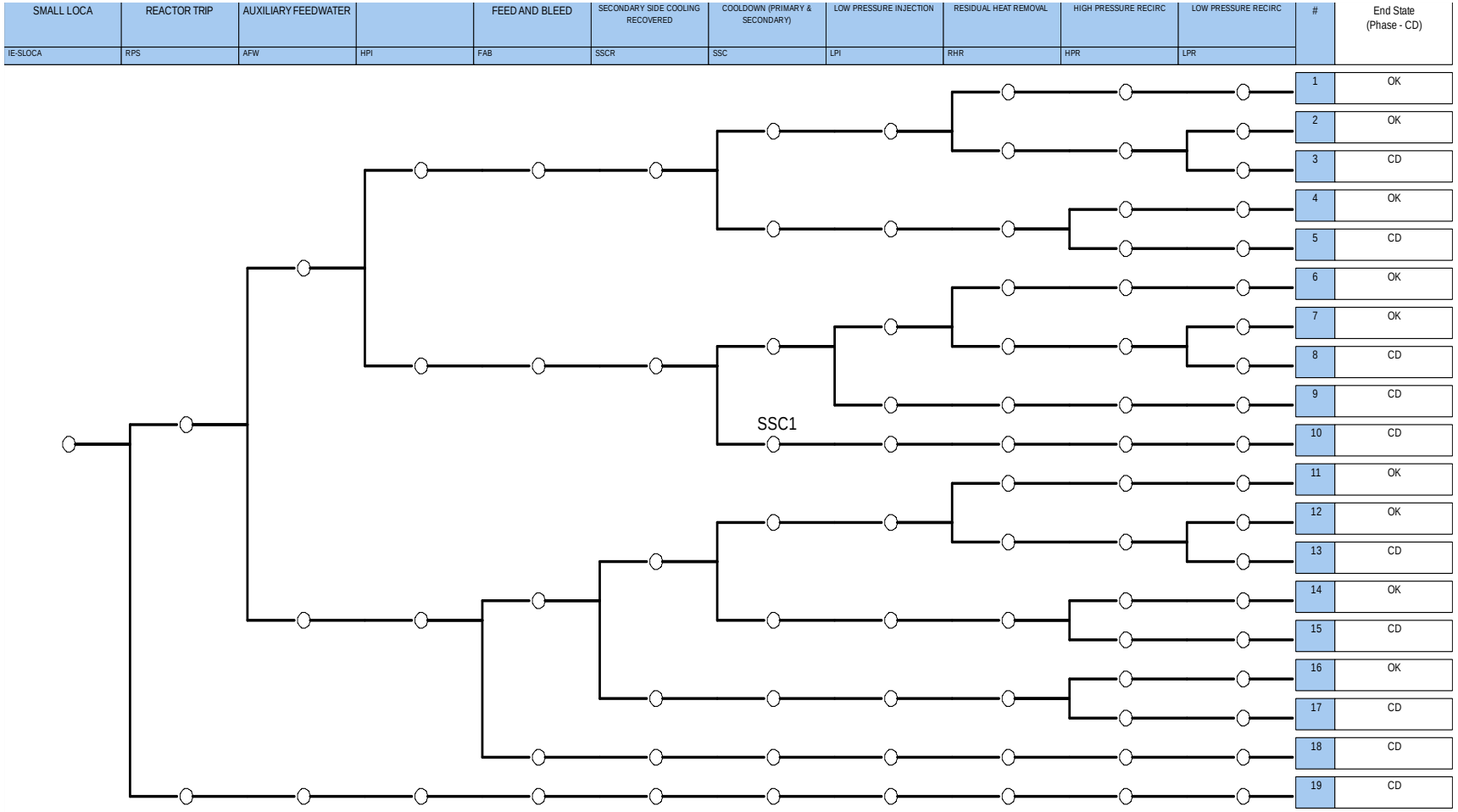

Figure 10. Generic PWR ET for small-break LOCA (INT-SLOCA).

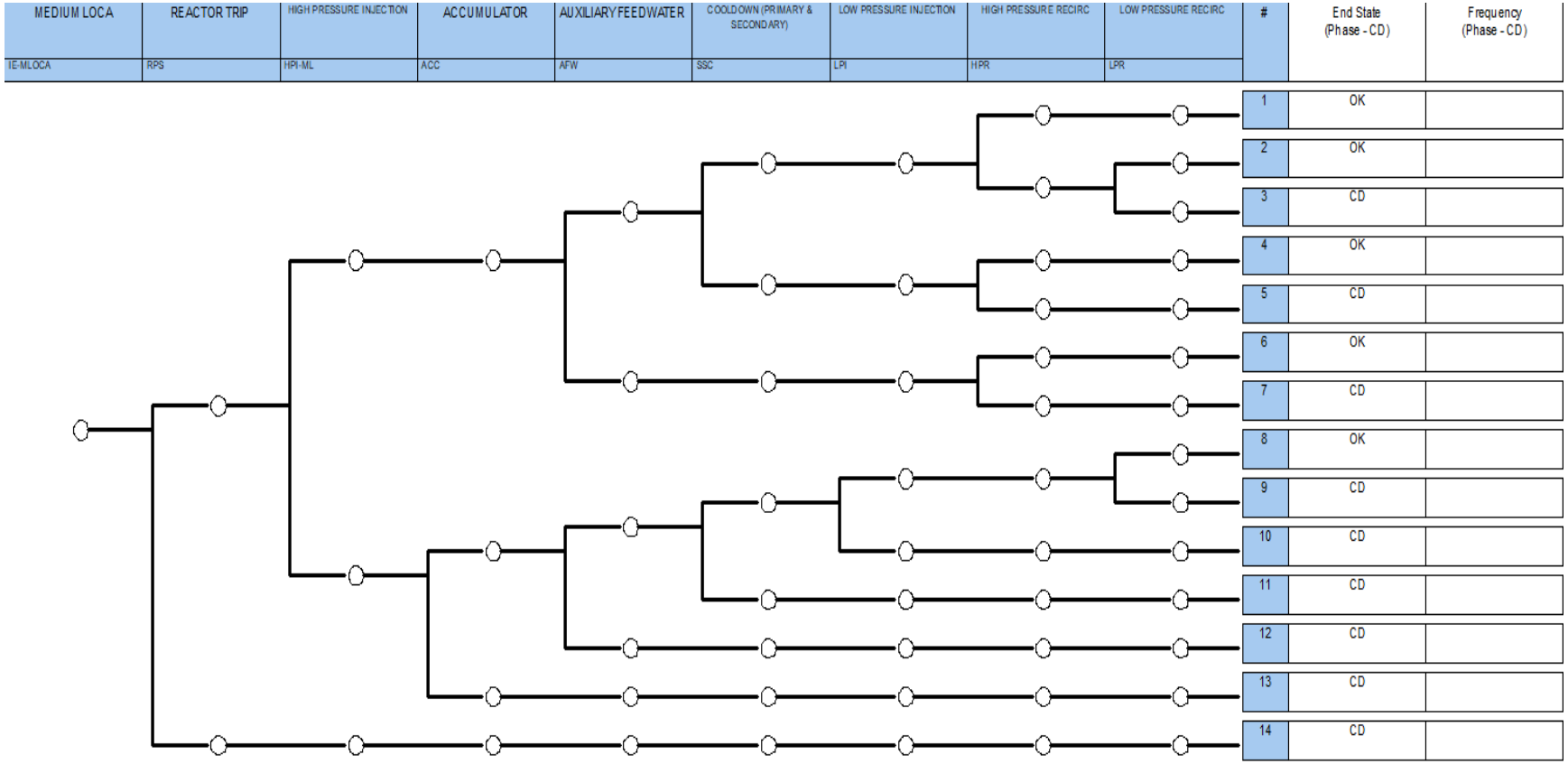

Figure 11. Generic PWR ET for medium-break LOCA (INT-MLOCA).

3.3 Original and Improved Fault Trees for HSSSR DIC Systems

3.3.1 Original Fault Tree for Reactor Trip System

The original RTS-FT in the generic PWR SAPHIRE model has identified analog/hardware failures in detail. The main logic of original RTS-FT is shown in Figure 12. A two-train analog RTS was modeled; the main failure modes include electric failures, CCF of rod cluster control assembly (RCCA) fail to drop, contribution of seismic events, operator errors, and RTS failures during test and maintenance.

This FT was quantified with SAPHIRE 8 using a truncation level of 1E-12, RTS failure probability is 4.288E-6 with five cut sets. Table 13 lists these cut sets for the original RTS-FT; it shows the main contributed basic events are the CCF of reactor trip breaker A, B and CCF of RCCA, which contribute about 65.77% of the total RTS failure. Results indicate hardware CCFs are the main concerns for the failure analog safety-related redundant I&C systems.

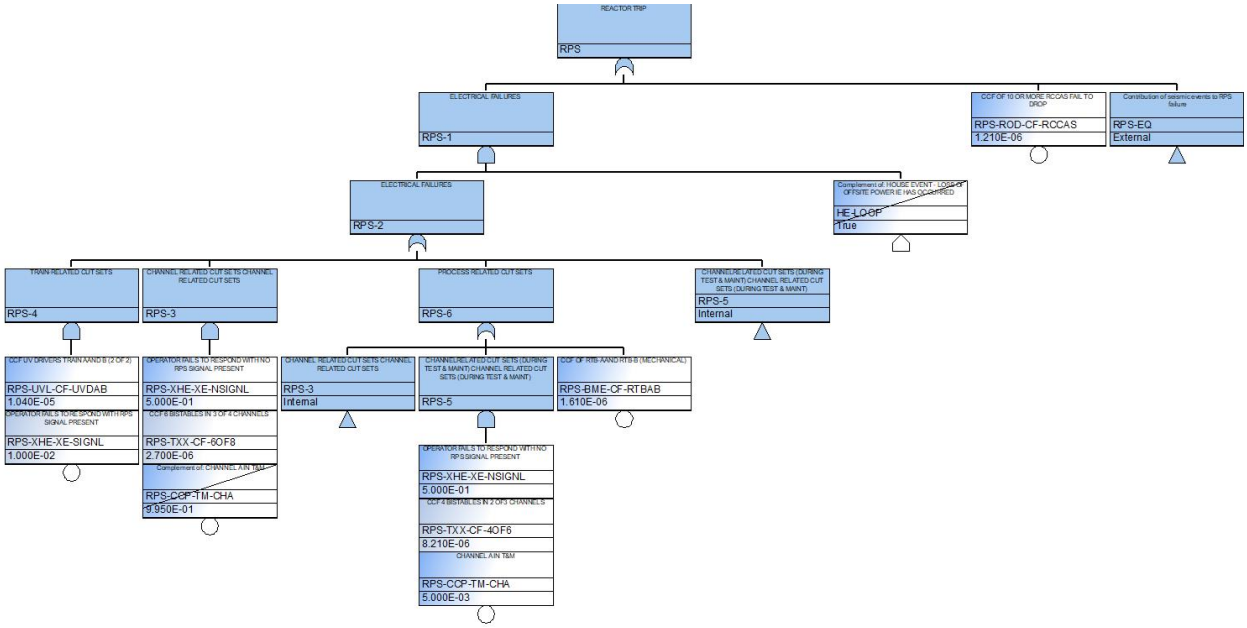

Figure 12. Main fault tree of original RTS-FT in the generic PWR SAPHIRE model.

Table 13. Cut sets for the original RTS-FT.

#	Probability	Total %	Cut Sets
1	1.610E-6	37.55	RPS-BME-CF-RTBAB
2	1.343E-6	31.33	RPS-CCP-TM-CHA, RPS-TXX-CF-6OF8, RPS-XHE-XE-NSIGNL
3	1.210E-6	28.22	RPS-ROD-CF-RCCAS
4	1.040E-7	2.43	RPS-UVL-CF-UVDAB, RPS-XHE-XE-SIGNL
5	2.052E-8	0.48	RPS-CCP-TM-CHA, RPS-TXX-CF-4OF6, RPS-XHE-XE-NSIGNL
Total	4.288E-6	100	-

3.3.2 Original Fault Tree for Engineered Safety Features Actuation System

In the original generic PWR SAPHIRE model, ESFAS failure is presented using a CCF of ESF actuation signal in both Train A and B, named ESF-VCF-CF-TRNAB with a probability as 6.420E-4. These CCF basic events are used in the FTs of several top events in IE-TRANS scenarios including AFW (representing failure of auxiliary feedwater), AFW-ATWS (representing failure of auxiliary feedwater for ATWS scenarios), HPI (representing failure of high-pressure injection), and LPI (representing failure of low-pressure injection). It should be noted another basic event is used to represent "operator fails to manually initiate safety features," which will be replaced by an integrated FT representing failure of digital HSI and operator errors in future work.

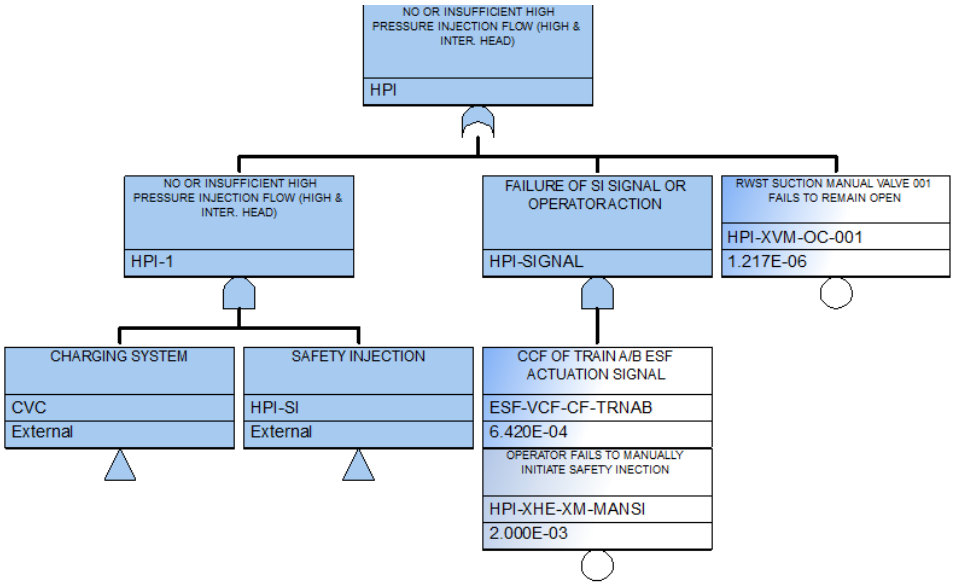

Figure 13. Main FT of HPI failure in the generic PWR SAPHIRE model where CCF of analog ESFAS is considered.

3.3.3 Improved Fault Tree for Digital Reactor Trip System

The integrated FT for the failure of 4-division RTS is added into the generic PWR SAPHIRE model and quantified in this section. This RTS-FT was developed using RESHA in FY-20 [18] and quantified using BAHAMAS in FY-21. Compared with the original FT for a 2-train analog RTS, this improved RTS-FT keeps part of failure modes: CCF of RCCA fail to drop, contribution of seismic events, operator errors, and RTS failures during test and maintenance and extends the electric failures to an integrated automatic RTS failure including both hardware and software failures. The main logic of this integrated RTS-FT is displayed in Figure 14. (please contact Han Bao han.bao@inl.gov for more details about this FT).

This FT was also quantified with SAPHIRE 8 using a truncation level of 1E-12, RTS failure probability is 1.270E-6 with 27 cut sets. Table 14 lists part of these cut sets with significant contributions. Compared with the original RTS-FT, the total failure probability of RTS system is reduced about 50%. Mechanical CCF of RCCA becomes the main contributor (>95% of total), the software CCFs do not significantly affect the reliability of digital RTS because of the highly redundant design and high reliability of PLC-based digital components.

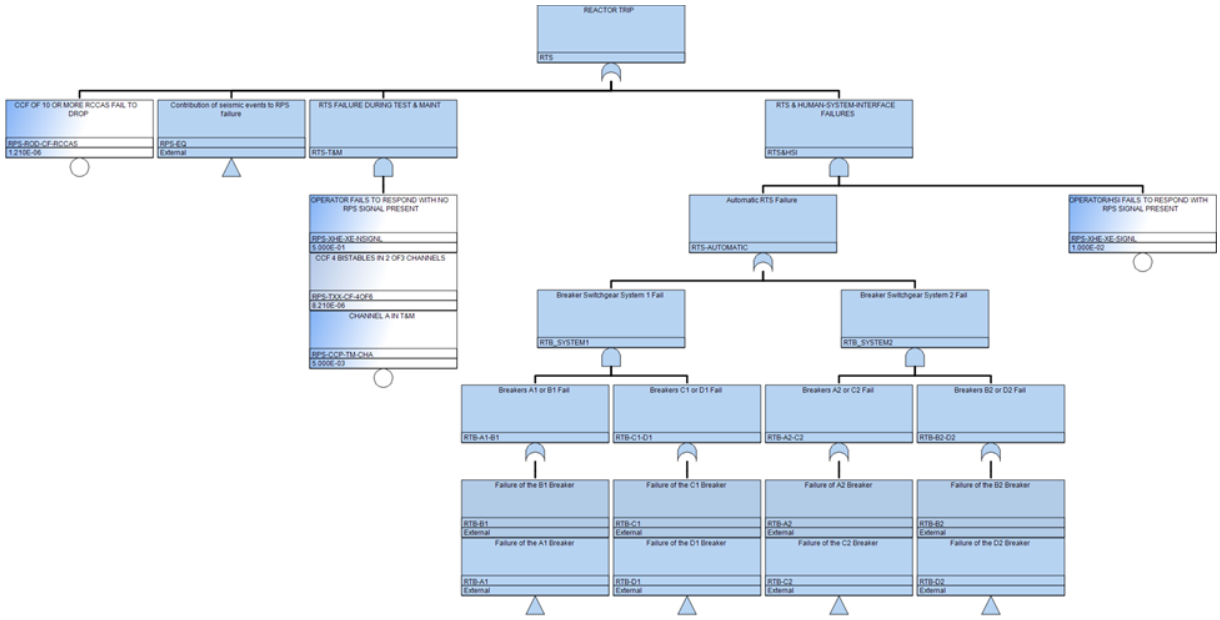

Figure 14. Main fault tree of improved RTS-FT using IRADIC technology.

Table 14. Cut sets for the improved RTS-FT.

FT Name	#	Probability	Total %	Cut Sets
Improved RTS-FT	1	1.210E-6	95.25	RPS-ROD-CF-RCCAS
	2	2.052E-8	1.62	RPS-CCP-TM-CHA, RPS-TXX-CF-4OF6, RPS-XHE-XE-NSIGNL
	3	1.976E-8	1.56	RPS-XHE-XE-SIGNL, RTB-SYS-2-HD-CCF
	4	1.976E-8	1.56	RPS-XHE-XE-SIGNL, RTB-SYS-1-HD-CCF
	Total	1.270E-6	100	-
Original RTS-FT	Total	4.288E-6	100	-

3.3.4 Improved Fault Tree for Digital Engineered Safety Features Actuation System

The integrated FT for the failure of 4-division ESFAS for the actuations of AFW, HPI, and LPI is added into the generic PWR SAPHIRE model and quantified in this section. This ESFAS-FT was developed using RESHA in FY 2020 [19] and quantified using BAHAMAS in FY 2021. Compared with the original ESFAS-CCF, this integrated ESFAS-FT has a complicated logic to match the 4-division digital ESFAS structure deployed in APR-1400, as shown in Figure 15. More details about this integrated ESFAS-FT can be found in [19]. (please contact Han Bao han.bao@inl.gov for more details about this FT).

This FT was also quantified with SAPHIRE 8 using a truncation level of 1E-12, ESFAS failure probability is 2.600E-5 with 13 cut sets. Table 15 lists part of these cut sets with significant contributions. Compared with the original ESFAS failure, the total failure probability of ESFAS system is significantly reduced. Hardware CCF of ESF-component interface modules (ESF-CIMs) becomes the main contributor, and the software CCFs do not significantly affect the reliability of digital RTS because of the high-redundant design and high reliability of PLC-based digital systems.

Accordingly, the impact of ESFAS failure to the actuations of safety features were estimated by solving the FTs of AFW, AFW-ATWS, HPI, and LPI; results are listed and compared in Table 16. All the failure probabilities of these safety features have been reduced due to the decrease of ESFAS failure probability.

