INL/EXT-20-59550

Light Water Reactor Sustainability Program

Redundancy-guided System-theoretic Hazard and Reliability Analysis of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants

Han Bao, Tate Shorthill, Hongbin Zhang

August 2020

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-20-59550

Light Water Reactor Sustainability Program

Redundancy-guided System-theoretic Hazard and Reliability Analysis of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants

Han Bao, Tate Shorthill, Hongbin Zhang

August 2020

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

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

SUMMARY

In fiscal year (FY)-2019, the Risk-Informed Systems Analysis (RISA) Pathway of the U.S. Department of Energy's (DOE's) Light Water Reactor Sustainability (LWRS) program initiated a project to develop a risk assessment strategy for delivering a strong technical basis to support effective, licensable, and secure digital instrumentation and control (DI&C) technologies for digital upgrades/designs. An integrated risk assessment for digital I&C (RADIC) process was proposed for this strategy, which aims to identify key digital-induced failures, implement reliability analyses on related digital safety instrumentation and control (I&C) systems, and evaluate the unanalyzed sequences introduced by these failures (particularly software common cause failures [CCFs]) at the plant level.

According to the RADIC guidelines and requirements, an approach for REdundancy-guided Systems-theoretic Hazard Analysis (RESHA) was developed in FY-2020 that aims to help system designers and engineers address digital-based CCFs and qualitatively analyze their effects on digital system vulnerability. It also provides a technical basis for implementing future reliability and consequence analyses of unanalyzed sequences and optimizing the use of defense-in-depth (DiD) analyses in a cost-effective way. This approach has been developed and applied for the hazard analysis of a digital reactor trip system (RTS) and engineered safety features actuation system (ESFAS). RESHA provided a means to identify software-based interactions and potential CCFs in highly redundant, state-of-the-art DI&C systems, by fully incorporating redundancy into the hazard analysis process. Embracing redundancy in the analysis allowed the work to meet its objectives in three ways: (1) RESHA defines a step-by-step approach for the hazard analysis of digital systems that can help engineers efficiently make design and risk mitigation decisions by providing them a means to systematically identify the most critical CCFs and hazards of DI&C systems; (2) RESHA identifies the critical hazards of a system, allowing utilities to effectively manage the cost of safety-rated DI&C by strategically eliminating unnecessary design features; and (3) RESHA provides a technical basis for reliability analysis by identifying crucial failure modes and qualitatively determining their effects on system vulnerability. Ultimately, RESHA helps improve the design of highly redundant DI&C through a detailed qualitative hazard analysis. RESHA also provides a technical basis for implementing cybersecurity, reliability, and consequence analysis on unanalyzed sequences and optimizing the use of DiD analysis in a cost-effective way.

Additionally, this work also developed a novel method, a BAyesian and HRA(human reliability analysis)-Aided Method for the reliability Analysis of Software (BAHAMAS), to perform reliability analysis of DI&C systems. Software failure probabilities are quantified using an integrated method that incorporates state-of-the-art techniques in the Bayesian Belief Network (BBN), HRA, and CCF modeling. BAHAMAS provides a means for analyzing new software systems where operational data is rarely available. Additionally, BAHAMAS provides flexibility, which allows the analyst to employ appropriate HRA methods or incorporate new or advanced methods to capture the desired details of any software development life cycle (SDLC). The case study relied on the use of the technique for human error rate prediction (THERP) for the quantification of faults in the SDLC. In this work, BAHAMAS has the potential

of meeting many of these attributes. By providing a clear method and allowing for flexibility in the use of HRA, the door has been opened to allow for reasonable assumptions for case-specific analysis. The method accounts for lifecycle activities and provides consideration for CCFs. With further work on verification and uncertainty quantification, the method has the potential to undergo such actions. Additionally, BAHAMAS can certainly incorporate environmental and other contributors of fault into the BBN. Additional operational considerations—particularly the interactions between the digital system and controlled processes—are partially accounted by consideration of the process model and control algorithm. BAHAMAS provides a flexible and useful tool for the quantification of software failures of DI&C systems and meets many of the desired attributes of an ideal quantitative software reliability method.

CONTENTS

FIGURES

TABLES

ACRONYMS

AOO	anticipated operational occurrence
APR-1400	Advanced Power Reactor 1400 MW
BAHAMAS	BAyesian and HRA-Aided Method for the reliability Analysis of Software
BBN	Bayesian Belief Network
BP	bistable processor
CAE	claim-argument-evidence
CCF	common cause failure
CCMT	cell-to-cell mapping technique
CCS	component control system
CFR	Code of Federal Regulations
CIM	component interface module
CPN	colored petri net
CREAM	Cognitive Reliability and Error Analysis Method
DFM	dynamic flowgraph methodology
DiD	defense-in-depth
DI&C	digital instrumentation and control
DNBR	departure from nuclear boiling ratio
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
ESF	engineered safety feature
ESFAS	Engineered Safety Features Actuation System
ET	event tree
ETA	event tree analysis
FMEA	failure mode effect analysis
FSD	fault size distribution
FT	fault tree
FTA	fault tree analysis
FY	Fiscal Year
GC	group-controller
HAZCADS	HAZards and Consequence Analysis for Digital Systems