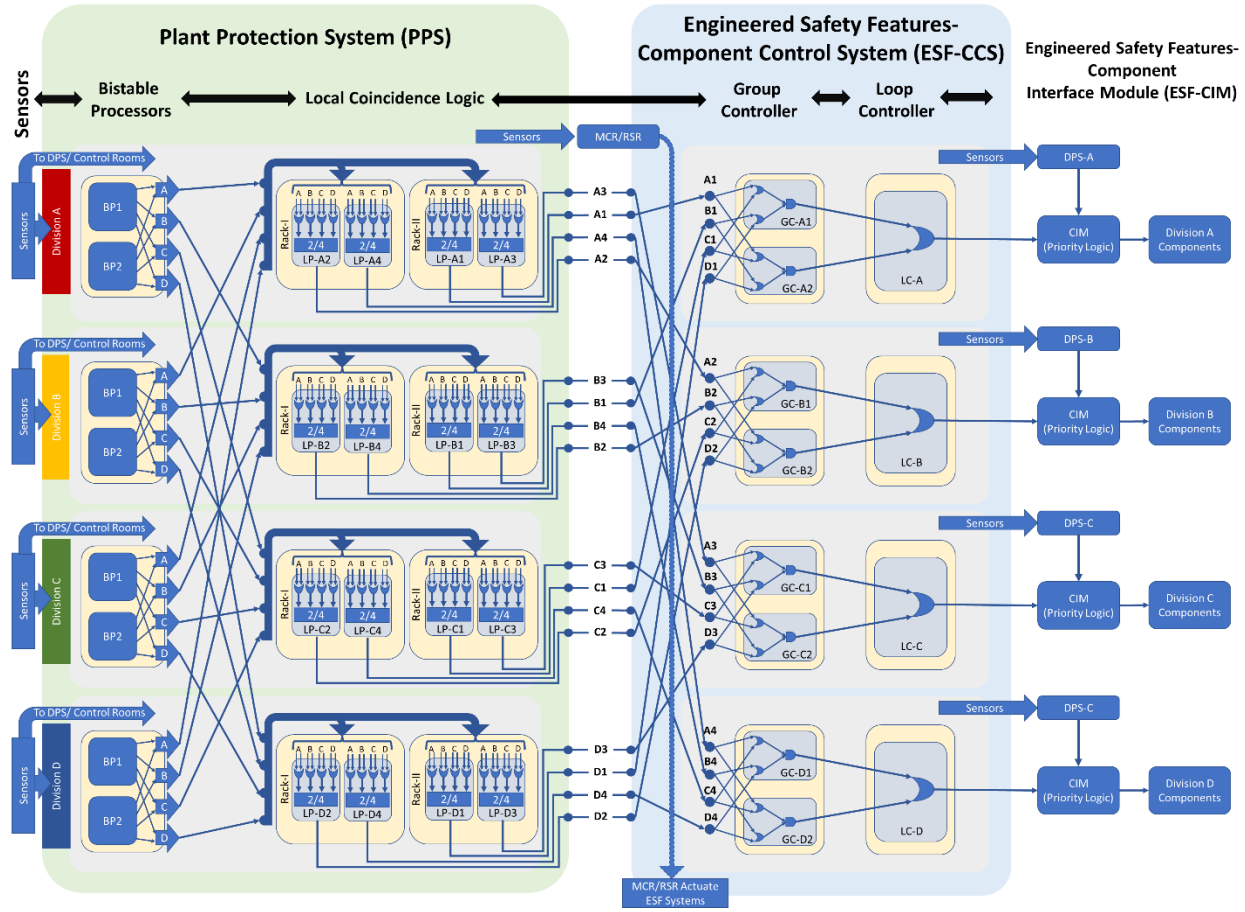

Figure 15. ESFAS functional logic.

Table 15. Cut sets of the improved ESFAS-FT.

FT Name	Probability	Total %	# of Cut Sets
Improved ESFAS-FT	2.600E-5	100	13
Original ESFAS failure	6.420E-4	100	1

Table 16. Comparison of the top events with original ESFAS-CCF basic event and improved ESFAS-FT.

Top Event	Probability		# of Cut Sets	
	Original	New	Original	New
Failure of AFW	1.487E-5	1.240E-5	1539	1551
Failure of AFT-ATWS	2.367E-4	2.343E-4	906	918
Failure of HPI	1.104E-5	9.803E-6	1163	1172
Failure of LPI	8.416E-4	2.258E-4	1567	1579

3.4 Accident Scenario Analysis for General Plant Transient

In this section, IE-TRANS, IE-SBLOCA, and IE-SBLOCA ET tree quantification results are discussed. As IE-ATWS and IE-LOSC are sub-ETs of IE_TRANS developed in the generic PWR SAPHRIE model, they will be discussed in the section on INT-TRANS.

3.4.1 INT-TRANS

The generic PRA model representing general plant transient as INT-TRANS was shown in Figure 7. The IE-TRANS ET was quantified with SAPHIRE 8 using a truncation level of 1E-12. Table 17 compares the quantified CDF with original and new ETs. The original total IE-TRANS CDF is 1.073E-6/year and half-reduced to 5.795E-7/year with the new FTs. There are 16 non-zero CDF sequences out of a total of 145 INT-TRANS accident sequences (i.e., the sequence end state is core damage).

INT-TRANS:21-16 from ATWS scenarios is one of the most risk-significant sequences with a CDF reduced from 5.388E-7/year to 1.596E-7/year and contributes 27.5% of the CDF of improved INT-TRANS. In this sequence, RTS fails to trip the reactor; primary and secondary side depressurization are not successful. Core damage occurs as long-term cooling cannot be established.

INT-TRANS:20 from TRANS scenarios is another risk-significant sequence with a CDF reduced from 3.895E-7/year to 3.286E-7/year, contributing 56.7% of the CDF of improved INT-TRANS. In this sequence, RTS successfully trips the reactor, core damage occurs as neither auxiliary feedwater is not available nor HPI could provide makeup water to the reactor coolant system.

Table 17. Comparison of INT-TRANS ET quantification results.

Sequence	CDF			# of Cut Sets	
	Original ET	Improved ET	Δ CDF/ Original CDF	Original ET	Improved ET
INT-TRANS:21-16	5.388E-07	1.596E-07	-70.38%	51	38
INT-TRANS:20	3.895E-07	3.286E-07	-15.64%	1060	1047
INT-TRANS:21-14	7.262E-08	2.150E-08	-70.39%	49	18
INT-TRANS:02-02-09	5.830E-08	5.830E-08	0	1248	1248
INT-TRANS:19	8.132E-09	6.738E-09	-17.14%	282	236
INT-TRANS:02-03-09	2.731E-09	2.731E-09	0	387	387
INT-TRANS:02-02-10	9.546E-10	9.546E-10	0	168	168
INT-TRANS:21-15	7.568E-10	2.124E-10	-71.93%	102	29
INT-TRANS:02-04-10	5.865E-10	5.865E-10	0	142	142
INT-TRANS:02-14-10	1.994E-10	1.994E-10	0	81	81
INT-TRANS:02-03-10	7.653E-12	7.653E-12	0	4	4
INT-TRANS:02-09-09	7.558E-12	7.558E-12	0	4	4
INT-TRANS:02-06-09	7.558E-12	7.558E-12	0	4	4
INT-TRANS:02-08-09	7.558E-12	7.558E-12	0	4	4
INT-TRANS:02-07-09	2.287E-12	2.287E-12	0	2	2
INT-TRANS:02-10-09	2.287E-12	2.287E-12	0	2	2
Total	1.073E-6	5.795E-7	-45.99%	3590	3414

3.4.2 INT-SLOCA

The generic PRA model representing small-break LOCA as INT-SLOCA was shown in Figure 10. The INT-SLOCA ET was quantified with SAPHIRE 8 using a truncation level of 1E-12. Table 18 compares the quantified CDF with original and new ETs. The original total INT-SLOCA CDF is 7.784E-8/year and reduced to 7.512E-8/year with the new FTs. There are seven non-zero CDF sequences out of a total of 10 INT-SLOCA accident sequences (i.e., the sequence end state is core damage).

INT-SLOCA:03 is the most risk-significant sequences with a CDF of 6.433E-8/year and contributes 85.6% of the CDF of improved INT-SLOCA. In this sequence, RTS successfully trips the reactor, and the auxiliary feedwater and HPI are available. Core damage still occurs, as long-term low-pressure cooling cannot be established.

Table 18. Comparison of INT-SLOCA ET quantification results.

Sequence (with CDF > 1E-9)	**CDF**			**# of Cut Sets**	
	Original ET	**Improved ET**	**Δ CDF/ Original CDF**	**Original ET**	**Improved ET**
INT-SLOCA:03	6.433E-08	6.433E-08	0	564	564
INT-SLOCA:05	2.867E-09	2.867E-09	0	97	97
INT-SLOCA:09	7.386E-09	6.647E-09	- 10.01%	142	142
Total	7.784E-8	7.512E-8	- 3.49%	838	838

3.4.3 INT-MLOCA

The generic PRA model representing medium-break LOCA as INT-MLOCA was shown in Figure 11. The INT-MLOCA ET was quantified with SAPHIRE 8 using a truncation level of 1E-12. Table 19 compares the quantified CDF with original and new ETs. The original total INT-MLOCA CDF is 6.279E-7/year and reduced to 5.032E-7/year with the new FTs. There are eight non-zero CDF sequences out of a total of nine INT-MLOCA accident sequences (i.e., the sequence end state is core damage).

INT-MLOCA:03 is the most risk-significant sequences with a CDF of 4.917E-7/year and contributes 97.7% of the CDF of improved INT-MLOCA. In this sequence, RTS successfully trips the reactor, and the auxiliary feedwater and HPI are available. Core damage still occurs, as long-term low-pressure cooling cannot be established.

The most significant CDF reduction appears in INT-MLOCA:10 from 1.305E-07 to 7.348E-09, which contributes to 98.8% of Δ CDF. In this sequence, RTS successfully trips the reactor, and the auxiliary feedwater is available. Core damage still occurs as neither HPI nor LPI can be established for long-term cooling. According to the change of failure probability of LPI shown in Table 16, LPI availability was highly increased by adding a FT of a more reliable 4-division digital ESFAS instead of using a conservative CCF. Improved ET models can provide a more accurate prediction for ESFAS failure and relevant sequences.

Table 19. Comparison of INT-MLOCA ET quantification results.

Sequence (with CDF > 1E-9)	CDF			# of Cut Sets	
	Original ET	Improved ET	Δ CDF/ Original CDF	Original ET	Improved ET
INT-MLOCA:03	4.917E-07	4.917E-07	0	722	722
INT-MLOCA:05	1.870E-09	1.870E-09	0	47	47
INT-MLOCA:07	8.999E-11	8.999E-11	0	26	26
INT-MLOCA:09	1.866E-09	1.866E-09	0	192	192
INT-MLOCA:10	1.305E-07	7.348E-09	-94.37%	47	47
INT-MLOCA:11	5.206E-10	2.778E-11	-94.66%	3	3
INT-MLOCA:12	5.320E-10	2.263E-11	-95.75%	8	2
INT-MLOCA:14	8.576E-10	2.540E-10	-70.38%	5	4
Total	6.279E-7	5.032E-7	-19.86%	1050	1043

3.5 Summary of Consequence Analysis

To compare the changes of CDF after adding integrated FTs of digital RTS and ESFAS to the generic PWR ET models; in this chapter, consequence analysis has been performed based on INT-TRANS and relevant accident scenarios. Integrated FTs of RTS and ESFAS include both software and hardware failures, particularly CCF, that may occur in a 4-division digital RTS and a 4-divison digital ESFAS. Results show the CDF of INT-TRANS accident scenarios are reduced significantly.

The original generic PWR SAPHIRE model has 24 event trees; most of them use a top event for the failure of a 2-train RTS. Its detailed FT consists of hardware failures, operator errors, and external hazards. ESFAS failure is modeled using a basic event as CCF of ESF actuation signal, which is embedded in the FTs of relevant safety features including auxiliary feedwater, HPI, and LPI. This CCF basic event is included in cut sets that have significant contributions to CDF.

This generic PWR SAPHIRE model represents the conditions of existing U.S. NPPs with traditional analog HSSSR DI&C systems.

By adding the integrated FTs of 4-division digital RTS and ESFAS into the PRA models, the safety margin obtained from the plant digitalization on HSSSR DI&C systems are quantitatively estimated. For example, results show RTS failure probability is half-reduced from 4.288E-6 to 1.270E-6; LPI failure probability greatly decreases from 8.416E-4 to 2.258E-4 due to the improvement of ESFAS-FT. This explains the significant reduction of CDF in these analyzed accident scenarios, as summarized in Table 20. More accident scenarios will be analyzed in FY 2022.

Plant modernization including the improvement of HSSSR DI&C systems such as RTS and ESFAS will make great benefits to plant safety by providing more safety margins to accident management.

Table 20. Changes of ET CDFs by adding digital RTS and ESFAS FTs into ETs.

ETs	Original CDF	New CDF	Δ CDF	Δ CDF/ Original CDF
INT-TRANS	1.073E-6	5.795E-7	- 4.935E-7	- 46%
INT-SLOCA	7.784E-8	7.512E-8	- 2.720E-9	- 3.4%
INT-MLOCA	6.279E-7	5.032E-7	- 1.247E-7	- 20%

4. ORTHOGONAL-DEFECT CLASSIFICATION FOR ASSESSING SOFTWARE RELIABILITY

4.1 Introduction

For the past decade, growing interest in digital control systems for NPP has led to the development of numerous case studies advocating for digitization. Improvements in the plant monitoring and control systems have the potential for faster detection through real-time network monitoring, boosting the overall safety and reliability of the system (e.g., asset performance management [46]). New DI&C implemented in the Korean OPR-1000 reactors [47] have already been added to the design of the APR-1400 reactor [48]. However, with increased hardware digitalization and integration of advanced software, a considerable amount of risk associated with CCFs and software single failures are also introduced.

In previous work, RESHA was applied to analyze the possible software hazards/basic events as well as software CCFs that can occur through highly redundant systems. The process is an adaptation of the STPA and HAZCADS methodologies and is a system-level analysis technique. It involves first identifying the target underlying control structure and feedback responses between different hardware and software modules. From there, a hardware FT is created to represent the possible failures associated with the digital systems. STPA is then utilized to identify UCAs communicated between modules that can potentially lead to a hazard. The identified UCAs are integrated into the hardware FT as software basic events. Triggering events are also identified to provide context for each software basic event. Commonly identified UCAs across different redundant divisions are classified as potential software CCFs. The following work to the RESHA application is to quantify failures identified.

While no industry wide consensus has been reached regarding the best method to quantify software failures, numerous methods have been presented. These approaches range from the use of evidence supported BBN to test-based failure quantification through exhaustive testing. However, each method has specific drawbacks related to the unique development conditions of each software. The proprietary nature of software also makes failure quantification difficult from the perspective of external investigators due to lack of available operational data. BAHAMAS, a methodology previously developed by this group, presupposed insufficiency of data and instead analyzed the completeness of the software development life cycle (SDLC) by examining the human elements of the design process. By assigning success probabilistic values for each phase in the SDLC and linking the process via a BBN, the method was able to provide preliminary failure quantification estimates. However, when operational and testing data are available, a new more detailed method to analyze failure probabilities is needed.

In this chapter, a detailed method to quantify software hazards identified with the RESHA method is presented and a case study is examined in next chapter. The method presumes operational and testing data are readily available to the reviewer. The method referred as ORCAS leverages existing methodologies in a unique combination to provide software specific failure quantification.

In instances where sufficient operational and development data are available, ORCAS attempts to address how specific software defects relate to identified UCAs and affect the overall system reliability. In contrast to RESHA, which is a top-down system analysis method, ORCAS examines the specific quality assurance and developmental information during the design, implementation, and testing of base software modules. The method is a bottom-up analysis method in that individual software modules and the intra-relationship between them are closely scrutinized for defects. The specific defect information is then used to inform on the overall system reliability by combining determined module failure probabilities with the integrated fault tree from RESHA.

4.2 Technical Background

In this section, a brief literature review on existing hazard quantification methods for digital software is conducted to provide context. Software hazards are notoriously difficult to quantify as each methodology has significant drawbacks and limitations. Furthermore, each developer will have a specialized software development process and quality assurance measures rendering generic methods ineffective. While no solid consensus has been formed in industry, many different methods have been proposed to address software failure probability. In this paper, the methods discussed are software reliability growth models (SRGM), test-based analysis, and metric-based analysis. Bayesian belief networks, while popular for software reliability, are evidence-based analysis methods and do not directly quantify hazard likelihoods. Only BAHAMAS [20], previously developed by INL, is analyzed due to its relevancy to RESHA.

4.2.1 Software Reliability Growth Models

The most well-known quantification methodology is SRGM. These methods use historical test data to ascertain whether a particular piece of software has demonstrated an acceptable failure rate. As an input requirement when developing the historical test databases, either the times between successive failures or the number of failures during different test intervals must be known. The data is then used to fit model parameters specific to the SRGM to predict future failure rate and testing hours required to achieve it. Unfortunately, these parametric values are specific to each SDLC and are difficult to generalize. As a result, while hundreds of SRGMs exist, each with slightly different assumptions, no one SRGM can significantly outperform the other. In addition, the majority of SRGM methods also assume once a failure is detected, the software can be repaired perfectly and instantaneously. Over time, this leads to reliability growth of the software as more tests are conducted. This assumption is not always valid as software repairs will sometimes add additional defects. While different SRGMs models have been proposed to overcome shortcomings, such as different empirical formulas, failure rate over time, repair rate, etc. No one SRGM can cover all limitations. In IEEE Standard 1633 [49], SRGMs can be broadly categorized into three high-level categories: exponential Non-Homogeneous Poisson Process (NHPP), non-exponential NHPP, and Bayesian. In this paper, only exponential NHPP models are examined due to their popularity and simplicity.

SRGMs in the category of exponential NHPP assumes the failure rate decreases exponentially with time, and the failure rate is proportional to the number of current faults in the software. The failure rate also remains constant over the intervals between failure detections. Model parameters are fitted to the historical test database after sufficient information has been collected, and evaluation of performance is determined based on fit accuracy. Some examples of models that fall under this category include [50], [51], [52], and [53].

While exponential NHPP SRGMs have been utilized extensively by the industry, application in ultra-reliable software is limited. This is because safety software is inherently simplistic in design to mitigate potential failure modes resulting in long mean time between failures (MTBF). The lack of consecutive failure data causes the estimation of software reliability extremely sensitive with larger error bars in addition to requiring long testing making SRGMs resource intensive.

4.2.2 Bayesian Belief Networks

Bayesian belief networks are probabilistic graphical models that depict a set of random variable nodes and their conditional interdependencies in a directed acyclic graph. Here, “directed” defines the direction of impact a node can have on other nodes. The BBN’s direction is typically one-way such that a child node cannot impact the parent node. “Acyclic” forces the BBN to not have any closed paths or loops, where the transition of N nodes will result in returning to the start node. Connections between nodes, known as edges, signify dependencies between random variables. In a BBN, multiple end state child nodes can exist, each representing a possible result from the BBN. An end node exists where there are no

departing edges only arriving edges. By considering the joint distribution of all parent nodes to an end node, the probability of occurrence can be calculated. The popularity of BBNs is due to flexibility in disparate sources of information. Factors, such as expert decision, test results, supporting evidence, etc., can all be incorporated into the BBNs when determining the end node.

However, each supporting evidence node must have a probability distribution assigned to it. These distributions are known as prior distributions (or prior belief), and the shape and magnitude influence the probability of the final results. When prior distributions are not known, uninformed distributions (also known as uniform distributions) are used as an initial estimate. The chief limitation to this is poor or skewed selection of prior distributions will significantly alter the distribution of the end node making the uncertainty quantification vital for BBNs.

BAHAMAS is one technique utilizing BBNs previously developed by INL to quantify software reliability. It was developed specifically for the reliability analysis of DI&C systems by analyzing the completeness of the SDLC. In the analysis, one key assumption is that software failures result from human errors during the SDLC and operational and test data was not readily available. This limitation of acquirable data is realistic as many companies do not publicly share failure data. As such, THERP in combination with BBN was utilized to quantify the potential failures associated in the SDLC. Specifically, it looks at the various stages of development that may be required for a piece of software and assigns a probability value for completion and adequacy. These values can then be propagated through the various stages of the SDLC to estimate the overall software reliability of the system.

4.2.3 Test-Based Analysis

Test-based analysis is the most direct approach to failure rate quantification but is also the most resource-intensive approach. By creating an exhaustive regime of tests, the functionality of software is guaranteed through test verification. A test may comprise simply of an input to a function block and a verification the output matches known or intended behavior (e.g., unit test). Test-based approaches generally rely on an automated testing framework, such as TestMaster, to generate input sequences to the function block. An “oracle” is also required to confirm the output to that test is correct. The key limitation to testing lies in the testing environment and whether it can adequately reproduce the true operational environment. Tests conducted in idealized environments that are under-representative of realistic operational profiles can lead to under prediction of the failure rate. Furthermore, comprehensive testing of large programs can be overly labor intensive and marginally beneficial at a certain point. While SRGM were developed to address the second limitation, each software is case specific, and no generic method has been adopted. Some notable examples of test-based analysis include computer aided software testing (CAST), Lint, and Selenium.

4.2.4 Metric-Based Analysis

Metric-based analysis uses good programming practices and quality software features as a way to measure the potential reliability of DI&C systems. Characteristics deemed inherent to reliable DI&C systems, such as requirements traceability, are used as a surrogate to gauge the probability of failure-free operation of a piece of software. The methodology assumes highly reliable software will also demonstrate completeness or “high marks” in each software quality metric. Through both implicit and explicit methods during the analysis of the system, design defects in the system can be identified in the various stages of the application development. Once metrics are quantified, through a variety of different estimation techniques, the reliability of the software can be predicted. Although the regulatory review process does not yet consider metric-based analysis for the assessment of current DI&C systems, the principles behind the methodology are rooted in highly effective software development processes recommended by industry experts.

Multiple publications by the NRC have already investigated metric-based methods, specifically the Reliability Prediction System (RePS). In NUREG-0019 [54], a preliminary study on the types of metrics

industry experts believe to represent quality software was performed. From that analysis, 40 software engineering measurements were identified, and their respective capability at predicting software reliability systematically ranked. In NUREG/CR-6848 [55], a small-scale validation study was conducted on the personnel access control system (PACS), and later in NUREG/CR-7042 [56], a large-scale validation study was conducted on a typical reactor protection system (RPS). The overall conclusion from the studies is using key software engineering metrics as a tool to bridge measures and reliability should be incorporated in the analysis of safety critical software [56].

However, while 40 metrics were identified in the original paper [54], not all methods were shown to be applicable or even accurate at the prediction of software reliability. Correlation-based metrics that utilized historical data were shown to be the most inaccurate at predicting reliability. For example, a historical correlation-based metric used in each study was the Gaffney "bugs per line of code" estimate [57]. This estimate was inaccurate in reliability prediction and exhibited an inaccuracy factor on the order between 10 to 1000. This is attributed to the specificity of software development and is unique for each company, team, and objective and each software piece should be treated as individual cases. It was found the use of historical data to generalize correlation to new software is inherently error-prone and susceptible to large inaccuracy factors.

Other metrics not based on correlations, but rather the actual source code and documentation were shown to be much more robust when determining software reliability. Requirements traceability, for example, is a semantic-based defect tracking method that attempts to map the software requirements specifications (SRS) document to the actual source code. This method directly analyzes both the source code and developmental documents to pinpoint potential defects located in the code and is case specific. Coverage factor is another prime example for highly relevant software metrics. The method relies on determining the number of tests implemented at the different levels of software to ensure each requirement has been adequately assessed and meets acceptance criteria. The testing requirements are reminiscent to the test-based methods; however, they incorporate additional development information when formulating the test space and can detect requirement inadequacies during the measurement process. The inaccuracy factor for these two metrics was on the order of ten or less.

4.3 Methodology

In this section, the methodology to conduct software reliability quantification is presented. The method proposed is referenced as ORCAS. It is an extension of the RESHA methodology and attempts to quantify the probability of each basic event identified from the integrated FT. Whereas RESHA is a qualitative methodology to determine software failures and software based CCFs, ORCAS strives only to quantify identified software failures through unique combinations of strong existing methods.

4.3.1 Terminology

To support the methodology and for clarity, commonly used terms in this section are defined first. A defect or fault is defined as a deficiency in the development wording or coding that may result in an unexpected or undesirable result. The terms are used interchangeability but represent the same definition. Defects may occur either as a miswording problem in the requirements or design specification of the software or as incorrect implementation within coding. A defect can also be a missing element, either as a missing vital requirement of the software or as a missing code block that implements a function. The former is considered an implementation defect as while the functionality was incorporated into the design and code, the exact method is flawed, limited, or insufficient. The latter is considered a design defect as a vital feature which should have incorporated, either in the design or code, is not there and would require a formal design change to implement. While not all defects or faults lead to software failures, generally higher defect counts in the code lead to higher failure rates. In addition, defects come in all different flavors, ranging from superficial to mission critical. Superficial defects could include a spelling mistake on the display monitor of a control system while a mission critical defect could result in the miscalculation of core flow rate leading to core damage. In this report, superficial defects, for example

those not identified in the integrated FT, are ignored and assumed to not affect the overall reliability of the system. Ultimately, higher mission critical defects cause lower reliability rates of the software.

Reliability is defined as the probability of failure-free continuous operation over a specified period. Conversely, failure rate is the number of failures within the same specified period [58]. These terms are used interchangeably as both measures are directly correlated to each other. Continuous operation is specified here due to the difference in actuation systems and control systems. The former does not operate continuously but instead only in specific instances of demand; therefore, it does not have a continuous failure rate. Instead, a failure rate on demand is used to quantify the reliability of actuation systems. Control systems, on the other hand, operate at all times and will only fail when a specific set of conditions are met. The continuous failure rate is thus how long the control system will operate before it encounters a set of trigger conditions that will cause the system to fail. In this report, only the continuous failure rate and continuous reliability are used.

Failures and losses are events that cause a specified loss to the operator, whether economic, environmental, human, or structural. Economical losses can include spurious trip scenarios where capital is lost by the company due to unavailability or outages triggered by the software. Environmental losses encompass release of radioactive products into the atmosphere, ground, or water systems that exceed allowed regulation release limits. Human losses include injury or loss of life either directly or indirectly due to a software event. Lastly, structural failures or losses, include damage to the reactor vessel, system, building, or grounds. For example, core damage due to failure to detect and trip the reactor from the control software.

UCA, a term specified from STPA, is a control action to a module that under a particular context and environment will lead to a loss. Here, control actions are specified as an action a controller makes to the controlled process that changes its internal state in response to feedback from the process. For example, a pump adjusting the input power in response to increases or decreases in flow rate. An additional term, unsafe information flow (UIF), is also introduced to categorize unsafe feedback from the controlled process that changes a controller's state, but the controller itself does not inherently have control. For example, core monitoring software receiving faulty sensor data signaling a spurious event. In such case, the software does not have the ability to control the core sensors but can still fail if the necessary protections were not built into the program. In these instances, under the wrong conditions, both will result in a loss.

The context and environment that causes the loss is defined as a triggering event (or trigger). The triggering event, within software, could be a particular operational profile or input sequence that causes unexpected or undesirable software response. Inappropriate human operations of the software are also considered triggers—for example, incorrect set points to a comparator. The different expected operational conditions (i.e., startup, transient, and shutdown) are all possible triggering events of software failures. Environmental conditions, such as excessive heat, cooling, flooding, etc., are also possible triggers.

4.3.2 Overview

In this section, an overview to the ORCAS approach is presented (Figure 16). On the left hand, a generic SDLC is shown. Within all generic SDLCs, there are three key stages to development: software requirements formulation, software design, and software implementation and testing. While not all SDLCs may follow this procedure exactly, all SDLCs will cover each of these stages at least once. From there, the software product information is what is analyzed for reliability. Ideally, the software product information should include the source code, the SRS document, and the software design description (SDD) document. Additional documents, such as the systems test document (STD) if available, can also be used to analyze software reliability. Section 4.3.3 describes what information is required for this step.

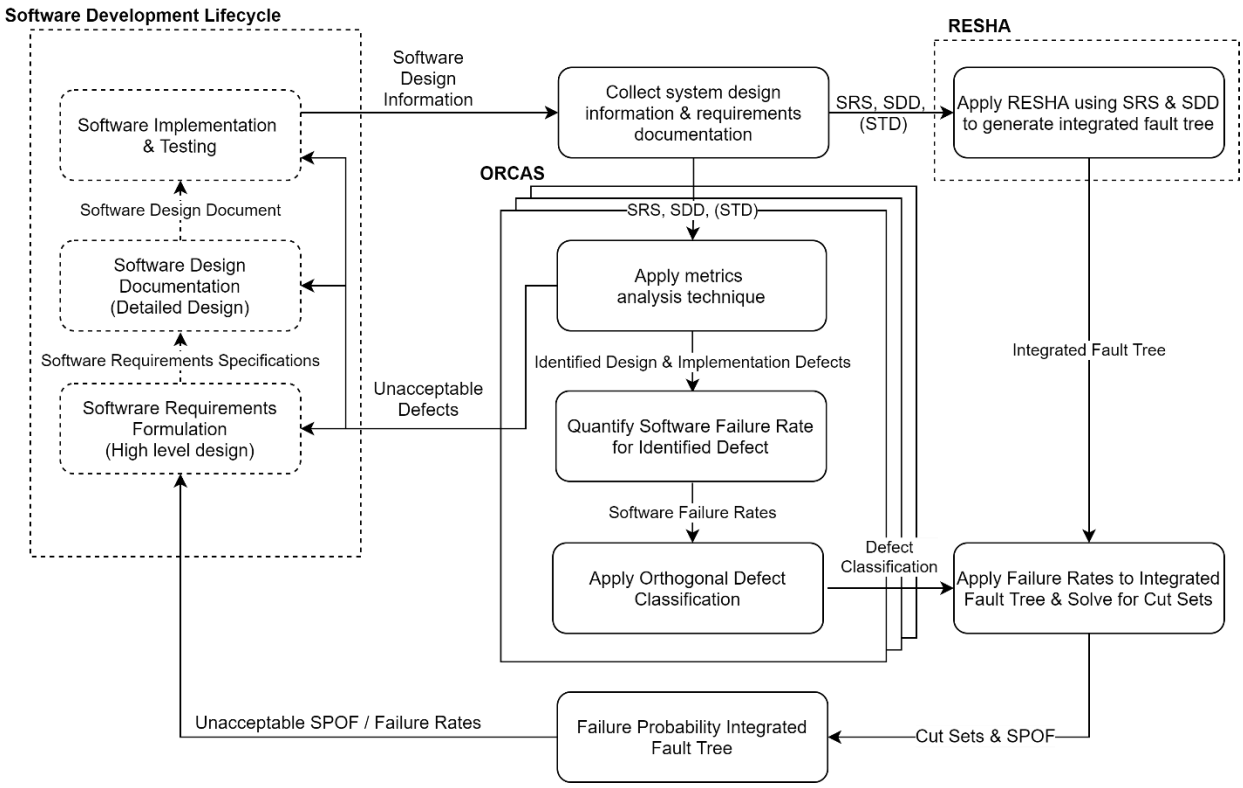

Figure 16. Overall workflow of software reliability quantification method ORCAS.

The SRS and SDD documents are used to conduct the first hazard analysis method, RESHA. From a top-down perspective, RESHA identifies qualitatively potential software hazards via STPA and HAZCADS. The top-down approach is able to identify key high-level failures between different component interactions. The output from the RESHA is the integrated FT which incorporates hardware, software, and potential software CCFs. While a top-down approach is meaningful in identifying hazards, it does not pinpoint defects correlated to identified hazards.

By supplementing RESHA with a bottom-up hazard analysis technique, defects can be identified and quantified for each software module of concern. In this case, the bottom-up analysis technique utilized in this report is an abbreviated version of the metric-based RePS [54]. Four metrics are utilized from RePS: requirements traceability, defect density, coverage factor, and test coverage. These metrics were selected due to the directness of analysis, examining source code and documents for defects as opposed to depending on historical correlations. The first two analysis techniques identify design defects in the development process and are aimed at finding requirement deficits. The latter two identify implementation defects in the source code and deficits in acceptance tests. Metric values are not used to quantify software failure probability (as is done in RePS), instead are only used to identify design and implementation defects. The outcome from the metric analysis is the design and implementation defects for each software module. Section 4.3.5 clarifies on how each metric is utilized to identify defects.

Post metric analysis of the software, unacceptable design and implementation defects may be discovered that are undesirable from the perspective of the developer. The developer may choose to return to the SDLC and use the information procured from ORCAS to enhance or improve the software design. If defects are undesirable but acceptable (i.e., cosmetic), the developers may also choose to continue the analysis to determine failure probabilities of individual modules and the overall system.

Once relevant defects have been determined, the connection to software basic events in the integrated FT can be made. While some defects will have clear and direct relationships to hazardous UCAs /UIFs

(and can be coupled directly), the difficulty lies in the linkage between defects that have unclear impact on overall software reliability. In such cases, a methodology developed by IBM called orthogonal defect classification is used to link the unclear defects to the identified software basic events within the integrated FT. The orthogonal defect classification (ODC) methodology systematically categorizes identified defects into distinct software impact groups. Each software impact group is then assumed to impact a set of UCAs/UIFs based on the semantic description of both the UCAs/UIFs and the ODC category. Section 4.3.8 clarifies the connection between UCA/UIF and ODC classes.

From there, the identified implementation defects are quantified through test-based analysis and operational data. This method produces the single failure rate. To ease exhaustive testing requirements, T-way testing is recommended as a reduced-order testing framework. The testing conditions and inputs act as triggering conditions for identified defects and quantify the probability of each failure occurrence. Section 4.3.6 clarifies on the methodology to develop T-way testing and fault probability.

For CCF rates, the UPM, described in Section 2.3.3 is used. The single failure rate is combined with a pre-determined beta factor to calculate the CCF rate. The beta factor is derived based on the software and hardware environmental and development conditions. Sections 2.3.3 and 4.3.7 clarify how to determine CCF rates and beta-factor values.

Lastly, after fault probability is quantified and classified, the values can be included into the integrated FT and unacceptable software failure probability as well as cut sets can be identified. During ORCAS, if at any stage an unacceptable defect (whether design or implementation) is detected, a return to the SDLC may be required to resolve defective software features.

4.3.3 Software Requirements Specification and Software Design Description

This is the most important step when determining software reliability. The collection of software requirements, through the SRS document, and the collection of implementation information, through the SDD document, is vital in determining the overall system architecture and potentially any latent design defects. The required information in the SRS include:

- Requirements specification for the target system
- Requirements specification for the application software
- Requirements specification for the system software
- Success criteria for algorithmic verification
- Success criteria for functional and non-functional verification
- Required target failure rates.

Within the SRS document, both functional and non-functional program requirements should be clearly outlined. Here, a functional requirement specifies something the software must be capable of performing. These requirements define the behavior of the system to adequately perform as expected and answers the question “what does the program need to do?” A non-functional requirement describes how the software must achieve the outlined goal—for example interface requirements, constraints, or performance requirements. These requirements attempt to answer, “how does the program need to do it?” In addition, each document should clearly define specific software-hardware interactions, constraints applied to the system, algorithmic simplifications, and software architecture.

4.3.4 Redundancy-guided Systems-theoretic Process Analysis

After the necessary requirements documentation is collected, the integrated FT can be build using the steps outlined in the RESHA process [2]. As a general overview, each step is briefly explained below to generate the integrated FT. Only Steps 1 through 5 are conducted, as Steps 6 and 7 are revised with the ORCAS software quantification methodology.

Step 1: Create a detailed hardware representation of the digital system

In this step, a physical and literal representation of the underlying hardware is constructed based on the source documentation collected previously. Specifically, the representation should identify information flow and feedback between separate modules.

Step 2: Develop hardware FT for the top event of interest

Based on detailed hardware representation, FTs can be developed to identify potential hardware and dependency failures. It is recommended for each event four different failure branches are included, namely the hardware stochastic failure, the hardware design failure, the dependency failure, and an unresolved software design failure branch (Figure 17). The first two failure branches describe the normal wear on hardware as well as potential failures in hardware design constraints. The dependency failure branch is allocated for when dependent signals from other components are missing causing a failure. The last branch describes failures associated with software implementation or design. Identification of software basic events and failure probability quantification is not required at this step.

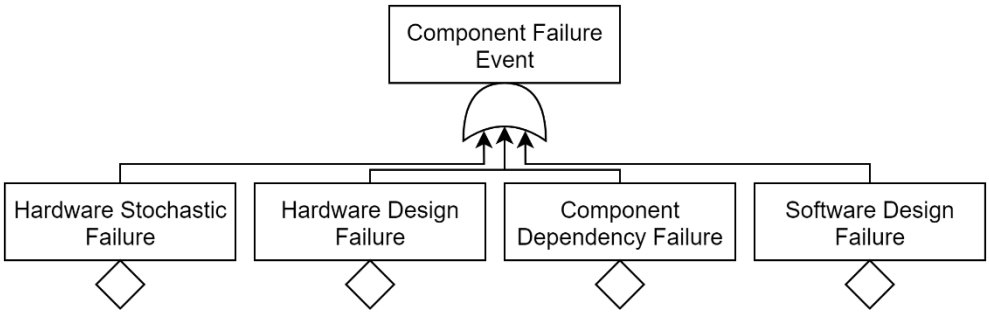

Figure 17. Generic component failure and recommended failure branches.

Step 3: Determine UCAs/UIFs Based on a Redundancy-guided Application of STPA

In this step, an expanded version of STPA is applied to identify potential software failures. In the original STPA, component interactions with other components leading to hazards are considered UCAs. Specifically, a UCA is only possible when a controller gives a command and receives feedback information from the dependent component that is a closed control loop (Figure 18). However, UIFs leading to unanticipated program faults that are not necessarily caused by CA are also possible. In the second scenario, a component provides fabricated or false data to another dependent component it relies on for decision-making. The dependent component processes the false data and outputs false signals, leading to a hazardous state. This results in two possible causes of software failures, either UCAs or UIFs. Thus, the revised STPA is used to identify these two types of events.

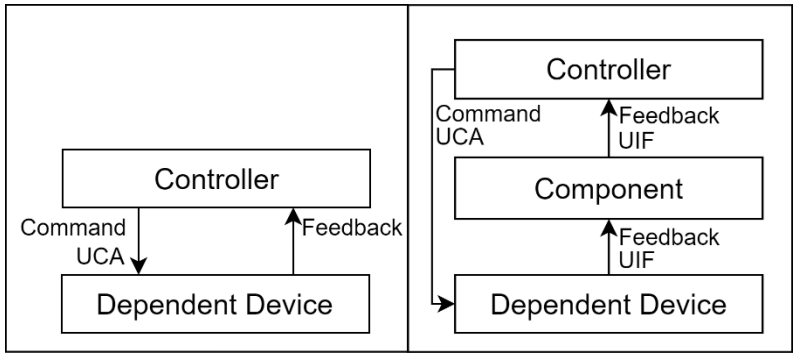

Figure 18. (Left) Original STPA control diagram. (Right) Revised STPA diagram for RESHA.

The first step in STPA is to define the purpose of the analysis and to identify losses, hazards, and the analysis scope. Defined losses are typical to any operational circumstance, while defined hazards are specific to the system.

Step 4: Construct an integrated ft by adding applicable UCAS/UIFS as basic events

In this step, based on the selected top event of the FT, relevant unsafe software actions are added into the hardware FT under the software design failure branch. The four UCAs identified in STPA including the following: control action is missing when needed, control action is provided when not needed, control action is provided too early, too late, or out of order, and control action is applied too long or stopped too soon. From these four categories, six subcategories are defined to better separate different scenarios. The re-classification serves only to specify what exactly the software failure is (Table 21). Multiple unsafe actions may also appear in more than one location in the FTs due to the interdependency of system components.

Table 21. Reclassified UCAs and UIFs from STPA.