HCL	Hybrid Causal Logic
HEP	human error prediction
HRA	human reliability analysis
I&C	instrumentation and control
IAP	integrated action plan
IEC	International Electrotechnical Commission
INL	Idaho National Laboratory
LC	loop-controller
LCL	local coincidence logic
LP	logic processor
LWR	light water reactor
LWRS	Light Water Reactor Sustainability
MCR	main control room
MIT	Massachusetts Institute of Technology
NPP	nuclear power plant
NRC	U.S. Nuclear Regulatory Commission
PPS	plant protection system
PRA	probabilistic risk assessment
QSRM	quantitative software reliability method
RADIC	Risk Assessment for Digital I&C
RCS	reactor control system
RESHA	REdundancy-guided Systems-theoretic Hazard Analysis
RG	Regulatory Guide
RISA	Risk-Informed Systems Analysis
RMS	radiation monitoring system
RPS	reactor protection system
RSR	reserve shutdown room
RTB	reactor trip breaker
RTS	reactor trip system
SDLC	software development life cycle
SAPHIRE	Systems Analysis Programs for Hands-on Integrated Reliability Evaluations
SFP	software failure probability
SNL	Sandia National Laboratories
SOI	system of interest
SP	selective processor

SPAR-H	Standardized Plant Analysis Risk-Human reliability analysis method
SPOF	single point of failure
SRGM	software reliability growth method
SRP	Standard Review Plan
SSC	system, structure, and component
ST	shunt
STPA	systems-theoretic process analysis
THERP	Technique for Human Error Rate Prediction
UCA	unsafe control action
U.S.	United States
UV	undervoltage

Redundancy-Guided System-Theoretic Hazard And Reliability Analysis Of Safety-Related Digital Instrumentation And Control Systems In Nuclear Power Plants

1. INTRODUCTION

Digital upgrades and plant modernization efforts offer the foremost path to performance and cost improvements of nuclear power plants (NPPs) [1]. Despite decades of experience with analog systems, the technical challenges associated with their continued use (e.g., signal drift, high maintenance costs, obsolescence, lack of industrial suppliers) have caused the nuclear industry to move toward digital instrumentation and control (DI&C) in favor of integrated circuitry and the modern microcontroller [2]. Compared with analog systems, DI&C systems offer significant advantages in the areas of monitoring, processing, testing, and maintenance [3] [4]. Notwithstanding the immediate attraction, the nuclear industry has been slow to adopt safety-rated DI&C because each new design must be shown to maintain or improve the status quo by means of a risk assessment [2]. Though many of the concepts for the risk assessment of analog systems carry over, DI&C systems present unique challenges. In 1997, the National Research Council detailed several technical challenges for the implementation of DI&C systems. Those relating specifically to the present work are: (1) the system aspects of digital systems; (2) the potential for software-based common cause failures (CCFs); and (3) the need for a risk assessment method tailored to DI&C systems [2].

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 [5]. Kirschenbaum et al. provide a useful summary of these concerns in their own work on the investigation of digital systems [6]. 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 [1] [7]. In order to make redundancy designs effective, diversity is employed, providing an alternative technology, method, technique, or means to achieve a desired result [8]. The diverse protection helps to eliminate the common features necessary for a CCF. NUREG/CR-5485 provides general guidance for modeling CCFs in risk assessments [9]. NUREG/CR-6303 was published by the NRC in December 1994 as “Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems.” In it, a method was described to identify design vulnerabilities to common-mode failure for computer-based nuclear reactor protection systems [10]. In October 1995, the U.S. Nuclear Regulatory Commission (NRC) called attention to top-level system aspect requirements of DI&C applications in NPPs, which were addressed in the general design criteria in Title 10 of the Code of Federal Regulations (CFR) 50, Appendix A [11]. NUREG/CR-6734 Vols. 1 and 2, published in 2001, provided guidance for reviewing high-integrity software requirements documents in NPPs, which contained a set of 45 failures that illustrate the need for and importance of specific requirements-review guidelines [12] [13]. NUREG/CR-7007, published in 2008 as “Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems,” provided guidance to determine how much diversity in a safety system is needed to mitigate the consequences of potential CCFs identified in the evaluation of safety system design features [14]. NUREG/CR-6819, published in 2008, reviewed the CCF data of emergency diesel generators [15], motor-operated valves [16], pumps [17], and circuit breakers [18] to gain further understanding of why CCF events occur and what measures may be taken to prevent, or at

least mitigate the effect of CCF events. Next, 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" [19]. In 2016, the NRC revised the Standard Review Plan (SRP) to fully adapt it and the associated regulatory guides to DI&C systems [20]. Chapter 7 of the SRP provided guidance for the review of the instrumentation and control (I&C) portions of: (1) applications for nuclear reactor licenses or permits; and (2) amendments to existing licenses. These reports provide a basis for dealing with CCFs. The need remains to identify the most significant CCFs to focus the application of diversity in design.

Diversity and DiD 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 [21]. 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. Furthermore, deterministic guidance available in Chapter 7 of the SRP does not consider digital system reliability quantitatively as part of determining the acceptability of a digital system for safety applications [22]. 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 [23], published in 2018, a DiD analysis for reactor trip systems 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) [24], and it updates the plan as a living document. One of the goals 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 [24].

Therefore, a need clearly exists to develop a risk assessment strategy to support quantitative DiD analyses for assuring the long-term safety and reliability of vital digital systems and reducing uncertainties in costs, time, and support integration of digital systems in the plant. Many efforts from regulatory, industrial, and academic communities have been made for risk analysis of DI&C systems, but there is no consensus method for the software reliability modeling of digital systems in an NPP. NUREG/GR-0020, published in 2000, reviewed existing methods for the analysis of hardware and software CCFs, and their applicability to digital embedded systems. It was found that there is a tight integration of the hardware and software components when embedded digital systems are in actual operation, the effects of hardware and software malfunctions must be analyzed in a unified manner. Some evaluations have been implemented by different researchers for the approaches to model digital protection systems in a probabilistic safety analysis for an NPP. The results showed that the introduction of software causes difficulties in a traditional PRA since probabilistic data are scarce and there is no commonly accepted method for assessing reliability data for software; as such, models often depend on expert judgement [4] [25] [26]. NUREG/CR-6962 [27] reviewed traditional PRA methods for digital systems and indicates that these traditional methods based on event tree/fault tree (ET/FT) and Markov modeling are useful for the PRA of DI&C systems, but still have limitations in the state-of-the-art for modeling digital systems, where additional research and development are needed. NUREG/CR-6942 [28] demonstrated how an existing NPP PRA could incorporate a digital upgrade of the I&C system by considering dynamic reliability modeling. NUREG/CR-6985 [29], published in 2008, implemented two dynamic methodologies—dynamic flowgraph methodology (DFM) and the Markov/cell-to-cell mapping technique (CCMT)—are on the benchmark digital feedwater control system. Zio [30] developed a method for processing accident scenarios generated in a dynamic reliability analysis of a NPP with DI&C, which takes into account both the system states reached at the end of the scenarios, but also the timing and magnitude of the occurred failure events, as well as the characteristics of the process evolution. Ma [31] evaluated the dynamic reliability performance of digital reactor protection systems using the colored petri net (CPN) considering the module repair time and CCFs, and fault tolerance techniques and fault