STPA-defined Control Action	**RESHA-defined UCAs or UIFs**
CA is missing when needed	(A) CA/IF is missing when needed
CA is provided when not needed	(B) CA/IF is provided when not needed
CA is provided too early, too late, or out of order	(C) CA/IF is provided too early
	(D) CA/IF is provided too late
	(E) CA/IF is applied in the wrong order
CA is applied too long or stopped too soon	(F) CA/IF is applied too long or too much
	(G) CA/IF is stopped too early or applied too little

Step 5: Identify Software CCFs from Duplicate UCAs/UIFs within Integrated FT

Duplicate unsafe software actions that appear in the integrated FTs are considered software CCFs. For divisions with no hardware or software diversity, a CCF can appear in all components as the failure modes are the same and can triggered by the same event. Furthermore, depending on the level of redundancy (i.e., division vs. system), different CCFs are possible.

4.3.5 Metric-Based Analysis

To determine software failure rate, two vital perspectives must be incorporated into the analyze of the software. The first is the completeness and adequacy of software requirement design. As most software failures are caused by inadequacies in constraint enforcement, this is reflected in poor SRS and SDD documentation. For example, a requirement inadequacy could be where the input range for a software module considers only the normal operating conditions and does have extended ranges for recovery scenarios. The second is the correct translation of the SRS and SDD into source code. The “correctness” of the translation is conventionally accomplished through acceptance tests and benchmarking against experimental results of implemented algorithms as well as unit/integration tests between different software modules.

The four metrics selected to analyze both perspectives include semantic analysis of the SRS and SDD documentation as well as operational profile testing of the source code. This combination of software quality assurance offers the robust coverage required by highly reliable software. In the following sections, each metric is described in detail.

- **Requirements Traceability**

From IEEE guidelines [59], requirements traceability (RT) is a semantic analysis technique that identifies the number of implemented requirements to the total number of listed requirements. This aids in determining which requirements were not implemented in the design and implementation phases and checks for overall coding completeness. RT does not confirm whether requirement development is complete, rather only if the requirements were adequately met within the source code. This can also be used to inform developers areas of code inadequacies and directions for future efforts. In instances where the source code is not available (due to proprietary constraints), the SDD is assumed to be the source code implementation. As the SDD is meant to be the detailed design of the source code, requirements not explicitly implemented within the SDD are also assumed to not be present in the source code. Typically, RT is composed of both forward and backwards requirements tracing in the documentation and code.

Forward tracing is where the SRS and SDD documents are scrutinized and compared to the source code. Line by line tracing for which requirements are associated to which chunk of code should be conducted. Any requirement identified in the SRS and SDD document but not present in the code is tagged as a design defect. For example, if a module is required to acknowledge every 1 second a signal is received but is not implemented in code, it is considered a design defect and can introduce risk to overall software reliability.

Backward tracing is where the code is first examined line by line and associated with requirements outlined in the SRS and SDD documents. Additional code that is implemented but not identified in the documents is also considered a design defect as the introduction of undocumented code and can lead to instability and increase software risk. For example, an additional constraint implemented in code but not mentioned in the SRS can lead to specific case scenarios of failure. Additional unspecified requirements implemented are counted as “Requirements Implemented” in the numerator below.

In instances where the source code is not available, the SDD is compared against the SRS in both forward and backwards tracing. Identified design defects can also be further broken down into functional and non-functional requirement defects. The metric is measured using the following equation and is separated based on modules:

$$\text{RT\%} = \frac{\text{Requirements Implemented}}{\text{Requirements Listed}} \tag{18}$$

This metric is only used to determine which modules are insufficient in implementation and can be used to inform the software designers development direction. The metric is not used in the quantification of software failures. Ideally, a value of 100% suggests perfect requirements implementation. Less than 100% suggests some specified requirements were not implemented and greater than 100% suggests additional unspecified requirement implementations. Both under and over are undesirable and can inform which module(s) need additional development work. Additional attention should be noted if the number of extra requirements is equal to number of missing requirements as this would also result in unity.

- **Defect Density**

From RT, a known limitation is the inability to determine whether requirement specifications are complete. To compensate for this limitations, defect density (DD) is used to measure the effectiveness and completion of the inspection process. DD also relies on semantic analysis of documentation but can be the most subjective of the four metrics as it depends on independent inspectors to review completeness of documentation and implementation at each stage of the SDLC. The inspections conducted are manual examinations of the source code modules and documents to ensure compliance with both quality assurance policies as well as product specifications and requirements. Inspection is conducted at all stages of the SDLC (i.e., requirements, design, coding) with emphasis on tracking how detected defects are remedied in later stages of the development. The premise behind DD is that defects detected early in the SDLC but not remedied only amplify code instability due to hierarchical nature of code development

(e.g., object inheritance). The DD is thus the measure of the number of defects detected per line of code during each of the inspection stages. In ideal inspection processes of the SDLC, the value of DD should decrease over the stages of the project down to a minimum acceptable DD value (e.g., one defect per thousand lines of code). The minimum acceptable DD is defined based on the quality assurance policy adopted by the project.

In scenarios where the DD does not decrease over the SDLC, it is an indication the inspection process is inadequate in detail, or the project checkpoints are separated too far from each other. The development should be halted until the inspection process is corrected or the project checkpoints are redefined.

One limitation to the inspection process required by DD is that it can be ineffective for unstructured projects (e.g., research) or projects that lack defined checkpoints. Thus, for the inspection to return meaningful design defects, the project structure and the inspector must be committed to the inspection process. Closely managed projects with guided quality management processes (e.g., NQA-1) that has well-defined defect detection goals are ideal for the DD inspection process.

The DD inspection process generically contains seven steps: planning, overview, preparation, inspection, discussion, rework, and follow-up. In the planning stage, the entrance criteria for selection are first formulated. This specifies required qualifications for selected reviewers. The agenda as well as materials to be inspected is then created in advance and distributed to the reviewers once selected. The following plan is the overview process which updates inspectors that are not completely familiar with the goals and scope of the project or inspection. In the preparation phase, details pertaining to the inspection process should be reviewed by each inspector (e.g., quality compliance policy). The inspection phase is where the code and documentation are scrutinized for consistency and compliance. Each inspector should conduct the inspection independently and list all discovered defects in a report. After each inspector has completed reviewing the material, a discussion on potential defect fixes can be conducted among all inspectors and the project team. In the rework and follow-up phases, proposed fixes to the code are either addressed or re-designed, and a report is submitted to detail what changes were made to remedy which discovered defects. A detailed description of the review process can be found in [60].

The metric, derived from [56], is measured using the following equation and is separated based on modules:

$$DD = \frac{1}{KLOC} \sum_{j} (D_i - F_i - R_i)_j \tag{19}$$

Here D_i is the unique defects detected at the current development stage, F_i are the defects fixed up until the current stage, and R_i is the number of duplicate defects found by inspectors at the current stage. The subscript j indicates the inspector and the subscript i indicates the stage of development.

This metric is only used to determine which stage(s) of the SDLC does not meet inspection goals and require additional development. The metric is not used in the quantification of software failures, only the process is used to detect defects in the code and documentation.

- **Test Coverage**

From IEEE guidelines [49], test coverage (TC) is a measure of the completeness of the testing process. This aids in determining how many of the implemented requirements had a test associated with it and were adequately tested. By determining the thoroughness of testing and potentially inadequate results, TC can assist reviewers in identifying implementation defects in the source code. Furthermore, independent test validation is conducted to verify tests are adequate and meet goals outlined by the SRS and SDD. Implemented tests should cover different levels of software implementation; for example, unit and acceptance tests should be implemented for each software module to ensure algorithmic and

functional correctness of implementation. Between separate modules, integration tests should be implemented to ensure appropriate interactions. The determination of test success for algorithmic implementation is benchmarked against the success criteria listed in the SRS document which should include experimental results as reasonable justification. For the determination of test success against function or non-functional requirement tests, the success criteria for input-output sequences need to present in the SRS document. This analysis can only be adequately accomplished if the source code is available.

Two components are required to conduct TC analysis: an automated test generator and an "oracle" to verify output correctness. T-way combinatorial testing can be leveraged for this metric to reduce the number of tests required for comprehensive testing. Additional information of T-way combinatorial testing can be found in Section 4.3.6.

The metric is measured using the following equation and is separated based on modules:

$$\text{TC\%} = \frac{\text{Requirements Implemented}}{\text{Requirements Listed}} \cdot \frac{\text{Requirements Tested}}{\text{Requirements Listed}} \tag{20}$$

This metric is only used to determine which modules are insufficient in requirements testing and can be used to inform the software designers development direction. The metric is not used in the quantification of software failures. Lastly, for each test, the following testing conditions are collected for software failure quantification in Section 4.3.7.

1. Number of tests conducted per requirement
2. Average CPU time per test
3. Number of faults per requirement test
4. Description of fault
5. Input condition or acceptance criteria that caused fault.

- **Coverage Factor**

In contrast to TC, coverage factor evaluates the fault tolerance capabilities of the software system. The metric attempts to quantify the resiliency of the program given the fault exits as well as the ability to automatically recover back to normal operation. Fault injection is typically used to gauge the reliability of a fault-tolerant system. To conduct CF analysis, the code is first separated into its component software modules. Each module is treated as a black box, and the complete input space is injected into the module. The output of the module is then benchmarked against intended output. Modules that fail to recover are considered implementation defects and are later quantified.

Two components are required to conduct CF analysis, an automated test generator and an "oracle" to verify output correctness. T-way combinatorial testing can be leveraged for this metric to reduce the number of tests required for comprehensive testing. Additional information of T-way combinatorial testing can be found in Section 4.3.6.

The metric is measured using the following equation and is separated based on modules and divisions:

$$\text{CF\%} = \frac{\text{Number of faults recovered}}{\text{Number of faults injected}} \tag{21}$$

This metric is only used to determine which modules are insufficient in resiliency and can be used to inform software designers development direction. The metric is not used explicitly in the quantification of software failures.

Similar to TC, for each test, the following testing conditions are collected for software failure quantification in Section 4.3.7:

1. Number of tests conducted per requirement
2. Average CPU time per test
3. Number of faults per requirement test
4. Description of fault
5. Input condition or acceptance criteria that caused fault.

4.3.6 T-way Combinatorial Tests

It is well known exhaustive testing is impossible for safety critical software due to the insurmountable computational and human resources required. Previously discussed SRGMs, attempt to model and predict the trend of software reliability in the process of testing; however, fidelity of reliability predictions depends highly on the model used and still depend on rigorous testing. Combinatorial testing can alleviate the testing requirement and is a method to improve effectiveness of software testing in addition to reducing cost. Previous empirical data gathered by NIST [61] have suggested software failures are triggered through the interaction of six or fewer variables. Two-way interaction testing, or pairwise testing, is a type of combinatorial testing regularly used by industry as a way for highly effective fault detection. However, pairwise testing has been shown to only capture a fraction of the total possible software faults (40–60%), and a high order of interaction testing is required. In this report, five-way interaction testing was selected to be the minimum requirement for T-way testing to satisfy comprehensive fault detection for safety critical software (95–100% detection rate from empirical data).

Combinatorial testing specifically aims to test rare occurrence combinations of triggers to discover latent faults that may be hidden in obscure software paths. These faults may only be triggered by the interaction between two or more input variables and are typically implementation defects remaining in the final source code. While exhaustive testing may attempt to test every combination to discover these faults, for an application with more than five non-binary inputs, the number of tests required would grow exponentially large and be infeasible. Combinatorial testing instead generates unique input combinations of the total input space based on the specified interaction strength (e.g., pairwise would generate combinations of two inputs).

Not all input variables are binary which make the generation of test combinations difficult for continuous variables. This can be resolved by utilizing equivalence partitioning and boundary value analysis to separate the input space into meaningful testing partitions. Equivalence partitioning separates the input space into distinct valid and invalid range partitions and samples a small random subset of within each partition. Partitions are generated based on the input of a functional and non-functional requirement of a specific module. During the testing process, the entire partition is assumed to be valid if all values in the subset pass. Similarly, the entire partition is invalidated if any of the subset values fails a test. In comparison, boundary value analysis generates additional subsets to be tested based on the limitations of the software programming environment.

4.3.7 Failure Probability Quantification

- **Single Failure Probability**

To determine failure probability based on detected defects depends on how they were detected. For defects detected through CF and TC, the testing conditions are already known and can be calculated directly from reliability equations. Defects determined from semantic analyze such as RT and DD require additional processing to determine how these defects affect the overall system.

Recall that CF and TC defects are detected through the implementation of individual unit and integration tests. As the condition of each test is known, such as number of tests, duration of each test,

etc., the failure probability can be estimated with the following equations derived from Musa's exponential software reliability model [51].

$$\lambda_i = \frac{\text{Defect}}{\text{Total Testing Time}} = \frac{k_i}{\tau_{total}} \tag{22}$$

$$F(\tau) = 1 - \exp[-\int_0^{\tau} \lambda_i(t)dt\ ] \tag{23}$$

Here λ_i is the failure rate, k_i is the i^{th} defect, τ is the execution time, and $F(\tau)$ is the failure probability for a particular defect. The execution time is used as opposed to real time as computer processors may idle for multiple cycles during a test to load and store instructions and will produce false lower failure probabilities.

For semantic defect detections, the majority of defects detected are those that are either missing or extraneous. In such cases, it is difficult to extrapolate how unimplemented code may impact the overall reliability of the system. In this report, no comprehensive method was developed or discovered from literature to quantify design defects. This is because incorporating failure modes for hypothetical code functionality would require considering alternative code versions or branch conditions, neither of which can be strongly supported. Take, for example, the following design defect "algorithm shall detect coupling faults between two address lines." This functionality was not originally incorporated in the code; had it been incorporated, its impact on software reliability would be unknown and speculative at best. Furthermore, no test was developed in the test plan to verify these functions, making quantification speculative and difficult. However, based on the premise this requirement is explicitly mentioned in the SRS and SDD, the impact on software reliability is non-zero. To include some degree of impact consideration on reliability associated with design defects, it is assumed each design defect will have the same single failure probability as the largest implementation defect.

- **Common Cause Failure Probability**

The CCF probability is determined using the method previously described in BAHAMAS. To determine CCF probability, either the total or single failure probability must be known. These values can be derived using either BAHAMAS, in scenarios where operational data is insufficient, or ORCAS, when data is available. The beta-factor method is used to gauge similarity between two software/hardware components. Specially, a higher beta value suggests a higher likelihood the two components will fail simultaneously. Lower beta values are achieved through hardware/software diversity, separation, and a range of other factors. Different methods have been proposed to determine beta factor; however, in this report, the UPM is used [62].

The beta factor, from UPM, can be determined using knowledge of the operational and developmental conditions of the hardware/software platform. The information required to determine the beta include software/hardware diversity, physical separation, experience, historical reliability studies, existing tests and checks, safety culture, control access, and development tests [62]. Each knowledge category is given a rating between A to E, where A is the worst and E is the best. The scores are aggregated, and a normalizing constant is used to divide the result producing the beta factor. In Equation (24), the formula to determine beta factor is shown. S_i are the scores for each knowledge category, and d is the normalization constant. Additional details can be found in Section 2.3.3.

$$\beta = \frac{\sum_i S_i}{d} \tag{24}$$

4.3.8 Failure Categorization via Orthogonal Defect Classification

One issue routinely encountered in software quantification is how each detected defect affects each basic event in the integrated fault. Some defects may affect the overall operability of the program, and the

direct relationship is obscure. For example, a timing defect where a program fails to release a thread resource for another program definitely affects the program; however, how this may relate to the identified UCA is not entirely clear. In such instances, orthogonal defect classification is needed to group failures together before generalizing the impact towards overall reliability. Once defects are identified through the metric-based analysis, the defects can then be categorized based on how they affect the overall system functionality. ODC was originally developed in 1992 by Chillarege at the IBM Watson Research group [63] as a way to provide fast and meaningful feedback to developers through a semantic bridge between statistical defect models and defect analysis. By determining the number of defects in each category through the software development lifecycle, two pieces of information are revealed. The first being where (in which modules) the software needs additional support and debugging to meet performance goals. The second is the anticipated reliability based on how the number of defects in each class changes over the testing process. The method formally identifies eight different defect classes (Table 22) to detect software defects.

The derived classes are simple yet comprehensive such that any programmer should have no issue or contention categorizing detected defects. Furthermore, in each class, a further distinction between something missing and something incorrect can be made to assist determination of defect separation.

Each ODC category is applied strategically to specific ODC categories based on the particular failure event. For UCA/UIF A, the software fails to actuate when needed. This UCA/UIF has the most potential root causes, as any issue in the software may cause a failure of actuation. Issues with the embedded algorithm implementation may cause incorrectly calculated values leading to a failure to trigger the software. Issues in condition/branch validation or checking are also potential root causes, where an incorrect setpoint triggers the software at the wrong value. For controllers that depend on shared resources, such as threads, a failure to release the resource in time can restrict the controller's ability to function accordingly. Lastly, unimplemented functions specified in the SRS or SDD directly impact this UCA/UIF as well. A controller cannot send a signal if the required functionality is missing.

For UCA/UIF B, the software spuriously actuates when not needed. This implies the functionality for actuation was implemented (whether fully or partially). Due to this, function defects do not apply to this UCA/UIF. However, defects in the logic (or decision algorithm) will cause spurious actuation. Consider scenarios where the wrong equation or wrong set point are assigned to an algorithm. Furthermore, software defects related to checking the set point or internal states can also cause unnecessary actuation.

For UCA/UIF C, D, E, and F categories, the root causes are associated with timing, checking, or algorithmic issues with the software. The assumption here is the function has been implemented otherwise it would be considered a UCA/UIF A scenario. Due to this, function defects do not apply to these UCA/UIFs as adequate functionality is implied. However, an error in the internal algorithm (e.g., wrong equation) can still result in early or late actuation. Furthermore, defects in the checking of branch conditions, variables, or states can also lead to early or delayed actuation. If the controller is also dependent on timing related resources, such as sampling frequency, defects related to timing can also cause early or late actuation. The process to integrate each fault into the UCA can be determined based on Figure 19.

Table 22. ODC defect classes and descriptions.

Defect Class	**Description**
Function	Defect significantly affects capability, end-user features, product application programming interface (API) interface with hardware architecture, or global structure(s). Repairs would require a formal design change.
Assignment	Defect affects initialization of code blocks or data structures.
Interface	Defect corresponds to errors in the interaction with other components via drivers, call statements, etc.
Checking	Defect corresponds to failures in program logic, validation of data or values used, loop conditions, etc.
Timing/Serialization	Defect are related to the failure of sharing and management of real-time resources, threads, locks, etc.
Build/Package/Merge	Defects are related to mistakes in the library systems, management of changes, or version control.
Documentation	Defects are related to the mismatch between publication and maintenance notes to the software or errors in the documents themselves.
Algorithm	Defects correspond to the efficiency or correctness of the solution implemented and can be fixed by (re)implementing the algorithm or data structure without the need for a formal design change.

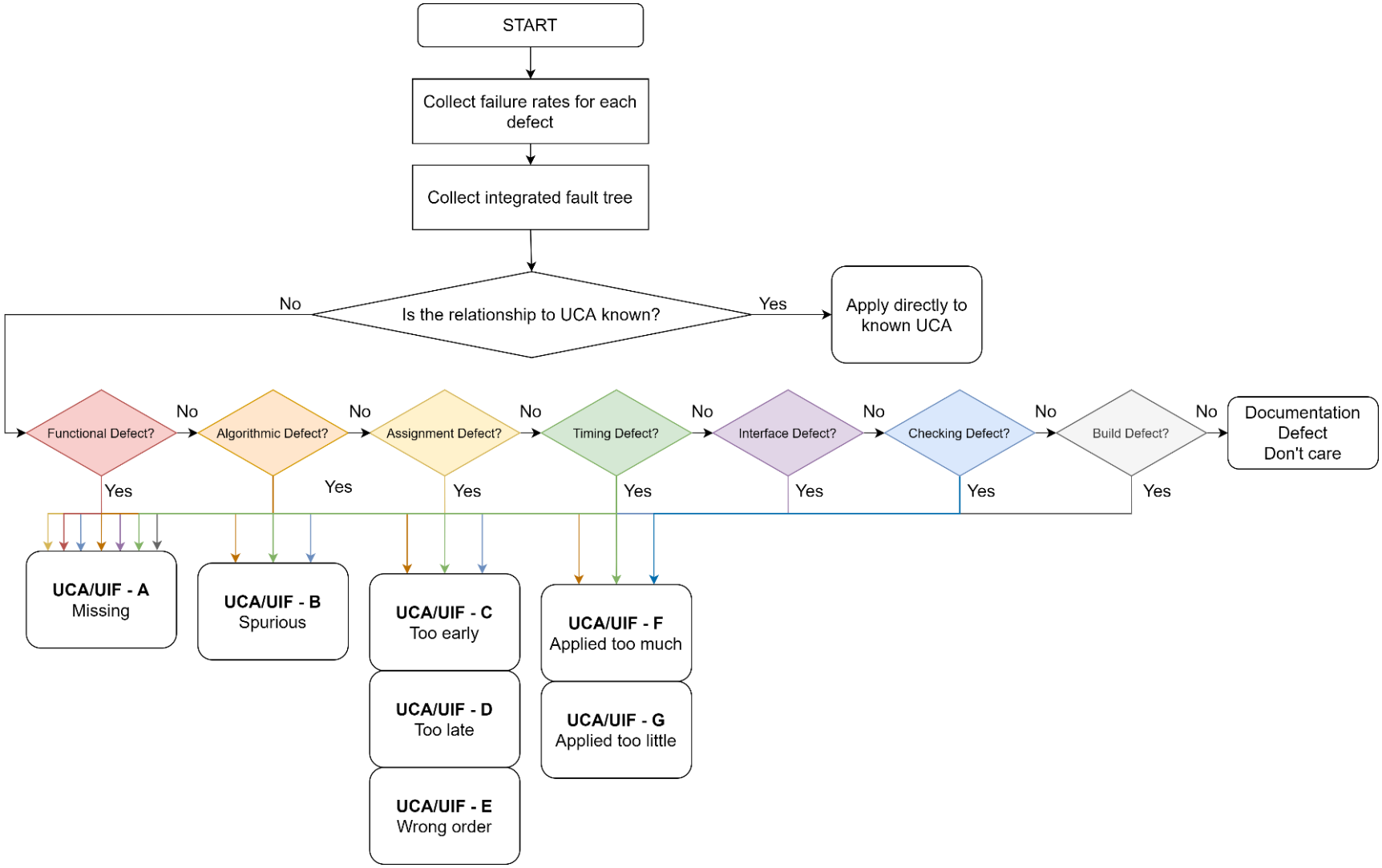

Figure 19. Flow chart for integrating detected faults with UCA/UIFs in an integrated fault tree.

4.3.9 Unacceptable Failure Probabilities from Integrated Fault Tree

Once the failure probability of each basic event is quantified and incorporated into the integrated FT, unacceptable failure probabilities can be identified, and design recommendations can be made. For example, if a certain program has a failure probability exceeding acceptance criteria, a separate implementation on different hardware may provide the acceptable diversity and protection.

4.4 Summary

In this chapter, a novel approach, ORCAS, was presented for the quantification of software hazards when sufficient operational and testing data available. The method incorporates elements of quality software development as well as strong analysis techniques to identify and link software defects to potential failure modes. The approach includes both semantic and test-based analysis to detect failures that can exist in different stages of the SDLC. In addition, considerations for comprehensive testing, yet computationally feasible, are also incorporated using T-way testing to generate valid, invalid, and edge case test conditions. Identified defects are then systematically linked and coupled with basic events via orthogonal defect classification. The semantic bridging presented by ODC allows flexibility but also solid guidelines in the coupling of root causes to software hazards.

5. INTEGRATED HAZARD AND RELIABILITY ANALYSIS OF DIGITAL HUMAN-SYSTEM INTERFACE RELEVANT TO REACTOR TRIP

In this chapter, the ORCAS method discussed in Chapter 4 is applied to the advanced HSI relevant to reactor trip safety developed from the APR-1400 HSI design. From documentation [46], the HSI of the APR-1400 consists primarily of four redundant information retrieval systems for the operator. During nominal reactor operation, the Qualified Indication and Alarm System-Non-Safety (QIAS-N) receives analog and digital signals from both safety and non-safety-related plant components. Under both nominal and safety relevant scenarios, the Qualified Indication and Alarm System-Safety (QIAS-P) acts as both a continuous source of accident monitoring information as well as a backup operator display module to the QIAS-N system [46]. The primary role of the QIAS-P system is to provide unambiguous indication of inadequate core cooling (ICC) as well as advanced warning of the approach towards it [46]. Both QIAS information systems are implemented using PLC-based control platforms. To introduce HSI diversity, an additional information processing system (IPS) also collects relevant sensor information and plant states. The IPS is implemented under a DCS-based platform, which is fundamentally different to the PLC-based platform. In speculated CCFs of either the QIAS or IPS systems, a last resort all analog Diverse Instrumentation System (DIS) is available that monitors the same key accident variables as the QIAS-P. This combination of analog and digital diversity in the HSI system ensures multiple information routes to the operator to safely shutdown the reactor under any condition. For the current work scope, only the QIAS-P system is analyzed using the adapted methodology.

5.1 Assumptions

In this subsection, the key assumptions during the development of the QIAS-P control architecture are highlighted, and the reasoning behind each are stated. Due to these assumptions, the developed control architecture from the APR-1400 DI&C manual [46] is only a representation of the true system and may not reflect real implementation or failures.

- **All set points are manually assigned, and alarms are comparators**

From documentation, there is explicit mention of alarms utilized in the QIAS-P system; however, the internal architecture is not specified. Therefore, it is assumed the most basic version of an alarm is implemented, that is, a comparator with a single set point trigger. The comparator only triggers when the set point is exceeded. These alarms set points are assumed to be manually assigned by the operator.

- **Internal architecture is approximated**

The APR-1400 HSI architecture was pieced together from multiple available NRC documentation and publications as the original verified schematic is proprietary. Therefore, the constructed diagrams included in this report are approximate representations based on the described connections and functionality. Some of the failures included in the integrated FT may not be reflected in the real system. In addition, results derived from the RESHA method are meant for application demonstration and are not meant to be real guidelines for the HSI system of the APR-1400.

- **Sensors act and fail in unison**

The QIAS-P system receives 61 core exit thermocouples (CET) and 34 heated-junction thermocouples (HJTC) sensors directly from the core. These sensors are assumed to operate as a single unit, and a failure of one sensor results in the failure of the entire array. This is to avoid partial failure modes and unsupported speculation on architecture response as the behavior and handling of the sensor information received from the QIAS-P system lacks detailed documentation.

- **Distributed network architecture is applied**

From the APR-1400 documentation, the QIAS systems are implemented with a network of PLCs that communicate to other modules via the safety data network (SDN). However, it is not explicitly mentioned how the QIAS systems communicates with the SDN. Typical industrial networks operate via distributed node-based communication (e.g., supervisory control and data acquisition) as opposed to centralized server-based communication. The QIAS-P representation will be assumed to be node-based as well, and each individual hardware module will contain a communications module connected directly to the SDN.

- **Hardware design failures are not considered in this case study**

This assumption applies to the hardware design choices made during the development of the digital and analog proportions of the QIAS-P. It includes, but is not limited to, which power supply to use, integrated chip sets, connectors, circuits, etc. Due to the unavailability of a detailed and proprietary hardware schematic, the hardware design failure branch is excluded from the FT and will not be shown.

- **QIAS-P redundant division architecture is not diverse**

While typical development practices are to introduce diverse design in redundant divisions of the QIAS-P to prevent CCFs, it is not explicitly mentioned in the documentation that diverse design was considered. Instead, system diversity is accomplished by designing the QIAS-N, IPS, and DIS differently. While it is possible hardware and software diversity were considered in the implementation of the QIAS-P, in this paper, we assumed the two redundant divisions are identical.

- **Operator-decision models are non-complex and based solely on data**

Due to the complex nature of human decision-making, misinformation propagated by the QIAS-P system to the operator does not guarantee failure. In fact, an additional separate human reliability analysis is required to survey the impact on falsified information on the operator-decision model. However, in this case study, it is assumed that failures by the QIAS-P system to communicate correct information or trigger the correct alarm will cause an immediate failure in the operator-decision model leading to a failing event.

- **QIAS-P has only a single display monitor in main control room for operator**

In normal control rooms, multiple display monitors are available to view all plant parameters. The QIAS-P system, from documentation also has a dedicated flat panel display to view key safety-related parameters. While realistically, the number of dedicated flat panel display should be greater than one, the exact number is not specified and will be assumed to be one.

- **QIAS-P digital components have similar hardware architecture**

For parameter calculators and alarms, it is assumed the same generic digital PLC is used. This implies the hardware failure probability for each PLC system is the same. For sensors, such as the HJTC and CET, it is also assumed the hardware is the same.

5.2 Software Requirements Specification and Software Design Description

In this case study, the system design documents were collected from a variety of publicly available sources on the APR-1400. The main resource which this case study is based around is the “Chapter 7 Instrumentation and Controls” document [46]. Additional publicly available sources used to reconstruct the system architecture include [47], [48], [64], [65], and [66]. However, due to the proprietary nature of the reactor design documents, only a representation of the true HSI system is portrayed in this analysis based solely on what is included and not included in the Chapter 7 documentation.

Furthermore, while the Chapter 7 document provides extensive information on the high-level design and objectives of the APR-1400 HSI control system, the document lacks the specific requirements, constraints, and design information required by this specific step. None of the SRS or SDD document

requirements listed in Section 4.3 were provided here. Due to this explicit unavailability of relevant information, an alternate software application RTS is considered, referenced as APP. The APP architecture was extensively analyzed in NUREG/CR-7042 [56]. In the NUREG report, the team conducting the analysis had direct access to the original SRS and SDD documentation, later document versions, as well as source code. The documents utilized in NUREG/CR-7042 are not publicly available for usage in this report, however the results drawn can be used as a surrogate in this report. The analysis conducted in the NUREG report are based on typical safety-related digital control modules that have been used in a NPP in prior decades. The modules contain both discrete and continuous, high-level analog and digital input and output circuits used to provide trip actuation, monitoring, alarm, and indication. Due to the similarity in functionality to the APR-1400 HSI system, the NUREG report can be used to represent potential failures modes of the QIAS-P system. The SRS and SDD documents used in the NUREG report cover all requirements listed in Section 4.3.

Specifically, the scope of the system analyzed by NUREG/CR-7042 includes two separate flux/delta flux/flow micro-processors (labelled as μp1 and μp2) and one communication module (CP). In addition, the μp have different software and hardware versions and operate in parallel with each other as redundant divisions. This parallels well with the two redundant divisions in QIAS-P as well as the micro-processors used to monitor in-core sensors, such as the heated-junction thermocouple. A complete list of the relevant requirements and design documents analyzed in NUREG/CR-7042 are shown below:

1. APP Instruction Manual
2. APP Module-Design Specification
3. APP Design Requirements
4. APP Module μp1 System SRS
5. APP Module μp1 System SDD
6. APP Module µp1 System Software Code
7. APP Module µp1 Flux/Delta Flux/Flow Application SRS
8. APP Module µp1 Flux/Delta Flux/Flow Application SDD
9. APP Module µp1 Flux/Delta Flux/Flow Application Software Code
10. APP Module µp2 System SRS
11. APP Module µp2 System SDD
12. APP Module µp2 System Software Code
13. APP Module µp2 Flux/Delta Flux/Flow Application SRS
14. APP Module µp2 Flux/Delta Flux/Flow Application SDD
15. APP Module µp2 Flux/Delta Flux/Flow Application Software Code
16. APP Module Communication Processor SRS
17. APP Module Communication Processor SDD
18. APP Module Communication Processor Software Code
19. APP Module Software Verification & Validation Plan
20. Final V&V Report for APP Module Software
21. APP Test Plan for μp1
22. APP Test Plan for μp2

23. APP Test Plan for Communication Processor
24. Test Summary Report for μp1
25. Test Summary Report for μp2
26. Test Summary Report for Communication Processor.

The extensive list of documents reviewed in NUREG/CR-7042 provides an excellent example of the level of detail required by this step in the collection of relevant information to the system architecture. The results from the NUREG/CR-7042 are used specifically in Section 5.4 and 5.6.1. All other sections rely on the APR-1400 Chapter 7 documentation.

5.3 Redundancy-guided Systems-theoretic Process Analysis

After the requirements and design documents are adequately collected, RESHA can be applied to develop the integrated FT.

5.3.1 Step 1: Create Detailed Hardware Representation of Digital System of Interest

From the Chapter 7 documentation, the hardware flow diagram is first re-constructed. For this report, the scope is only to the QIAS-P system. Peripherals (e.g., QIAS-N and secondary sensors) are not directly monitored and used by the system are not included.

The QIAS-P monitoring system is composed of two redundant divisions (A and B) isolated both physically in location and in communication from each other. Within each division, there is a digital processing module (PM) and an analog retrieval module (AM). The two modules operate sequentially to retrieve, process, and check values and alarms sent to the operator information terminal in the MCR.

The primary purpose of the PM is to check and determine intermediate values for an operator display and use in the internal alarm system. From documentation, the PM is composed of five primary parameter calculators and alarms, namely the HJTC, reactor vessel level, reactor coolant saturation margin, ICC, and the CET temperature. The PM also includes an HJTC heater power controller, a maintenance/interface test panel, and various secondary parameter calculators and alarms. One of the subroutines of the PM is to modulate the HJTC reference power level based on a power set point. This reference level is used to calculate the heat differential across the HJTC sensing junction. A manual override of the HJTC power controller is also available to switch control to the DIS in a postulated CCF.

The primary purpose of the AM is to retrieve signals from the various sensors and subsystems and either convert them from analog to digital (via analog to digital converter) or to repeat or convert the signal (via signal conditioners). In total, the AM module receives 32 HJTC and 61 CET sensor values directly from the core. In addition, it also receives the hot and cold leg temperatures, pressurizer pressure, and reactor vessel head as well as Type A, B, and C variables from the SDN or auxiliary processing cabinet. The different types of sensors (A, B, and C) denote which plant parameter is measured and how it impacts overall plant logistics. For example, Type A sensor information are key vital monitoring parameters directly relevant to reactor trip. The QIAS-P system collects all Type A sensors. How the Type A sensory information is used in the QIAS-P is not specified and are thus considered peripheral (for analysis purposes). To ensure sensor diversity, half of the analog HJTC and CET values are also routed to the DIS.

In Figure 20, a single division of the QIAS-P system is shown. Each of the primary parameter calculators and alarms are modeled as individual hardware components for a total of ten separate independent modules. Each parameter calculator provides real-time monitoring information to the main control room and to the alarms. When an alarm set point is exceeded, the alarm modules will “actuate” and will alert the main control room of the problem.

To condense the information from the hardware schematic, a flow diagram was developed to illustrate the communication between higher level modules and the feedback to the RTS. In Figure 21, the two divisions of the QIAS-P system are shown operating in parallel to each other. The type of information or control flowing between the various components are also shown. The unanalyzed components include the RTS redundant branch A and B, PPS primary and auxiliary processing cabinets, ESF-CCF primary and auxiliary processing cabinets, the QIAS-N operator's module, and the IPS operator’s module.

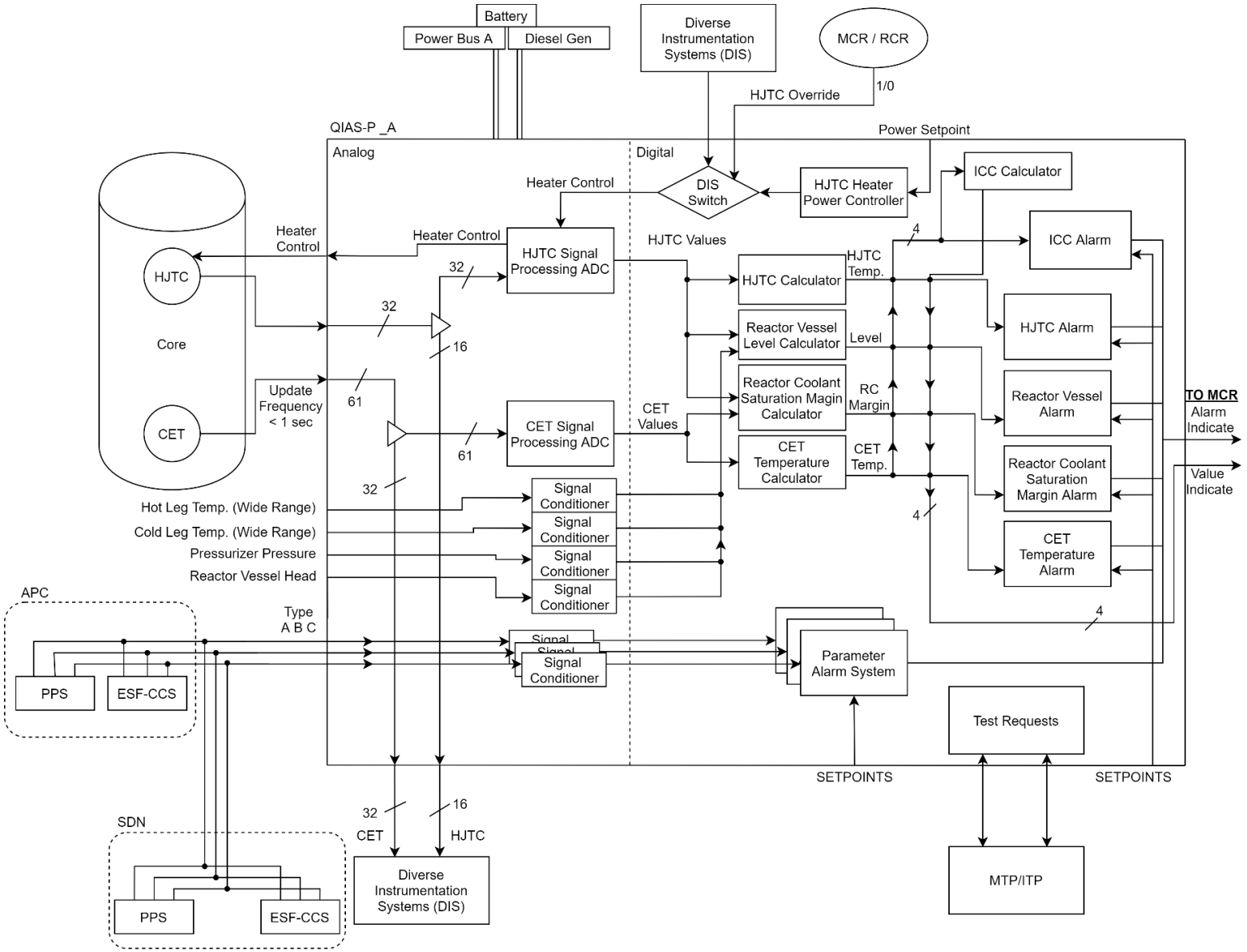

Figure 20. Division A of the QIAS-P system with component and information flow [46].

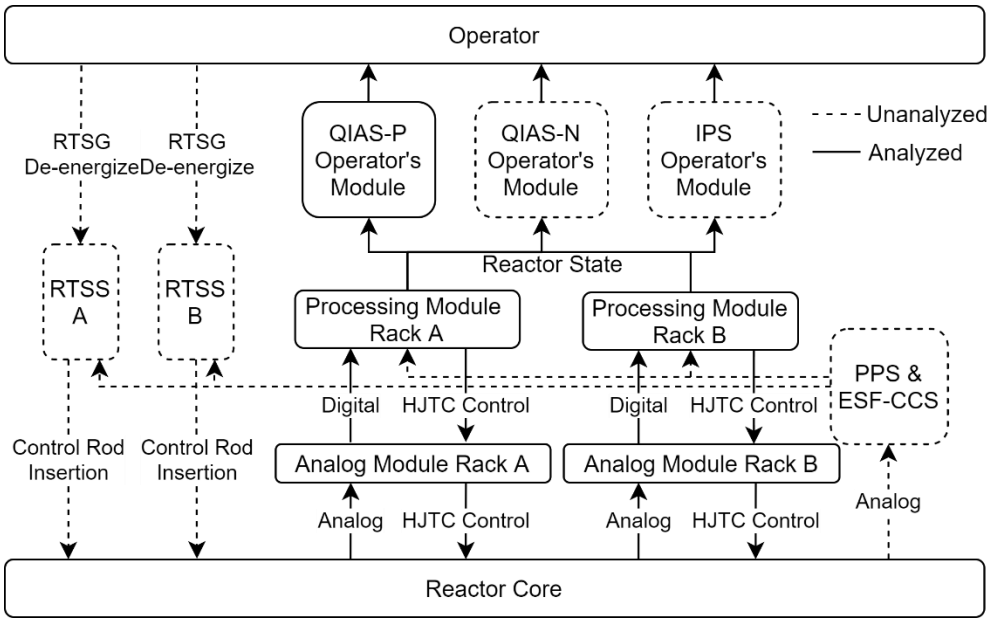

Figure 21. Condensed qualified indication and alarm control system-safety (QIAS-P) flow diagram [46].