coverage are introduced to calculate the different failure rates. In 2018, A platform based on the claim-argument-evidence (CAE) theory was designed by Guo and Zou [32] for the demonstration of software reliability and the identification of vulnerability elements, which influenced the reliability of DI&C system software life cycle. These researches indicate timing and interdependency of digital-based failures pose new challenges for the risk assessment of DI&C systems. Besides, to identify the root causes of software failures and estimate the failure probabilities of NPP software, Bayesian networks have been applied for incorporating software failures into an NPP PRA model [33, 34, 35, 36]. Some existing approaches about hazard analysis and reliability analysis have been reviewed in Section 2.

In FY-2019, the Risk-Informed Systems Analysis (RISA) Pathway of the U.S. Department of Energy's (DOE's) Light Water Reactor Sustainability (LWRS) program 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 [37]. An integrated RADIC process was proposed for this strategy, which aims to identify key digital-induced failures, implement reliability analyses on related digital safety I&C systems, and evaluate the unanalyzed sequences introduced by these failures (particularly software CCFs) at the plant level. According to the guidelines and requirements of the RADIC process, an approach for REdundancy-guided Systems-theoretic Hazard Analysis (RESHA) was developed in FY-2020 that aims to help system designers and engineers address digital-based CCFs and qualitatively analyze their effects on digital system vulnerability. It also provides a technical basis for implementing future reliability and consequence analyses of unanalyzed sequences and optimizing the use of DiD analyses in a cost-effective way. This approach has been developed and applied for the hazard analysis of digital reactor trip system (RTS) and engineered safety features actuation system (ESFAS). Relevant description and case studies are shown in Section 3. A method for reliability assessment of digital control systems with consideration for the quantification of CCFs is described in Section 4, which is defined as a BAyesian and HRA(human reliability analysis)-Aided Method for the reliability Analysis of Software (BAHAMAS). Section 5 summarizes the conclusion and future work about risk assessment of DI&C systems.

2. TECHNICAL APPROACHES

The purpose of Section 2 is to provide an overview of approaches for hazard and reliability analysis for DI&C systems.

2.1 Approaches for Hazard Analysis

Hazards can be understood as a state or condition that can lead to a loss of something of value to stakeholders [38]. Hazard analysis is defined as the process of examining a structure, system, or component (SSC) in order to identify hazards and triggers of hazards with the goal of eliminating, mitigating, or controlling them [39]. The 2013 Electric Power Research Institute (EPRI) report on DI&C hazard analysis methods discusses common strategies, as well as indicating their respective strengths and weaknesses [40]. In order 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 [22]. Lessons learned from the NRC's investigation of multiple analysis methods indicate there "may not be one preferable method for modeling all digital systems" [22]. However, combinations of methods may prove beneficial. A recent advancement in hazards analysis developed jointly by 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) [41]. The current work incorporates this concept of combining FTA and STPA as part of the approach for RESHA.

FTA is a conventional, top-down, risk assessment tool that is used to identify the faults that contribute to the failure of a selected top event, which may depend on the combinations of multiple smaller contributors, known as basic events. A key aspect of FTA is the determination of the "cut set," which is a collection of events that, when combined, will result in the failure of the selected top event. There are often many variations of cut sets comprising of single or multiple events [42]. For a hazard analysis, the determination of the cut set is critical and can be done without probability, or failure rate, data [43]. FTA is the workhorse of the nuclear industry and has been used extensively to model control systems.

STPA is an analysis method based on system theory that is used to capture the unsafe interactions between system components, in addition to component failures [38]. Its top-down analysis focuses on the identifying constraints on behavior and the interaction of components [44]. The main parts of STPA are focused on the construction of a control structure and the analysis of unsafe component interactions. The control structure highlights the controller interactions, while the analysis identifies necessary behavior constraints to reduce or eliminate hazards. Together, FTA and STPA will be used to provide a clearer picture of the hazards in DI&C systems, as shown in Figure 1.

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 [38]. 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. Consequently, there is not a clear representation of how to apply STPA or create a control structure for a system containing multiple layers of redundancy. In a 2014 Massachusetts Institute of Technology (MIT) technical report, STPA is applied to an aircraft automatic braking system and redundant features are only mentioned in the context of causal factors [44]. In 2018, Rejzek and Hilbes replicated this idea for an application of STPA to nuclear-related DI&C systems [45].Their application stresses that the control structure only incorporate the functional aspects of the system, again leaving the

details of redundancy for the context portion of STPA [45]. While this aspect of STPA does not appear to lessen its effectiveness, waiting to include the finer details of a system may hinder the identification of CCFs. As a possible consequence of this complication, STPA does not appear to be used specifically to identify and address CCFs in the context of DI&C.