5.3.2 Step 2: Develop Hardware FT for Top Event of Interest in Digital System

By using the detailed hardware model developed in Stage 1 and the guidelines from Figure 17, the full hardware FT can be developed. The top event in the FT is defined as the "Operator fails to initiate reactor trip causing reactor damage." Based on this top event, the full FT includes 41 hardware stochastic failure basic events and 33 component dependency failure branches. There are also 26 unresolved software design failure branches. A system-level hardware FTs is developed and shown in Figure 22 as an example. System-level redundancies (i.e., IPS, QIAS-N, and DIS) are identified but unanalyzed.

Values for PLC and sensors failure are derived from [65], [67], and [68]. In Table 23, the values for hardware failure are presented below and conform to the assumptions listed in Section 5.1.

Table 23. Hardware total failure probability for QIAS-P digital components.

Hardware Name	**Failure Probability**
Heated-junction thermocouple sensor	1.05E-07
Heated-junction thermocouple sensor controller	2.21E-06
Core exit thermocouple	1.05E-07
Signal conditioner	1.00E-06
Analog to digital converter	7.13E-06
Parameter calculator	2.21E-06
Parameter alarm	2.21E-06

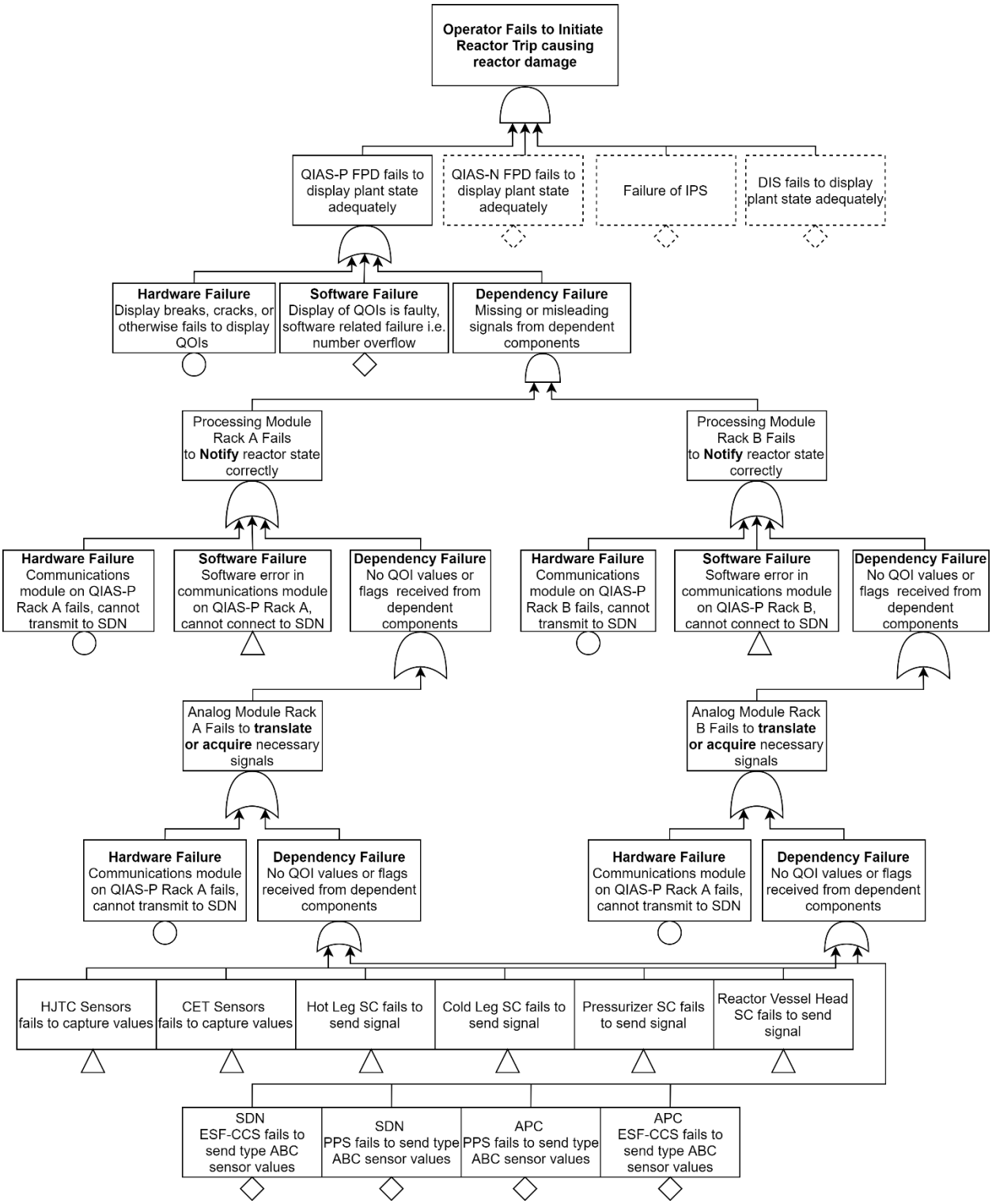

Figure 22. QIAS-P system-level hardware FT with empty software failure branches.

5.3.3 Step 3: Determine UCAs/UIFs Based on RESHA

From the control diagram (Figure 20), three UCAs were identified related to the HJTC controller, 15 UIFs were identified related to the digital parameter calculators, and 10 UIFs were identified related to

the digital alarm systems. Examining the HJTC controller, the UCAs involved are the controller fails to provide a power reference level to the HJTC sensors when needed (Type A UCA), and the controller provides a reference level but is either too high (Type F UCA) or too low (Type G UCA). For parameter calculators, the prominent UIFs are the calculator fails to output a value when needed (e.g., zero, null, and inf) (Type A UIF) and the value calculated is either too high (Type F UIF) or too low (Type G UIF). For alarms, based on the assumption from Section 5.1, there are only two UIFs per alarm, either it fails to trigger when needed (Type A UIF) or triggers when not needed (Type B UIF). It should be noted all calculators and alarms have the same software basic events as the specific functionality of each component is unavailable. (e.g., ICC calculator and HJTC calculator).

5.3.4 Step 4: Construct an Integrated FT by Adding Applicable UCAs/UIFs as Basic Events

In this stage, based on the selected top event of the FT, relevant unsafe software actions are added into the hardware FT under the software design failure branch. The fully integrated FT includes all UCAs/UIFs identified in Stage 3 can be seen in a separate document as the full SAPHIRE model. (please contact Han Bao han.bao@inl.gov for more details about this FT). A partial integrated FT can be seen in Figure 23 and describes Division A ICC alarm, ICC calculator, HJTC sensor, and HJTC power controller.

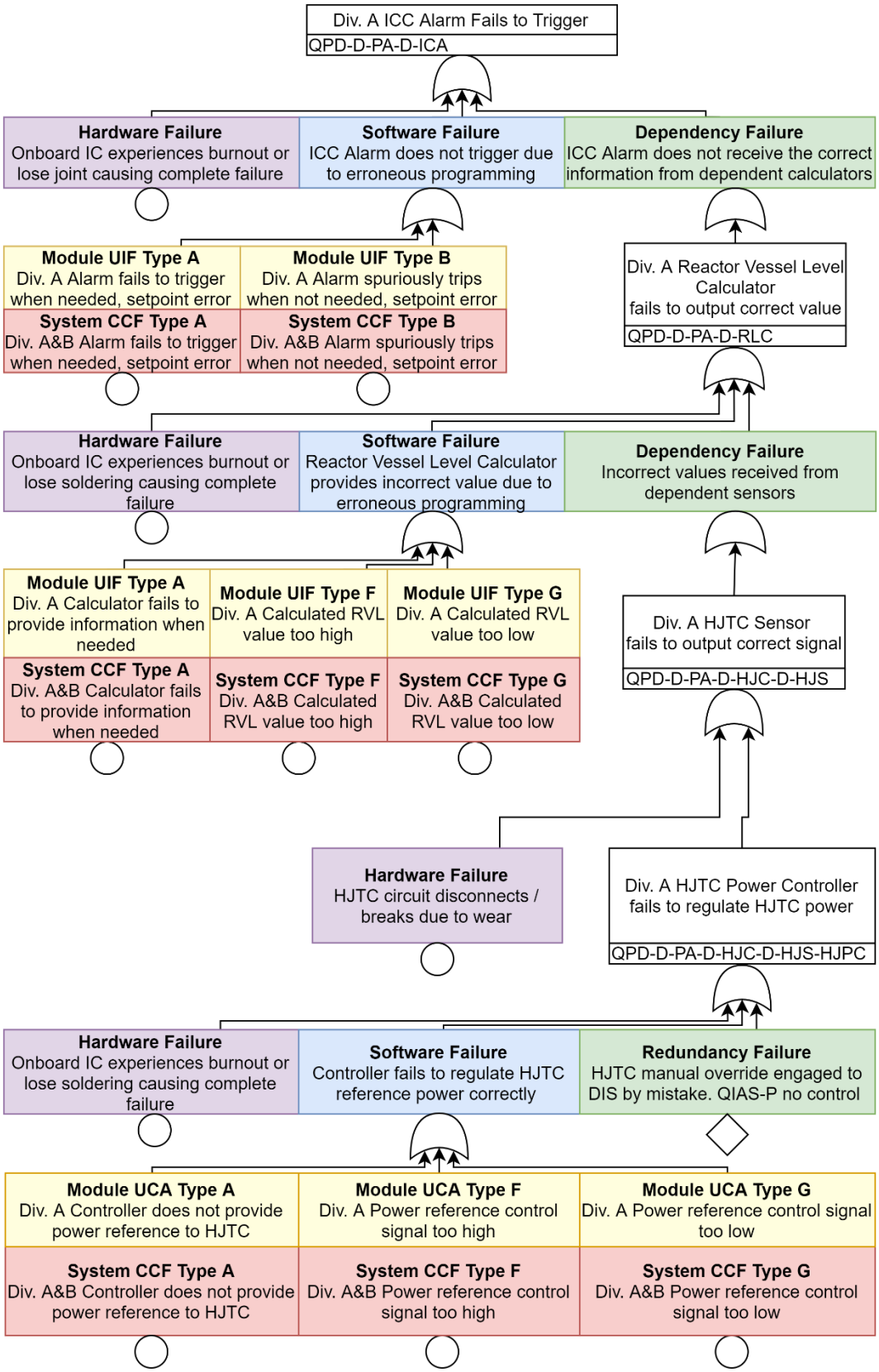

Figure 23. Partial integrated FT from Division A of the QIAS-P.

5.3.5 Step 5: Identify Software CCFs from Duplicate UCAs/UIFs within Integrated FT

At the QIAS-P system level, the lack of software diversity results in 28 software CCFs affecting all digital components. Each parameter calculator has three CCFs associated with UIF A, F, and G. Any software failure in the calculator has the possibility to trigger both Division A and Division B UIFs and is thus considered a Type 4 CCF (fifteen calculator related CCFs). For example, a logic defect in the software can be guaranteed to manifest a parameter reading error in both division calculators. This is because the input conditions that triggers a defect in Division A calculator will also be experienced by the Division B calculator. This applies to each division alarm as well, each with two CCFs associated with UIF A and B. For the five alarm modules, this corresponds to ten Type 4 CCFs. Lastly, for the single HJTC controller, three CCFs exist associated with UCA A, F, and G (3 controller related CCFs). The route cause behind all Type 4 system-level software CCFs is the explicit lack of diversity.

5.4 Metric-Based Analysis

In this step, the metrics selected from the RePS methodology [56] are applied to the target system documents and source code. The metrics applied are the RT, DD, TC, and coverage factor (CF). As mentioned previously, the unavailability of specific data prevented detailed analyze of the APR-1400 HSI system. Instead, data from NUREG/CR-7042 on the RTS analysis is used as a surrogate.

There are two primary components analyzed in NUREG/CR-7042, the microprocessor (μp) and the communication module. The functionality of the μp is to process and collect sensor information and conducts internal calculation. This is similar to the sensors, signal conditioners, calculators, and HJTC controller in the QIAS-P system. For the communication processor, its role is to handle alarm notification and parameter monitoring to the main control room. While no individual component represents the CP in the APR-1400 HSI system, it is assumed each parameter calculator and alarm system has an internal communication processor. Therefore, the defects discovered for the APP modules will be applied to all corresponding APR-1400 HSI components.

5.4.1 Requirements Traceability

From NUREG/CR-7042, comprehensive RT was conducted for the APP code and related documentation. The documents reviewed include the SRS, SDD, and source code relevant to RTS actuation and monitoring systems, specifically, micro-processors (μp) vital to monitoring and communication. In total, of the two micro-processors and communication module described in the system, there were seven discovered defects. The system was also analyzed across different operational profiles (i.e., initialization, power-up, main program, and diagnostics). Within the different operating profile, only the power-up self-test and calibration did not meet 100% traceability and suggests potential development effort needed for future versions. Specifically, only 343 of 348 requirements were achieved translating to an average RT of 98.56%. The discovered defects for this operational profile can be seen in Table 24.

Table 24. RT Defect tag, description, and location.

Tag	Location	Requirement Description	Defect Description
D1	μp1	Increment the EEPROM test counter if the Tuning in Progress flag set	Requirement not implemented in code
D2	μp1	Algorithm shall detect coupling faults between two address lines	Requirement not implemented in code
D3	μp1	Copy contents of table to Dual Port RAM	Requirement not implemented in code
D4	μp1	Give up semaphore	Requirement not implemented in code
D5	μp2	Algorithm shall detect coupling faults between two address lines	Requirement not implemented in code
D6	CP	Algorithm shall detect coupling faults between two address lines	Requirement not implemented in code
X1	μp2	(Additional code)	Specified code block implemented but not documented

For all defects where the requirement was specified but not implemented in code, these are considered design defects and given a tag D#. For the defect where additional functionality was implemented, but the requirement was not specified, this is still considered a design defect but is given a tag X# for excessive.

5.4.2 Defect Density

For the DD metric, the SRS, SDD, and source code were inspected with two inspectors and one moderator. Each inspector operated independently and recorded defects using the procedure described in Section 4.3. The moderator then reviewed all defects in the code stage and corrected code mistakes discovered during inspection. Defects that could not be corrected as the source of the defect was unknown are listed in Table 25. In total, four defects were discovered related to design and implementation inadequacy specified in the SRS or SDD.

From the defects discovered, D8 and D9 were due to ambiguity in requirement specification and thus not implemented fully. D7 and D10 were specified but not implemented in code. All defects are considered design defects as they are all missing requirements. Regarding the density of defects, of the 4825 lines of code reviewed, four defects were discovered for a value of 0.829 defects/KLOC.

5.4.3 Test Coverage

For TC, the test plan for each APP component was examined and traced to a corresponding requirement from the SRS and SDD. In addition, separate verification tests were conducted on each module. Only defects that required modification to the SRS, SDD, or code are listed. In total seven defects were discovered through the independent test validation of the components (Table 26).

Table 25. DD Defect tags, description, and location.

Tag	Location	Requirement Description	Defect Description
D7	μp1	Check cannot detect coupling failure but only stuck at high or low failures.	Requirement not implemented in code
D8	μp2	If trip condition is calculated, the logic will force a different incorrect calculation for the second trip calculation.	Requirement ambiguity
D9	μp2	Processor has 16-bit address, but only 13 bits are examined. Remaining three bits are not tested and may cause error.	Requirement ambiguity
D10	CP	Check cannot detect coupling failure but only stuck at high or low failures.	Requirement not implemented in code

Table 26. TC Defect tag, description, and location.

Tag	Location	Requirement Description	Defect Description
I1	μp1	Check does not cover all address lines	Requirement implementation error
I2	μp1	During Power-On, processors cannot detect missing inputs and does not indicate fatal error	Requirement implementation error
I3	μp1	Inputs changed from continuous to discrete caused spurious trip	Requirement implementation error
I4	μp1	Dual port ram detects the wrong module ID	Requirement implementation error
I5	μp2	Online RAM test is incomplete	Requirement implementation error
I6	μp2	Online EEPROM failure not identified as fatal failure	Requirement implementation error
I7	CP	Cannot initialize variable successfully	Requirement implementation error

Regarding the completeness of testing, of the 3757 code statements over the three components, 3379 were tested, representing an average TC of 90.8%. Broken down by module, the coverage for μp1, μp2, and CP are 88.6%, 93.9%, and 89.8% respectively. The varying degrees of code coverage present potential areas where additional testing may be needed.

5.4.4 Coverage Factor

For CF, faults were manually injected into the software of the micro-processors and communication modules to trigger software defects. The faults injected prompt the system into recovery mode and to check existing state variables. In most scenarios, the system either returns to normal, remains in recovery mode, or enters a failed state. Here a failed state is any state where the intended function is not achieved or is spurious (i.e., trip signal activated when not needed). The defects discovered from the fault injection testing process can be seen in Table 27.

Table 27. CF Defect tag, description, and location.

Tag	Location	Requirement Description	Defect Description
I8	μp1	Cannot detect incorrect value of the variable SA_TRIP_1_DEENRGZE	Requirement implementation error
I9	μp1	Cannot detect fAnalog_Input_6	Requirement implementation error
I10	μp2	Cannot detect incorrect value of the variable Trip_condition	Requirement implementation error
I11	μp2	Cannot detect incorrect value of the variable AIN	Requirement implementation error
I12	μp1	Cannot detect incorrect value of the variable chLEDs_Outputs	Requirement implementation error
I13	μp2	Cannot detect incorrect value of the variable have_dpm	Requirement implementation error

Of the 11,710 faults injected into μp1 and μp2, 2355 tests resulted in a failed state. This translates to a CF of 79.88% as define in Section 4.3. This suggests the software is only ~80% robust at handling and detecting erroneous scenarios and transitioning to a recoverable or normal state. It also suggests development in the resiliency of program is potentially needed.

5.5 Failure Categorization via Orthogonal Defect Classification

In this section, the discovered defects are categorized into ODC classes and assigned to UCA/UIF basic events. Following the methodology discussed in Section 4.3, each defect can be classified. Documentation defects are excluded as they do not impact software reliability and instead only impact usability which is not explored in this report. In Table 28, the separation of defects into ODC classes can be seen.

Table 28. Grouped ODC defect classes and tags.

Defect Class	Defect Tag
Function	X1, D1, D2, D3, D4, D5, D6, D7, D10
Assignment	I7
Interface	None
Checking	D9, I1, I2, I3, I4, I5, I6, I8, I9, I10, I11, I12, I13
Timing / Serialization	None
Build / Package / Merge	None
Algorithm	D8

For design defects, D1–D7 and D10, are functionality specified in the SRS and SDD but are explicitly missing from the code. They are all classified as function defects as they are functional/non-functional requirements. In addition, the resolution of these defects would require a formal code design change to implement them. Design defects D8 and D9 were implemented but due to requirement ambiguity did not consider all scenarios. Specifically, D8 is a trip detection algorithmic error as the program follows an incorrect logic path when a real trip signal is detected. This is associated with an algorithmic defect class. D9 is also a programming error due to incomplete examination of the entire bit address. This is considered a checking defect class as the program fails to adequately validate the entire data address before utilizing. For implementation defects, I1–16 and I8–I13 are all associated with insufficient data

validation of inputs, addresses, or tests. These are all considered checking defects. Defect I7 is a failure to initialize a variable successfully and is considered an assignment defect.