It is essential that STPA have a clear application process for highly redundant systems, including their relationship to characteristics not clearly featured in the standard STPA analyses, to identify CCFs. For this STPA application, all layers of a redundancy design should be incorporated clearly and early in the design analysis. It is proposed that reframing STPA in a redundancy-guided way, in combination with FTA, will provide a means to effectively identify the unique hazards, failure modes, and CCFs associated with highly redundant safety-based DI&C systems. By following this approach, the triggers of the most significant hazards to advanced DI&C systems can be identified and either mitigated or eliminated.

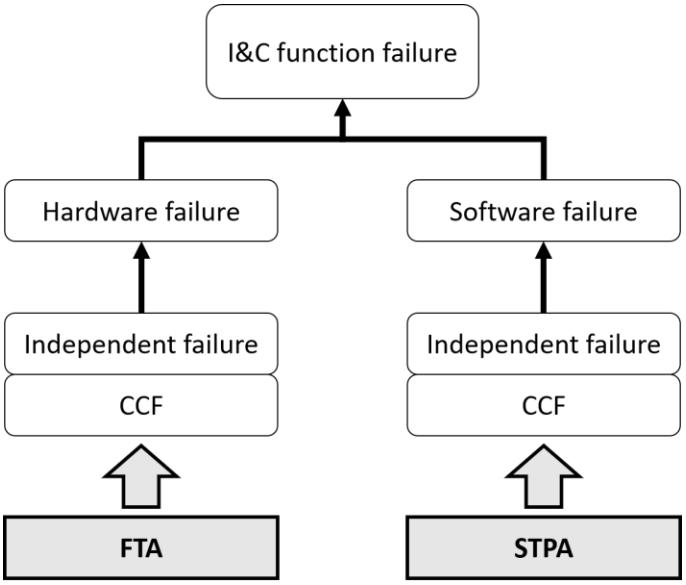


Figure 1. FTA and STPA combined for a full picture of I&C failures adapted from [37].

2.2 Approaches for Reliability Analysis

The desire to model and understand risk, especially concerning digital technologies, prompted considerable research. Of the numerous methods for reliability analysis, a great number of them are based on software reliability growth models, Bayesian Belief Network (BBN) methods, test-based methods, rule-based methods, and metric-based methods [4]. In addition, many software reliability methods have been formulated either to incorporate the timing of events via dynamic modeling [46], or to follow a more traditional static approach using ETs and FTs.

Software reliability growth methods (SRGMs) rely on the theory that software reliability grows or improves with time and updates. The SRGM is a method of modeling software reliability using empirical formulas that have been developed from software failure test data. When applied to analyze a new software, test data for the new software is matched to existing models to provide estimates of software reliability [4]. IEEE standard 1633-2016 [47] includes a collection of SRGM and provides some guidance for when each method might be best applied. Some of the SRGM listed include those by Jelinski and Moranda [48] and Musa [49]. Ultimately, each application is case-specific (i.e., there is no one perfect SRGM for all digital systems). The application of SRGMs tends to evaluate the whole system, rather than separate modules. A main limitation of SRGM is its dependence on available performance data. In addition, the demonstration of a highly reliable system will likely require significant testing (i.e., a limitation shared by the test-based approaches) [4].

BBN methods use an analysis method that relies on Bayes' theorem and graphical models (e.g., a network) to depict the influence of one event on another while also involving dependencies between events. The BBN can incorporate disparate information, which can be advantageous for performing analyses [4]. Recent BBN use has been described in NUREG/CR-7233 for providing an analysis of digital systems by including features of life cycle development [50]. BBN tends to be challenged by their dependence on expert opinion, which can be a source of uncertainty.

Test-based methods rely on analyzing the expected or designed behavior of a system or subsystem based on an expected operational profile [4] [51]. Test-based approaches to reliability might include BBNs and SRGM as these can, and do, rely on test data. The strictly test-based approaches can be generalized as white- or black-boxed. Black box testing is focused on the external inputs and outputs to a system and the analysis of a system in its operational profile. Consequently, the internal workings are masked, thus the term "black box" [52]. By contrast, "white box" testing considers the internal mechanisms of a system [52]. The increased resolution of white box testing can more clearly show the failure modes of a system. Perhaps the most significant challenge of test-based methods is the time constraint that limits the ability to quantify the reliability of a system. Testing all possible inputs might be impractical for demonstrating high reliability, but would be required for safety systems. The rule-based approach is akin to rule- or engineering standard-based engineering design. The idea being to follow design rules that can ensure reliability. The International Electrotechnical Commission (IEC) Standard [53] is indicated in [4] as an example. A limitation of this form is that the methods have not been validated. Metric-based software reliability prediction relies on software attributes, such as bugs per line of code, software maturity, completeness, defect density, etc., to provide a means for determining reliability [54]. The metric-based approach provides the opportunity to capture many unique features that are involved with software development processes.

Some of the most recent advancements for reliability analysis have been those that incorporate timing of events into the analysis [55]; these analyses can be classified as dynamic PRA. The goal of dynamic PRA is to explicitly model the behavior of a system over time [56]. Dynamic PRA can provide an advantage over conventional PRA (e.g., ET/FT) by providing more realistic sequence timing and better representation of thermal hydraulic processes and operator responses [46]. In the past 20 years, there has been significant interest in both dynamic and traditional models. The NRC has sponsored two major parallel investigations—one research path focused on the use of traditional PRA methods (e.g., ET/FT) for the reliability analysis of digital systems [27] [57] and the other focused on dynamic PRA approaches [5] [28] [29]. Dynamic PRA will be saved for later research efforts in part to support our goal to maintain compatibility and familiarity with conventional PRAs currently in use by the industry.

In 2014, Mosleh provided a useful discussion on PRA, including its origin and progression to dynamic, simulation-based methods [46]. Mosleh highlights the successes of classical PRA methods in estimating the core melt frequencies for NPPs in the range of 5E-5 to 5E-4 (e.g., the WASH-1400 study) as compared to global experience in five meltdowns of around 10,000 hours resulting in 5E-4. Mosleh then introduced an intermediary class of PRA, the hybrid methods, which expand the capabilities of conventional methods, but are not quite dynamic simulations. One method referred to in particular is the Hybrid Causal Logic (HCL) method [58], which expands the conventional ET/FT by providing casual factor details to the FT through the use of a BBN to use its flexibility to incorporate the "human, organizational, and social-technical roots" into causal factors of the events within the FT [46]. Even though they may not account for system behavior over time, the hybrid models do provide greater detail than conventional methods. This concept of hybrid methods has been mirrored by others specifically interested in digital systems.

In 2013, Chu et al. prepared the NUREG/CR-7044 [59], which provides a discussion of two candidate methods for software reliability analysis—a test-based method and a BBN method. They appeared to have high potential for meeting many of the desirable attributes previously mentioned. Then, in 2018, they furthered their investigations using BBN for quantifying software failure of protection systems at NPPs [50]. This work incorporates design requirements and software life cycle development processes, but in its initial demonstration, it required expert elicitation for many of its processes. Like many BBN-based methods, a lack of data tends to drive the need for expert elicitation.

In this work, a novel method is developed for quantifying software failure probabilities. The approach covers a novel application of previous works by the implementation of BBN methods in combination with human reliability analysis (HRA) for the quantification of FT basic events identified using STPA-based methods. Lastly, the method addresses modeling CCF as an option for the analysis.

2.2.1 Application of Human Reliability Analysis

An investigation into causal factors of software failure led to several important findings. First, according to some [2] [60], software doesn't fail. At least, the software code does not fail randomly like mechanical components. Software performs exactly how it has been designed to perform, any unwanted action or behavior is due to a fault within the code that has been activated based on certain inputs. An error or mistake in the software development will result in a fault (e.g., a bug, wrong settings, wrong parameters, etc.) that will remain idle until activated. Once activated, the fault may result in a failure of the software to perform as desired. Some faults are benign, such as a typo, whereas others can be quite serious. The main reason for unwanted software behavior is due to improper design requirements, verification, validation, and human errors in the software development life cycle (SDLC).

A human error in the SDLC will result in a fault in whatever system is being created. For years, the quantification of human actions has been the subject of HRA [61]. There have been quite a number of methods developed for HRA over the years; some of the more well-known methods include the Technique for Human Error Rate Prediction (THERP) [61], the Standardized Plant Analysis Risk-Human (SPAR-H) reliability analysis method [62], and the Cognitive Reliability and Error Analysis Method (CREAM) [63]. HRA tends to deal with human error in three ways. The first focuses on those actions completed prior to an event (i.e., errors in maintenance, testing, or calibration). The second deals with actions made that are the direct initiator of an event. And the third is related to human actions made after or in response to an event [64]. Because most software faults are associated with faults in SDLC, the HRA applications used in BAHAMAS will be of the first type. The use of HRA typically provides a human error probability for an action performed. It is assumed that the human error probability is the source of any faults within the software. The probability of human error will be used in a BBN as a root cause of faults within the software, and ultimately, as a source of software failure.

2.2.2 Application of Bayesian Belief Network

The BBN is an acyclic graphical approach to model relationships between parent and child nodes [4]. BAHAMAS takes a basic event from a FT (e.g., a child node) and models the influence of causal factors (e.g., parent nodes) on the child node in the form of a BBN. For the BBN evaluation, probabilities must be determined for causal factors, as well as the conditional influence the causal factors have on a basic event. Experience with the construction and evaluation of BBN has emphasized that a challenge of the BBN is uncertainty associated with the nodes and their conditional relationships. It is assumed that the BBN can be constructed such that all but the final conditional probabilities can be assigned the binary probability of "1" or "0." This is completed by assuming that any software failure, whatsoever, is due to a fault. Software defects, requirements defects, installation errors, etc., can all be considered as "faults." Therefore, the root nodes can represent all sources of a fault, while the remaining BBN simply combines the results into "a probability of faults existing in the system," as shown in Figure 2. This construction effectively limits the uncertainty from conditional relationships to the final two nodes of the network.

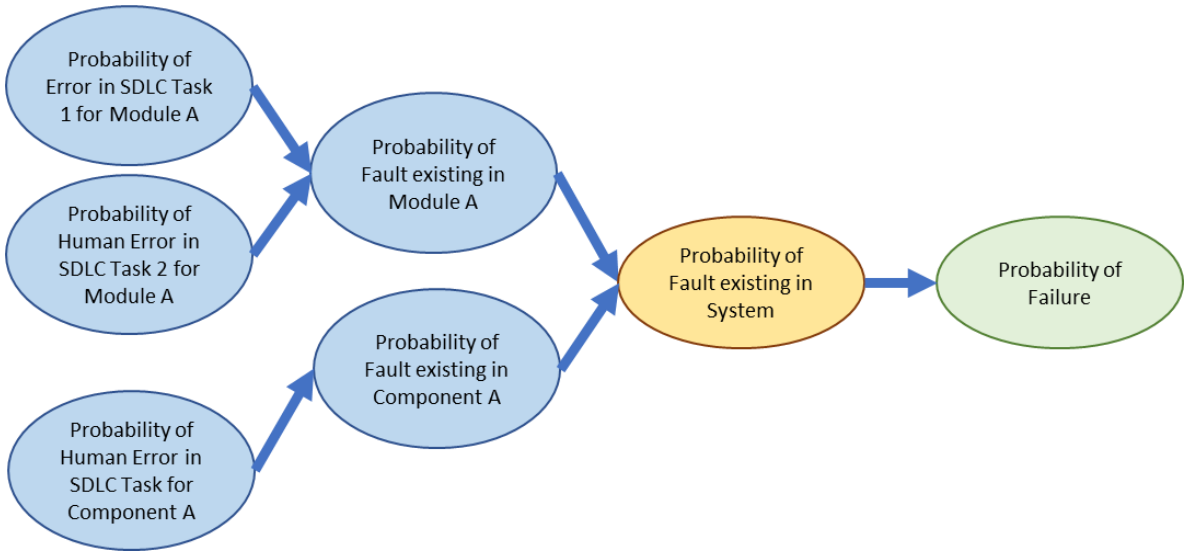


Figure 2. Basic BBN format: root nodes are furthest to the left, while the end node is furthest to the right.