For UCA/UIF coupling, using Figure 19 as a guideline, Table 29 shows identified UCAs/UIFs and the corresponding ODC classes. Only UCAs/UIFs identified in the integrated FT are included. ODC classes that do not have any defects in them (i.e., timing, are also excluded).

Table 29. Coupling between identified UCAs/UIFs and non-empty ODC classes.

UCA/UIF	UCA/UIF Description	Corresponding ODC Classes
UCA A	HJTC Controller fails to provide a reference signal	All
UCA F	HJTC Controller provides a reference signal too high	Algorithm, Checking
UCA G	HJTC Controller provides a reference signal too low	Algorithm, Checking
UIF A	Calculator/alarm fails to provide signal when needed	All
UIF B	Alarm provides a spurious control signal	Algorithm, Checking
UIF F	Calculator provides a value too high	Algorithm, Checking
UIF G	Calculator provides a value too low	Algorithm, Checking

5.6 Failure Probability Quantification

5.6.1 Single Failure Probability

In this section, the single failure probability for each previously detected defect class is quantified. In total, 24 software defects were discovered split across three different ODC classes and two component groups. To quantify the failure probability of each of these defect classes and groups, the testing condition and environment must be known. In NUREG/CR-7042, these testing parameters were not provided, and thus, the failure rate must be approximated using real software failure data provided. APP software operating data was collected over a period of 281 months from multiple NPPs. Over that period, two software failures were detected, both spurious trip actuation of the APP module. In this report, it is assumed this real failure rate, λ, is the single failure rate for identified defects, normalized per hour. The calculation of lambda is seen in Equation (25).

$$\lambda = \frac{2 \text{ defects}}{281 \text{ months} \cdot 30 \frac{\text{days}}{\text{month}} \cdot 24 \frac{\text{hours}}{\text{day}}} = 9.885 \cdot 10^{-6} \frac{\text{failures}}{\text{hour}} \quad (25)$$

The failure probability determined from Equation (23) normalized per hour is also $9.885 \cdot 10^{-6}$. The corresponding failure probability per defect is applied to each ODC class. As each defect is assumed to be mutually exclusive and independent, the failure probability for each class is the product of the number of defects by the single failure probability. In Table 30 the total failure probability per class is shown.

Table 30. ODC defect classes, tags, and failure probabilities.

Defect Class	Defect Tag	Total Class Failure Probability
Function	X1, D1, D2, D3, D4, D5, D6, D7, D10	$8.896 \cdot 10^{-5}$
Assignment	I7	$9.885 \cdot 10^{-6}$
Checking	D9, I1, I2, I3, I4, I5, I6, I8, I9, I10, I11, I12, I13	$1.285 \cdot 10^{-4}$
Algorithm	D8	$9.885 \cdot 10^{-6}$

The defect class failure probabilities can now be applied to the UCA and information flows by for all components in the QIAS-P system. Only one division is shown as both divisions are identical (Table 31).

Table 31. Single failure probabilities for UCAs/UIFs for all QIAS-P components.

Component	UCA/UIF	Single Failure Probability
HJTC Controller	UCA A	$2.372 \cdot 10^{-4}$
	UCA F	$1.483 \cdot 10^{-4}$
	UCA G	$1.483 \cdot 10^{-4}$
HJTC Calculator	UIF A	$2.372 \cdot 10^{-4}$
	UIF F	$1.483 \cdot 10^{-4}$
	UIF G	$1.483 \cdot 10^{-4}$
HJTC Alarm	UIF A	$2.372 \cdot 10^{-4}$
	UIF B	$1.483 \cdot 10^{-4}$
ICC Calculator	UIF A	$2.372 \cdot 10^{-4}$
	UIF F	$1.483 \cdot 10^{-4}$
	UIF G	$1.483 \cdot 10^{-4}$
ICC Alarm	UIF A	$2.372 \cdot 10^{-4}$
	UIF B	$1.483 \cdot 10^{-4}$
RVL Calculator	UIF A	$2.372 \cdot 10^{-4}$
	UIF F	$1.483 \cdot 10^{-4}$
	UIF G	$1.483 \cdot 10^{-4}$
RVL Alarm	UIF A	$2.372 \cdot 10^{-4}$
	UIF B	$1.483 \cdot 10^{-4}$
RCS Calculator	UIF A	$2.372 \cdot 10^{-4}$
	UIF F	$1.483 \cdot 10^{-4}$
	UIF G	$1.483 \cdot 10^{-4}$
RCS Alarm	UIF A	$2.372 \cdot 10^{-4}$
	UIF B	$1.483 \cdot 10^{-4}$
CET Calculator	UIF A	$2.372 \cdot 10^{-4}$
	UIF F	$1.483 \cdot 10^{-4}$
	UIF G	$1.483 \cdot 10^{-4}$
CET Alarm	UIF A	$2.372 \cdot 10^{-4}$
	UIF B	$1.483 \cdot 10^{-4}$

5.6.2 Common Cause Failure Probability

The quantification of CCF failure probability utilizes the method previously introduced in BAHAMAS, specifically the UPM [62]. This method accounts for a wide range of possible triggers to calculate the beta factor for CCFs. The assumptions for beta factor determination are similar to those defined in Section 5.1. To re-cap, the software and hardware for both redundant divisions of the QIAS-P are identical but physically isolated from each other. In Table 32, the beta factor determination is broken down based on the various UPM metrics.

Table 32. Beta-factor scoring based on environmental and development conditions.

Metric	**Score**	**Score Value**	**Score Reasoning**
Diversity	A	1750	QIAS-P Division A&B are identical
Separation	E	8	Division A&B racks are physically isolated
Understanding	A	1750	Less than 10 operating years of software experience
Analysis	C	100	Reliability studies have been conducted on QIAS-P development
Man-Machine Interface	D	40	Tests and checks exist for QIAS-P software
Safety Culture	E	5	QIAS-P simulations in normal and emergency conditions exist
Control	D	25	Limited access to hardware modules and interfaces
Tests	D	15	Detailed checks have been performed for a reasonable period of time

To determine the beta factor, the score value column is summed up and divided by a scaling value of 50,000. This scaling value was determined separately in [36]. The beta factor based on the UPM for the QIAS-P system was determined to be 0.0738. To determine the probability of CCF, Equation (28) along with the single UCA/UIF failure probability is used.

$$P_{total} = P_{single} + P_{CCF} \tag{26}$$

$$P_{total} = \frac{P_{single}}{1-\beta} \tag{27}$$

$$P_{CCF} = \beta \cdot P_{total} \tag{28}$$

In Table 33, the CCF for each component and corresponding UCA/UIF is shown.

Table 33. CCF rates for all QIAS-P components.

Component	CCF	CCF Probability
HJTC Controller	CCF A	$1.851 \cdot 10^{-5}$
	CCF F	$1.157 \cdot 10^{-5}$
	CCF G	$1.157 \cdot 10^{-5}$
HJTC Calculator	CCF A	$1.851 \cdot 10^{-5}$
	CCF F	$1.157 \cdot 10^{-5}$
	CCF G	$1.157 \cdot 10^{-5}$
HJTC Alarm	CCF A	$1.851 \cdot 10^{-5}$
	CCF B	$1.157 \cdot 10^{-5}$
ICC Calculator	CCF A	$1.851 \cdot 10^{-5}$
	CCF F	$1.157 \cdot 10^{-5}$
	CCF G	$1.157 \cdot 10^{-5}$
ICC Alarm	CCF A	$1.851 \cdot 10^{-5}$
	CCF B	$1.157 \cdot 10^{-5}$
RVL Calculator	CCF A	$1.851 \cdot 10^{-5}$
	CCF F	$1.157 \cdot 10^{-5}$
	CCF G	$1.157 \cdot 10^{-5}$
RVL Alarm	CCF A	$1.851 \cdot 10^{-5}$
	CCF B	$1.157 \cdot 10^{-5}$
RCS Calculator	CCF A	$1.851 \cdot 10^{-5}$
	CCF F	$1.157 \cdot 10^{-5}$
	CCF G	$1.157 \cdot 10^{-5}$
RCS Alarm	CCF A	$1.851 \cdot 10^{-5}$
	CCF B	$1.157 \cdot 10^{-5}$
CET Calculator	CCF A	$1.851 \cdot 10^{-5}$
	CCF F	$1.157 \cdot 10^{-5}$
	CCF G	$1.157 \cdot 10^{-5}$
CET Alarm	CCF A	$1.851 \cdot 10^{-5}$
	CCF B	$1.157 \cdot 10^{-5}$

5.7 Unacceptable Failure Probabilities from Integrated Fault Tree

Using Table 31 and Table 33 for single and CCF probabilities, the top event failure probability can be calculated. SAPHIRE 8 was the software used to calculate the failure probabilities from the integrated FT. The top event analyzed is "Operator fails to initiate reactor trip causing reactor damage." The total failure probability of this event was calculated to be $4.463 \cdot 10^{-4}$. In Table 34, different cut sets for this event are shown. As multiple cut sets will have the same probability, the table is truncated to only show different set likelihoods. For example, Cut Sets 1–10 all have the probability but describe different failures. The columns, from left to right, describe the cut set order, failure probability, cut set tag, and description of failure. Cut sets are ordered by magnitude with largest occurrence frequency at the top. Cut sets with the same magnitude are then sorted by alphabetical order.

Table 34. Cut sets derived from SAPHIRE 8 for top event of interest.

#	Failure probability	Cut Set Tag	Failure Description
0	$5.000 \cdot 10^{-5}$	QPD-H	QIAS-P display hardware failure
1	$1.851 \cdot 10^{-5}$	QPD-ICA-CFA	CCF of division A&B ICA module, fails to actuate when need
11	$1.157 \cdot 10^{-5}$	QPD-ICA-CFB	CCF of division A&B ICA module, spurious activation
26	$5.626 \cdot 10^{-8}$	QPD-PA-ICA-YA, QPD-PB-RSC-YA	Div. A ICA module, UIF A and Div. B RSC module, UIF A
126	$3.518 \cdot 10^{-8}$	QPD-PA-ICA-YA, QPD-PB-RSC-YF	Div. A ICA module, UIF A and Div. B RSC module, UIF F
426	$2.199 \cdot 10^{-8}$	QPD-PA-ICA-YF, QPD-PB-RSC-YF	Div. A ICA module, UIF F and Div. B RSC module, UIF F

Cut Set 1 is the failure of the display panel for QIAS-P monitored parameters. This assumes there is only one monitor for all parameters. Given that most control rooms will have multiple backup monitors, this cut set is not informative and can be excluded to examine the DI&C cut sets. Excluding the FPD, the DI&C failure probability of the QIAS-P system (including both software and hardware) was calculated to be $3.824 \cdot 10^{-4}$ (magnitude of cut sets from Table 34 does not change). Cut Sets 1–25 are all CCF events of the QIAS-P system and represent 93.85% of possible events to occur. This is expected as both Divisions A and B were assumed to have no hardware or software diversity. Cut Sets 26–426 represent single failures from each division and represent only 6.15% of all possible failures. If diversity is assumed and CCF cut sets are excluded for the calculation of the top event, the DI&C failure probability from only single failures is $2.35 \cdot 10^{-5}$.

5.8 Results and Discussion

In this section, the overall results from the ORCAS method are discussed. The strengths, limitations, and weakness of ORCAS is also elaborated.

In summary, for the QIAS-P system, the top event was assigned to be "Operator fails to initiate reactor trip causing reactor damage." For simplified operator-decision models, this event is triggered whenever the QIAS-P HSI system fails to properly display monitored parameters or alarms. The total probability of this top event was calculated (from SAPHIRE 8) to be $4.463 \cdot 10^{-4}$ from the integrated FT. This includes hardware and software single and CCF failures. The top contributing basic event to this event was the failure of the display monitor of the QIAS-P system. Considering main control rooms will have multiple monitors, this failure can be excluded to analyze the DI&C contribution to overall failure probability. For the two division of the QIAS-P system, the failure probability was determined to be $3.824 \cdot 10^{-4}$. It was discovered the majority of the contributing basic events was the software CCFs between the digital modules among the two divisions. This is due to the assumption there is no hardware or software diversity between the divisions as it was not mentioned in the APR-1400 instrumentation and control report [46]. Considering only the single software and hardware failures, the failure probability was determined to be $2.35 \cdot 10^{-5}$. This value was determined using the defects identified in the APP RTS in NUREG/CR-7042 [56]. Due to the strong similarities in the APP RTS monitoring and control system to the APR-1400 DI&C, all defects identified were applicable and possible. However, analyze for the APR-1400 is still preferred as this would reveal design and implementation defects that currently exist in the system.

In total, the APP RTS software was operational for 281 months in legacy reactors. During this operating period, only two relevant software failures were detected. This corresponds to a failure probability of $9.885 \cdot 10^{-6}$. Comparing our result with the APP RTS reliability, the single failure probability of the APR-1400 is on the same order of magnitude. For the same duration (281 months), four software-related failures would manifest in the APR-1400 (not considering CCF or display monitor failure). If software CCFs are considered, approximately 77 software failures are expected for the QIAS-P system. Given that within the APR-1400 system, the QIAS-P is only one of four redundant HSI systems, it is expected even without diversity among QIAS-P divisions, the failure probability of the top event should be lower than $2.35 \cdot 10^{-5}$.

Design recommendations can also be derived from this methodology. Design defects, D1–D10, are all functions that were specified in the SRS and SDD but were not implemented in the code. RT and DD were the primary analysis techniques to identify these deficits. Future development directions can be focused on these missing design defects.

Regarding identified implementation defects, the testing and fault injection methodologies (TC and CF) were able to discover an additional 13 defects. These defects were all related to faulty or inadequate checking or algorithmic implementation of the software and provides additional design direction for future versions.

While the semantic and test-based metric analysis techniques were capable of discovering software defects, each method has limitations. For RT, the invalidating assumption is the requirements specifications were complete for the software application. This is not always the case, and in many scenarios, designers may encounter a need for additional functionality. Defect X1 is a good example of incomplete requirements specifications. In the defect, additional code was implemented that was not specified in either the SRS or SDD. Depending on the software development team, the SRS and SDD may have serious requirement deficiencies which cannot be caught by RT analysis and will skew defect detection. In addition, compliance to quality assurance policies is absolutely required to ensure adequate documentation which can be difficult, expensive, or tedious for engineering teams.

While DD, the second semantic analysis technique, was designed to ensure engineering design process is complete, it also has serious limitations. The most prominent is the subjectivity of inspectors which are required to ensure code and documentation compliance. Based on the qualification and decision made by the inspectors, it is possible no defects are discovered leading to undetected defects passing through the various software development lifecycles. Furthermore, the measure of DD has no acceptable maximum boundary and cannot inform on software reliability. For example, in the case study, it was determined the APP software had a DD value of $0.829\ \text{defects/KLOC}$. It is difficult to justify if this signifies good or bad software quality. Rather it only suggests defects exist in the software. It is clear semantic analysis alone is not sufficient for defect detection, and test-based methods are required.

For TC, while having comprehensive testing can directly inform on overall reliability, it is also highly dependent on the testing environment. Unrealistic or idealized testing environments can lead to overconfident reliability results. This is especially true for complex operational environments where the number of variables in the system are too large account for every possible scenario. While T-way combinatorial testing attempts to resolve this problem through boundary value analysis and equivalence partitioning, extensive combinatorial testing is still required and may require considerable computational resources. Furthermore, only 90.8% of software requirements were tested. Of the remaining 9.2%, it is possible defects exist that were undetected due to under testing of the required functionality. Based on the detection rate of TC, it suggests one potential defect may remain in the untested code portion.

Fault injection is another test-based method utilized in this report, referenced as CF, to detect software resiliency. However, fault injection is also susceptible to idealized testing environments that may be under-representative of the real operational profile. T-way combinatorial testing is also recommended here to reduce the number of redundant tests required but is limited by the scenarios considered. For CF,

while 11,710 fault scenarios were tested, it is still difficult to justify whether a sufficient number of scenarios were considered. To accomplish comprehensive testing (whether TC or CF), detailed knowledge of the operational profile, software interface, and recovery options are required, all of which can be difficult obtain.

In summary, while each software quality metric has drawbacks, the combination of semantic and test-based analyses is designed to detect the majority of significant software defects. The two fault detection perspectives can detect a wide range of defects that can manifest in different locations in the software development lifecycle from requirements specifications to fault injection.

6. CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

This report documents the research activities in FY-21 that quantitatively evaluate CCFs (especially software CCFs) in HSSSR DI&C systems and in NPPs using the IRADIC technology.

Chapter 2 presented the CCF analysis of a highly redundant digital RTS. An approach for performing CCF analysis given the limited data and multiple CCCGs is developed and demonstrated using a case study. The approach relies on the modified BFM to account for multiple CCCGs and PBF-2 to define beta factors for each CCCG. Together these two methods will provide a means to overcome the limitations of conventional methods. Because of the unique and highly redundant structure of digital RTS, there is a need to model several of its components as part of multiple CCCGs.

The modified BFM was selected to model components with multiple CCCGs as it was constructed for such a case. Normally CCF methods rely on historical data or experience to define model parameters. However, with limited data available concerning the RTS, a solution for quantifying model parameters had to be found. Without the parameters, the modified BFM would not work. Several elicitation methods were reviewed, and PBF-2 was selected. The novel application of PBF-2, together with the modified BFM, allows for a successful quantification process for the multiple CCCGs under a limited-data scenario. In conclusion, the CCF modeling approach developed for this work provides an effective means of quantification given the scope of the case study. The proposed approach provides a means to account for software CCFs in PRA for a limited-data scenario.

Chapter 3 describes consequence analysis based on INT-TRANS and relevant accident scenarios. The changes of CDF after adding integrated FTs of digital RTS and ESFAS to the generic PWR ET models are compared and discussed. Integrated FTs of RTS and ESFAS include both software and hardware failures, particularly CCF, that may occur in a four-division digital RTS and a four-division digital ESFAS. Results show the CDF of INT-TRANS accident scenarios are reduced significantly.

By adding the integrated FTs of four-division digital RTS and ESFAS into the PRA models, the safety margin obtained from the plant digitalization on HSSSR DI&C systems are quantitatively estimated. For example, results show RTS failure probability is half-reduced from 4.288E-6 to 1.270E-6; LPI failure probability greatly decreases from 8.416E-4 to 2.258E-4 due to the improvement of ESFAS-FT. This explains the significant reduction of CDF in these analyzed accident scenarios. It indicates plant modernization including the improvement of HSSSR DI&C systems such as RTS and ESFAS will make great benefits to plant safety by providing increased safety margins to accident management.

In Chapter 4, a novel approach, ORCAS, was presented for the quantification of software hazards when sufficient operational and testing data was available. The method incorporates elements of quality software development as well as strong analysis techniques to identify and link software defects to potential failure modes. The approach includes both semantic and test-based analysis to detect failures that can exist in different stages of the SDLC. In addition, considerations for comprehensive testing, yet computationally feasible, are also incorporated using T-way testing to generate valid, invalid, and edge case test conditions. Identified defects are then systematically linked and coupled with basic events via orthogonal defect classification. The semantic bridging presented by ODC allows flexibility but also solid guidelines in the coupling of root causes to software hazards.

In previous work in FY-19 and 20, RESHA was applied to analyze the possible software hazards/basic events as well as software CCFs that can occur through highly redundant systems. ORCAS, a detailed method to quantify software hazards identified with the RESHA method, is presented and a case study is examined. This method leverages existing methodologies in a unique combination to provide software specific failure quantification.

In Chapter 5, the ORCAS method is applied to the advanced HSI relevant to reactor trip safety developed from the APR-1400 HSI design. The HSI of the APR-1400 consists primarily of four redundant information retrieval systems for the operator. During nominal reactor operation, the QIAS-N receives analog and digital signals from both safety and non-safety-related plant components. Under both nominal and safety relevant scenarios, the QIAS-P acts as both a continuous source of accident monitoring information as well as a backup operator display module to the QIAS-N system. The primary role of the QIAS-P system is to provide unambiguous indication of ICC as well as advanced warning of the approach towards it. Both QIAS information systems are implemented using PLC-based control platforms. For the current work scope, only the QIAS-P system is analyzed using the adapted methodology.