The conditional relationship between the final two nodes requires more consideration. The conditional relationship between fault probability and failure probability must be assigned. Research by Chu et al. [50], Kang et al. [65], and Lee et al. [66] demonstrates a method for transforming the number of faults existing in a software after its SDLC into probability of failure on demand. In their work, software failure probability (SFP) is related to the number of faults in a software at the end of development by a fault size distribution (FSD) parameter. The FSD serves as a proportional relationship between SFP and the number of faults at the end of SDLC. This relationship is given in Eq. (1):

$$SFP = N_{final\ fault\ number} * FSD \tag{1}$$

Their method is completed in two parts. The first evaluates the SFP based on operational experience of similar software systems. In the second, a SDLC analysis method is used to determine the number of final fault numbers in a "generic" software that assumes or represents the same quality in verification, validation, and other SDLC activities as the sample set for SFP. Then using Eq. (1), the FSD can be found, which is then used for specific cases to evaluate the SFP.

$$SFP_{specific} = specific\ N_{final\ fault\ number} * FSD \tag{2}$$

Upon closer inspection, BAHAMAS is not far removed from the $N_{final\ fault\ number}$ because it relies on the "probability of faults" existing within a software. In fact, the "probability of faults" might be thought of as:

$$Pr(faults) = \frac{N_{final\ fault\ number}}{Total\ Faults\ introduced} \tag{3}$$

Via substitution we can create a similar equation to Eq. (1).

$$SFP = Pr\ (faults) * new\ parameter \tag{4}$$

The application of these equations requires several things: (1) operational or testing data must be gathered to provide the generic SFP; (2) the BBN must be created for the generic software case and evaluated for the generic Pr $(faults)$; (3) the $new\ parameter$ can be determined; (4) the BBN can be assigned probabilities for the specific case and evaluated for the specific Pr $(faults)$; and (5) the specific Pr $(faults)$ and $new\ parameter$ can be used to solve for the specific SFP.

2.2.3 Modeling of Common Cause Failures

The definition of CCFs varies. But for our purposes, a CCF event is the occurrence of two or more failure events “simultaneously” due to a shared cause [1] [67] [68]. For a CCF to occur, there must be some link between the components shared event. Some researchers break CCFs into root causes and coupling factors [69], [70]. The root cause is the cause of the event while the coupling factor is the link or reason why the root cause resulted in a CCF event [71].

Root causes might include design requirements, manufacturing mistakes, a lack of set procedures, and environmental conditions. Coupling factors might include identical design, hardware, installation staff, procedures, environment, location, etc. When a root cause and coupling factor are known, they can be used for the explicit modeling of CCFs. However, the number and variations of root causes and coupling factors can quickly become unmanageable in PRA [71]. Implicit CCF modeling provides a way to simplify the modeling of CCFs and lead to more manageable analyses. According to [71], guesses for explicit models might be better than choosing to use an implicit model exclusively. And both implicit and explicit models can be used in the same PRA. If possible, at least some potential CCFs should be modeled explicitly [71]. But the rest may be modeled using implicit methods. The selected implicit methods depend on the assumptions made for the risk assessment mainly surrounding the multiplicity of CCFs to be modeled (m/n or all n components failing together).

There are a number of implicit CCF modeling techniques. The most common is the beta factor method [71]. With the beta factor method, CCFs are modeled to result in the failure of all n/n components. There is no consideration for the multiplicity of CCFs. There are methods that deal with multiplicity (e.g., multiple Greek, multiple beta factor [71]), but they do not explicitly indicate which of the identical components may fail together. They only express a failure as a probability for resulting in some m/n components. For the initial application of this work, CCF analysis will neglect the multiplicity of common failures and rely on the commonly used beta factor method.

3. REDUNDANCY-GUIDED SYSTEM-THEORETIC HAZARD ANALYSIS

Section 3.1 describes the details of the proposed approach for RESHA. Section 3.2 and Section 3.3, respectively, describe the demonstration on a representative digital RTS and ESFAS.

3.1 Approach Description

To deal with the complexity problem of redundancy and identify software CCFs effectively, the system-theoretic hazard analysis is proposed to integrate and reframe the STPA process in a redundancy-guided way as a seven-step process, the key outcomes of which are an integrated FT, including software failures and hardware failures, identified CCFs, and minimal cut sets to discover the single points of failure (SPOFs) leading to the loss of function of the entire digital system. SPOF refers to a situation in which a single part of a system fails, and the entire system loses function as a result. The proposed RESHA approach is illustrated in Figure 3. The steps of the RESHA approach are briefly described below.

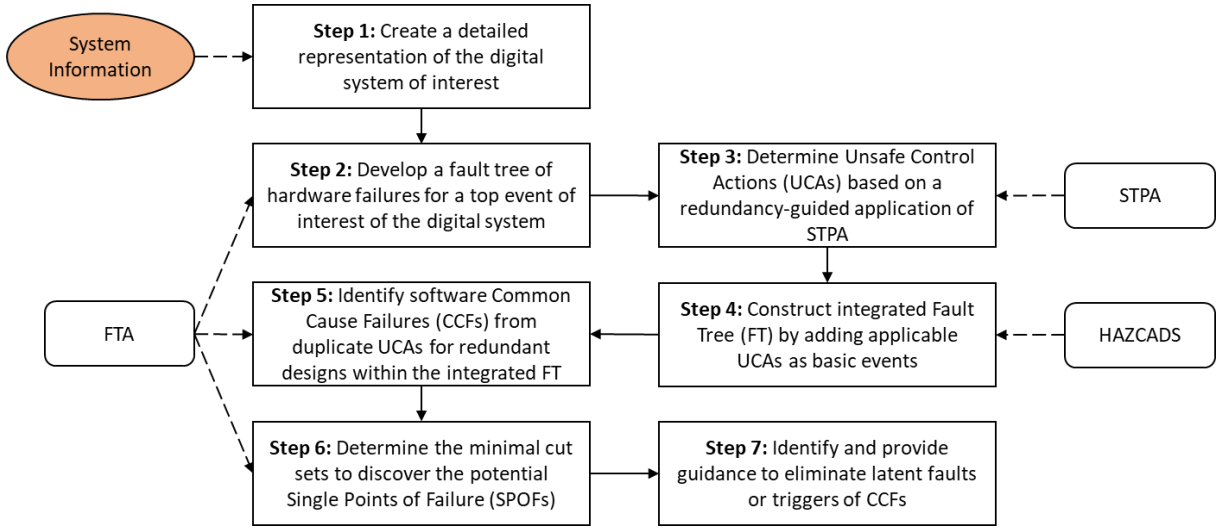


Figure 3. Workflow of the proposed RESHA approach.

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

The purpose of Step 1 is to establish a system sketch to serve as a blueprint for the hazard analysis. This is done by gathering system design information, including wiring, piping and instrumentation diagrams, existing logic diagrams, etc. This information is then used to create a system sketch, the main goal being to map out the processors, sensors, controllers, components, interactions, and connections of the system. The point of this step is not to necessarily fit everything into one diagram, but to gain a sufficient understanding of the system in order to complete the hazard analysis; the level of detail provided in this step lays out the foundation for the work.

In this step, detailed information on the structure and functions of the digital system of interest should be collected, gathered, and classified. Normally, a DI&C system has a three-level hierarchical architecture [43]: (1) divisions that process the signal path from sensor to actuator level (e.g., the four-division design in APR-1400 ESFAS), (2) units that perform a specific task by using several modules (e.g., an acquisition and processing unit or a voter unit), and (3) modules that realize a specific part of the function processing (e.g., input/output modules, processors). The representation should contain information on hardware structure and be created to a detailed level that captures sufficient design information affecting system function and reliability. In this work, most efforts on hazard identification and reliability modeling reach

to the level of modules, which is the smallest hardware component to implement a specific part of the entire function processing independently. In addition, based on the requirements and purposes of the risk analysis phase, practical assumptions and reasonable simplifications of the hardware representation should be stated and explained in this step. The representation figure should clearly display the information flow between different divisions, units, and modules. For the analysis on digital systems with redundancy designs, the complexity of redundancy should be illustrated in the hardware representation. It builds the basis for the construction of hardware FTs and redundancy-guided multilayer control structure.

Several key points are considered for Step 1:

- Step 1 emphasizes the boundary conditions and scope of the analysis. These should be clearly understood as they will be revisited in Step 2 and Step 3.
- Though the RESHA has been developed to analyze digital systems, this system sketch should also include the hardware structural arrangement (i.e., the components of the system in addition to details collected for the digital structural arrangement).
- The level of detail included in a hazard analysis can extend beyond module level failures to the components and sensors providing input to process modules. The level of detail to be included in the hazard analysis depends on the scope of the investigation.

Step 2: Develop an FT of hardware failures for a top event of interest of the digital system.

Based on the hardware representation created in Step 1, a FT is developed in this step to include hardware failures to the detailed level required for representing the loss of functions. For analysis of a digital system with redundancy designs, the structure of a hardware FT should follow the levels of redundancy from the division to the unit and to the module levels. This kind of redundancy-guided structure makes it convenient to add in a software failure identified in the next step. The probability quantification of each basic event is not required in this step and will be performed in the integrated reliability analysis. The main assumption for this step is that all software failures will be captured using STPA in Step 3. Therefore, only hardware failures will be included in FT, which is created using the two-part process adapted from the U.S. National Aeronautics and Space Administration (NASA) Handbook [42].

Step 2A: Define the boundary of the analysis (revisited from Step 1). This includes selecting a top event and resolution for the analysis. Top events are based on the purpose of the system of interest (SOI). The failure of the SOI's most significant function is a priority event to be analyzed by the FT—a top event. Step 3A may also be visited briefly to ensure the proper selection of top events.

Step 2B: Construct the FT. Starting from the top event, construction proceeds by determining the "necessary and sufficient immediate events" contributing to failure of the top event [42]. This process is repeated down the tree by analyzing each subsequent event step-by-step, ending with an event that can be resolved no further (either by way of fact, or by the discretion of the analyst) [42]. This event is known as a basic event. The failure of each event depends on a logical combination (e.g., "and," "or," and "n/m" functions) of basic events, which should be built into the FT based on the system diagram from Step 1. The structure of the FT should capture the details of redundancy that will aid in the subsequent steps of the hazard analysis. In most cases, the highest level of redundancy will be associated with protecting the main function of the SOI. For instance, a system may have two or more divisions, independent and redundant in function, to ensure the reliability of that system. Redundancy also extends to the units and modules of digital systems, for which commonality becomes a possible source of CCF. The hardware CCFs should also be included in the FT. Software details will be added in Step 3 of the RESHA.

Step 3: Determine Unsafe Control Actions (UCAs) based on a redundancy-guided application of STPA.