For the QIAS-P system, the top event was assigned to be "Operator fails to initiate reactor trip causing reactor damage." For simplified operator-decision models, this event is triggered whenever the QIAS-P HSI system fails to properly display monitored parameters or alarms. The total probability of this top event was calculated (from SAPHIRE 8) to be $4.463 \cdot 10^{-4}$ from the integrated FT. This includes hardware and software single and CCF failures. The top contributing basic event to this event was the failure of the display monitor of the QIAS-P system. Considering main control rooms will have multiple monitors, this failure can be excluded to analyze the DI&C contribution to overall failure rate. For the two division of the QIAS-P system, the failure rate was determined to be $3.824 \cdot 10^{-4}$. It was discovered the majority of the contributing basic events was the software CCFs between the digital modules among the two divisions. This is due to the assumption there is no hardware or software diversity between the divisions as it was not mentioned in the APR-1400 instrumentation and control report. Given that within the APR-1400 system, the QIAS-P is only one of four redundant HSI systems, it is expected even without diversity among QIAS-P divisions, the failure rate of the top event should be lower than $2.35 \cdot 10^{-5}$.

6.2 Future Works

In this project, the IRADIC technology has been developed and demonstrated for a digital design of RPS and ESFAS of existing plants with multilayer software CCFs, human interactions with these systems, and plant responses. This technology complements other approaches being developed for deploying DI&C technologies and emphasizes risk-informed approaches used to facilitate the adoption and licensing of HSSSR DI&C systems. Currently, only qualitative assessment is required for evaluating design attributes and quality measures of DI&C systems because there is no appropriate approach for performing quantitative assessment. To deal with the technical issues in addressing potential software CCF issues in HSSSR DI&C systems of NPPs, the IRADIC technology provides:

- An integrated and best-estimate, risk-informed capability to address new technical digital issues quantitatively, accurately, and efficiently in plan modernization progress, such as software CCFs in HSSSR DI&C systems of NPPs.
- A common and modularized platform for I&C designers, software developers, plant engineers and risk analysts to efficiently prevent and mitigate risk by identifying crucial failure modes and system vulnerabilities, quantifying DI&C system reliability, and evaluating the consequences of digital failures on the plant responses.
- A technical basis and risk-informed insights to assist NRC and industry in formalizing relevant licensing processes relevant to CCF issues in HSSSR DI&C systems.
- An integrated risk-informed tool for vendors and utilities to meet the regulatory requirements and optimize the D3 applications in the design of digital HSSSR systems.

The IRADIC technology expects to develop the capability of quantitative assessment to fill the technical gaps, it follows the trend and the need in digital modernization of existing NPPs. The work scope of this project in FY-22 is to further improve and demonstrate IRADIC technology based on the

achievements from FY-19 to FY-21, particularly in developing a software CCF modeling method, enhancing the IRADIC flexibility in coupling various state-of-the-art reliability analysis methods for different conditions, and providing best-estimate consequence analysis to fully evaluate the safety margins introduced by plant digitalization in different accident scenarios.

Key activities in FY-22 include:

- Evaluate and improve existing CCF modeling methods for estimating probabilities of potential software CCF in HSSSR DI&C systems. Build up the capability of software CCF modeling and embed it into the IRADIC technology. Uncertainty quantification, sensitivity analysis and validation will be performed with limited available data.
- Collaborate with university partners to further improve the reliability analysis approach in IRADIC for DI&C systems, and develop a "plug-and-play" capability for IRADIC to couple with various the-state-of-the-art software reliability analysis methods for different conditions.
- Complete systematic hazard analysis and reliability study for human-machine interface relevant to actuation of engineered safety features.
- Perform an integrated risk-informed analysis (including hazard, reliability, and consequence analysis) of additional accident scenarios (e.g., SBO, loss of offsite power, and loss of main feedwater) affected by failures in HSSSR DI&C systems for generic plants using improved IRADIC technology.
- Adjust and demonstrate the IRADIC technology on the risk assessment and design optimization of AI-guided advanced control systems. Considering the potential and trend to introduce AI/ML techniques in the design and operation of LWRs, it is necessary to ensure the reliability and availability of these AI-guided operator-advisory software or digital platforms. By accurately identifying potential risks and estimating their impacts to plant safety (e.g., CDF), IRADIC can provide risk-informed insights to support the decision-making of operators to deal with potential software/digital/cyber failures, especially software CCF. Besides, IRADIC can optimize D3 applications in the design stage of AI-guided HSSSR DI&C systems, especially to the designs of ML-based digital twins. By estimating the safety margins obtained by the plant digitalization in a best-estimated means, suggestions can be provided to the designers.

7. REFERENCES

[1] Bao, H., H. Zhang, and K. Thomas. 2019. "An Integrated Risk Assessment Process for Digital Instrumentation and Control Upgrades of Nuclear Power Plants." Idaho National Laboratory, Idaho Falls, ID. https://doi.org/10.2172/1616252.

[2] Bao, H., H. Zhang, and T. Shorthill. 2020. "Redundancy-guided System-theoretic Hazard and Reliability Analysis of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants." Idaho National Laboratory, Idaho Falls, ID. https://www.osti.gov/servlets/purl/1668835.

[3] Aldemir, T., D.W. Miller, M.P. Stovsky, J. Kirschenbaum, P. Bucci, A.W. Fentiman, and L.T. Mangan. 2006. "Current State of Reliability Modeling Methodologies for Digital Systems and Their Acceptance Criteria for Nuclear Power Plant Assessments." NUREG/CR-6901, U.S. Nuclear Regulatory Commission, Washington, D.C.. http://purl.access.gpo.gov/GPO/LPS100342.

[4] Kirschenbaum, J., P. Bucci, M. Stovsky, D. Mandelli, T. Aldemir, M. Yau, S. Guarro, E. Ekici, and S.A. Arndt. 2009. "A Benchmark System for Comparing Reliability Modeling Approaches for Digital Instrumentation and Control Systems." *Nuclear Technology* 165, no. 1: 53-95. https://doi.org/10.13182/NT09-A4062.

[5] Thomas, K., and K. Scarola. 2018. "Strategy for Implementation of Safety-Related Digital I&C Systems." INL/EXT-18-45683, Idaho National Laboratory, Idaho Falls, ID.

[6] Wierman, T.E., D.M. Rasmuson, and A. Mosleh. 2007. "Common-Cause Failure Databased and Analysis System: Event Data Collection, Classification, and Coding." NUREG/CR-6268, Rev. 1, Idaho National Laboratory, Idaho Falls, ID.

[7] U.S. Nuclear Regulatory Commission. 1979. "A Defense-In-Depth and Diversity Assessment of the RESAR-414 Integrated Protection System." NUREG-4093, U.S. Nuclear Regulatory Commission, Washington, D.C..

[8] U.S. NRC. 2016. "Historical Review and Observations of Defense-in-Depth. NUREG/KM-0009, U.S. Nuclear Regulatory Commission, Washington, D.C.

[9] U.S. NRC. 2016. "Standard Review Plan for the Review of Safety Analysis Reports for Nuclear Power Plants: LWR Edition — Instrumentation and Controls." NUREG-0800, U.S. Nuclear Regulatory Commission, Washington, D.C.

[10] U.S. NRC. 1995. "Use of Probabilistic Risk Assessment Methods in Nuclear." Federal Register 60, no. 158: 42622-42629.

[11] U.S. NRC. 2018. "Plans for Addressing Potential Common Cause Failure in Digital Instrumentation and Controls." SECY-18-0090, U.S. Nuclear Regulatory Commission, Washington, D.C.

[12] U.S. NRC. 2019. "Integrated Action Plan to Modernize Digital Instrumentation and Controls Regulatory Infrastructure." ML19025A312, U.S. Nuclear Regulatory Commission, Washington, D.C.

[13] Arndt, S.A., and A. Kuritzky. 2010. "Lessons Learned from the U.S. Nuclear Regulatory Commission's Digital System Risk Research." *Nuclear Technology* 173, no. 1: 2-7. https://doi.org/10.13182/NT11-A11478.

[14] Clark, A.J., A.D. Williams, A. Muna, and M. Gibson. 2018. "Hazard and Consequence Analysis for Digital Systems – A New Approach to Risk Analysis in the Digital Era for Nuclear Power Plants." *in Transactions of the American Nuclear Society* 119, no. 1: 888-891.

[15] Leveson, N.G. and J. P. Thomas, *STPA Handbook*. March 2018.

[16] Nuclear Energy Institute. 2021. "Guidance for Addressing CCF in High Safety Significant Safety-related DI&C Systems." NEI.

[17] U.S. NRC. 2002. "An Approach for Using Probabilistic Risk Assessment in Risk-Informed Decisions on Plant-Specific Changes to the Licensing Basis." Regulatory Guide 1.174 Rev. 1. U.S. Nuclear Regulatory Commission, Washington, D.C.

[18] Shorthill, T., H. Bao, H. Zhang, and H. Ban. 2020. "A Redundancy-Guided Approach for the Hazard Analysis of Digital Instrumentation and Control Systems in Advanced Nuclear Power Plants." https://arxiv.org/abs/2005.02348v1.

[19] Bao, H., T. Shorthill, and H. Zhang. 2020. "Hazard Analysis for Identifying Common Cause Failures of Digital Safety Systems using a Redundancy-Guided Systems-Theoretic Approach." *Annals of Nuclear Energy* 148, 107686. https://doi.org/10.1016/j.anucene.2020.107686.

[20] Shorthill, T., H. Bao, Z. Hongbin, and H. Ban. 2021. "A novel approach for software reliability analysis of digital instrumentation and control systems in nuclear power plants." *Annals of Nuclear Energy* 158: 108260. https://doi.org/10.1016/j.anucene.2021.108260.

[21] Zhang, H., R. Szilard, S. Hess, and R. Sugrue. 2018. "A Strategic Approach to Employ Risk-Informed Methods to Enable Margin Recovery of Nuclear Power Plants Operating Margins." Idaho National Laboratory, Idaho Falls ID. https://doi.org/10.2172/1514992.

[22] U.S. NRC. 2016. "Guidance for Evaluation of Diversity and Defense-In-Depth in Digital Computer-Based Instrumentation and Control Systems Review Responsibilities." in *NUREG-800.* U.S. Nuclear Regulatory Commission, Washington, D.C..

[23] NEA. 2004. "International Common-Cause Failure Data Exchange. ICDE General Coding Guidelines - Technical Note." NEA/CSNI/R(2004)4, Nuclear Energy Agency.

[24] Mosleh, A., D. Rasmuson, and F. Marshall. 1998. "Guidelines on Modeling Common-Cause Failures in Probabilistic Risk Assessment." NUREG/CR-5485, U.S. Nuclear Regulatory Commission, Washington, D.C..

[25] Parry, G.W. 1991. "Common Cause Failure Analysis: A Critique and Some Suggestions." *Reliability Engineering and System Safety* 34, no. 3: 309-326. https://doi.org/10.1016/0951-8320(91)90106-H.

[26] Paula, H.M., D.J. Campbell, and D.M. Rasmuson. 1991. "Qualitative cause-defense matrices: Engineering tools to support the analysis and prevention of common cause failures," *Reliability Engineering & System Safety* 34, no. 3: 389-415. https://doi.org/10.1016/0951-8320(91)90110-S .

[27] O'Connor, A. and A. Mosleh. 2013. "A General Cause Based Methodology for Analysis of Dependent Failures in System Risk and Reliability Assessments." Ph.D. diss., University of Maryland.

[28] Hokstad, P. and M. Rausand. 2008. "Common Cause Failure Modeling: Status and Trends." in The Handbook of Performability Engineering, edited by K.B. Misra, 621-640 London, Springer. https://doi.org/10.1007/978-1-84800-131-2_39.

[29] U.S. Nuclear Regulatory Commission. 2011. "Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8.0." NUREG/CR-7039 Vol. 3, U.S. Nuclear Regulatory Commission, Washington, D.C.

[30] Ma, Z., R.F. Buell, J.K. Knudsen, and S. Zhang. 2020. "Common-Cause Component Group Modeling Issues in Probabilistic Risk Assess." INL/CON-20-57077, Idaho National Laboratory, Idaho Falls, ID.

[31] Bobbio, A., L. Portinale, M. Minichino, and E. Ciancamerla. 1999. "Comparing Fault Trees and Bayesian Networks for Dependability Analysis." In Computer Safety, Reliability and Security, edited by Felici M. and Kanoun K. https://doi.org/10.1007/3-540-48249-0_27.

[32] Chu, T.L., M. Yue, M. Martinez-Guridi, and J. Lehner. 2010. "Review of Quantitative Software Reliability Methods." Brookhaven National Laboratory. https://doi.org/10.2172/1013511.

[33] Zitrou, A. 2006. "Exploring a Bayesian Approach for Structural Modelling of Common Cause Failures." Ph.D. diss., University of Strathclyde.

[34] Kancev, D. and M. Cepin. 2012. "A new method for explicit modelling of single failure event within different common cause failure groups." *Reliability Engineering and System Safety* 103: 84-93. https://doi.org/10.1016/j.ress.2012.03.009.

[35] Johnston, B.D. 1987. "A Structured Procedure for dependent Failure Analysis (DFA)." *Reliability Engineering* 19, no. 2: 125-136. https://doi.org/10.1016/0143-8174(87)90107-7.

[36] Humphreys, R.A. 1987. "Assigning a Numerical Value to the Beta Factor Common Cause Evaluation." in Reliability '87, Proceedings of the Sixth Conference Birmingham, England; 2C/5/1-2C/5/8.
[37] Lindberg, S. 2007. "Common cause failure analysis: Methodology evaluation using Nordic experience data." Ph.D. diss., Uppsala Universitet.
[38] Paula, H.M., G.W. Parry, D.J. Campbell, D.B. Mitchell, and D.M. Rasmuson. 1990. "A Cause-Defense Approach to the Understanding and Analysis of Common Cause Failures." NUREG/CR-5460, U.S. Nuclear Regulatory Commission, Washington D.C.
[39] 1998. "The Betaplus Common Cause Failure Model." *Safety and Reliability* 18, no. 1: 16-23. https://doi.org/10.1080/09617353.1998.11690672.
[40] Technis. 2021. "Betaplus." Wilde Analysis Ltd., Accessed July 2021. https://www.wilderisk.co.uk/software/technis-products/betaplus/.
[41] Korea Electric Power Corporation and Korea Hydro & Nuclear Power Co., Ltd. 2018. "APR1400 Design Control Document Tier 2. Chapter 7: Instrumentation and Controls.", South Korea.
[42] Varde, P.V., J.G. Choi, D.Y. Lee, and J.B. Han. 2003. "Reliability Analysis of Protection System of Advanced Pressurized Water Reactor-APR 1400." KAERI/TR-2468/2003, Korea Atomic Energy Research Institute, South Korea.
[43] Ma, Z., C. Parisi, H. Zhang, D. Mandelli, C. Blakely, J. Yu, R. Youngblood, and N. Anderson. 2018. "Plant-Level Scenario-Based Risk Analysis for Enhanced Resilient PWR – SBO and LBLOCA." Idaho National Laboratory, Idaho Falls, ID. https://doi.org/10.2172/1495192.
[44] Ma, Z., C. Parisi, C. Davis, J. Park, R. Boring, and H. Zhang. 2019. "Risk-Informed Analysis for an Enhanced Resilient PWR with ATF, FLEX, and Passive Cooling." Idaho National Laboratory, Idaho Falls, ID. https://doi.org/10.2172/1777257.
[45] Ma, Z., C. Davis, C. Parisi, R. Dailey, J. Wang, S. Zhang, H. Zhang, and M. Corradini. 2019. "Evaluation of the Benefits of ATF, FLEX, and Passive Cooling System for an Enhanced Resilient PWR Model." Idaho National Laboratory, Idaho Falls, ID. https://doi.org/10.2172/1777262.
[46] K. E. P. Corporation and L. T. D. Korea Hydro & Nuclear Power Co. 2018. "APR 1400 Instrumentation and Controls." 2018.
[47] Lee, M., S. Song and D. Yun. 2012. "Development and application of POSAFE-Q PLC platform." International Atomic Energy Agency 43 no. 50. .
[48] Chung, H.-Y., and D.-W. Kim. 2003. "Design of advanced power reactor (APR1400) I&C system." *IFAC Proceedings Volumes* 36, no. 20: 729-734. https://doi.org/10.1016/S1474-6670(17)34557-3.
[49] 2017. "IEEE 1633-2016 - IEEE Recommended Practice on Software Reliability." in IEEE Std 1633-2016 (Revision of IEEE Std 1633-2008). https://doi.org/10.1109/IEEESTD.2017.7827907..
[50] Goel, A.L. and K. Okumoto. 1979. "Time-dependent error-detection rate model for software reliability and other performance measures." *IEEE Transactions on Reliability* R-28, no. 3: 206-211. https://doi.org/10.1109/TR.1979.5220566.
[51] Musa, J.D. and K. Okumoto. 1984. "A logarithmic poisson execution time model for software reliability measurement." *in Proceedings of the 7th international conference on Software engineering*, 230-238.
[52] Jelinski, Z. and P. Moranda. 1972. "Software reliability research." Statistical Computer Performance Evaluation, 465-484. https://doi.org/10.1016/B978-0-12-266950-7.50028-1.
[53] Schneidewind, N.F. 1975. "Analysis of error processes in computer software." *ACM SIGPLAN Notices* 10, no. 6: 337-346. https://doi.org/10.1145/390016.808456.
[54] Smidts, C. and M. Li. 2000. “Software Engineering Measures for Predicting Software Reliability in Safety Critical Digital Systems” NUREG/GR-0019, U.S. Nuclear Regulatory Commission.
[55] Smidts, C.S. and M. Li. 2004. "Preliminary Validation of a Methodology for Assessing Software Quality." NUREG/CR-6848, U.S. Nuclear Regulatory Commission.

[56] Smidts, C.S., Y. Shi, M. Li, W. Kong, and J. Dai. 2011. "A large scale validation of a methodology for assessing software reliability." NUREG/CR-7042, U.S. Nuclear Regulatory Commission.
[57] Gaffney, J.E. 1984. "Estimating the number of faults in code." *IEEE Transactions on Software Engineering* SE-10, no. 4: 459-464. https://doi.org/10.1109/TSE.1984.5010260.
[58] American Nuclear Society Standards Committee. 2016. "Glossary of Definitions and Terminology," American Nuclear Society.
[59] IEEE. 2006. *IEEE 982.1-2005 - IEEE Standard Dictionary of Measures of the Software Aspects of Dependability*.
[60] Strauss, S.H. and R. G. Ebenau, *Software Inspection Process*. New York: McGraw-Hill Inc., 1993.
[61] Kuhn, D.R., R. N. Kacker, and Y. Lei. 2010. "Practical Combinatorial Testing." Sp 800-142, National Institute of Standards and Technology.
[62] Lindberg, S. 2007. "Common cause failure analysis: Methodology evaluation using Nordic experience data." Corpus ID: 997682.
[63] Chillarege, R., I.S. Bhandari, J.K. Chaar, M.J. Halliday, D.S. Moebus, B.K. Ray, and M.-Y. Wong.1992. "Orthogonal defect classification-a concept for in-process measurements." *IEEE Transactions on Software Engineering* 18, no. 11: 943-956. https://doi.org/10.1109/32.177364.
[64] Choi, K.C., S.W. Song, G.M. Park, and S.J. Hwang. 2012. "Reliability Analysis for Safety Grade PLC (POSAFE-Q)." IAEA.
[65] Park, J.-j., D.-i. Kim, and D.-j. Kim. 2008. "Development of Qualified Indication and Alarm System-PAMI based on POSAFE-Q." Transactions of the Korean Nuclear Society Spring Meeting, Gyeongju, Korea.
[66] Shin, Y.C., H.Y. Chung, and T.Y. Song. 2003. "Advanced MMIS design characteristics of APR1400." International conference on global environment and advanced nuclear power plants; Kyoto, Japan.
[67] Lee, J.-K., K.-I. Jeong, G.-O. Park, and K.-Y. Sohn. 2014. "A Quantitative Reliability Analysis of FPGA-based Controller for applying to Nuclear Instrumentation and Control System." *The Journal of the Korea institute of electronic communication sciences* 9, no. 10: 1117-1123. https://doi.org/10.13067/JKIECS.2014.9.10.1117.
[68] Department of Defense. 1990. *Military Handbook, Reliability Prediction of Electronic Equipment*. Washington D.C.