In this step, part of the STPA process is applied to identify the UCAs as potential software failures. First, based on the requirements and purposes specified in Step 1, key losses and system-level hazards are identified. In STPA, a loss impacts something of value to stakeholders or the public (e.g., a loss of human life or a human injury, property damage, environmental pollution, or any other loss that is unacceptable) [43]. A hazard is defined as, "a system state or state or setoff conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss" [38]. The identification of hazards is tightly connected to the function and operating requirements of the SOI.

Second, according to the redundancy information in the hardware FT, a redundancy-guided multilayer control structure is modeled. A control structure is defined as, "a system model composed of feedback control loops," [38] which illustrates the interactions between controllers and a controlled process, including sensors and actuators. A generic control loop is shown in Figure 4. Generally, controllers provide control actions to conduct certain processes. A controller includes control algorithms representing a controller's decision-making process while a process model represents the controller's internal criteria used for its decision-making. The actions provided by a controller can be influenced by the controller's process models, control algorithms, and feedbacks.

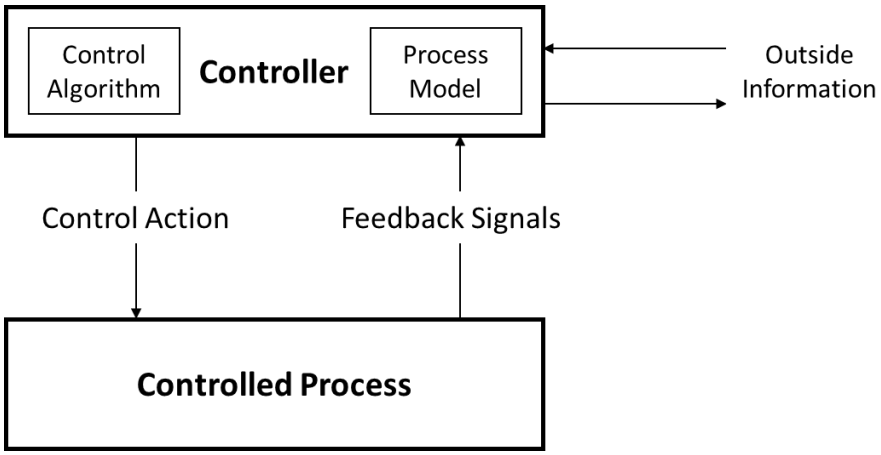


Figure 4. A generic control structure in the STPA application.

In a digital system, all information exchanges—including the decision-making process of the controllers, control and implementation of control actions, performance of controlled process, and feedbacks from controlled process—have a potential to fail the function of the digital system when it is needed or send spurious signals that are not needed. These systematic failures could be initiated by the UCAs as a result of an unrealistic process model, an inappropriate control algorithm, an incorrect feedback, or outside information. Therefore, the potential software failures can be understood and analyzed by identifying these UCAs. To deal with the complexity problem of redundancy and to identify software CCFs effectively, control structure is built in a redundancy-guided way. The redundancy-guided multilayer control structure zooms in on systematic information exchanges on each redundancy level because CCFs are tightly connected with redundancy designs. Each control structure layer is created with numbered control actions and feedback signals until a final, redundancy-guided, multilayer control structure is created for the complete system of interest, as shown in Figure 5.

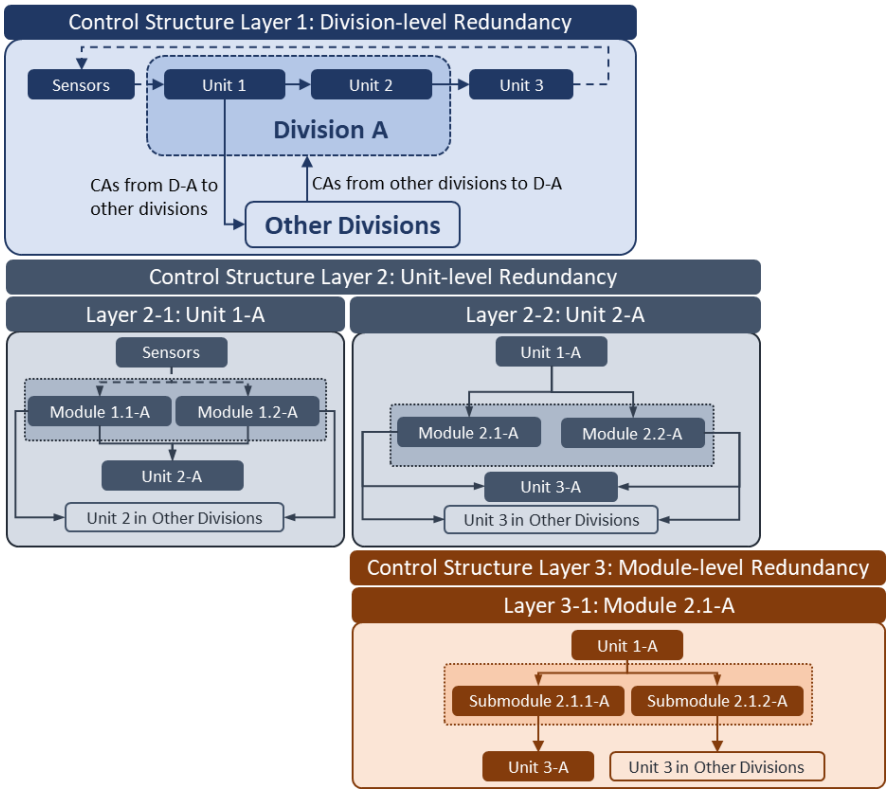


Figure 5. Illustration of a multilayer control structure.

Third, the UCAs are identified according to the multilayer control structure and specified hazards. A UCA is defined as, "a control action that, in a particular context and worst-case environment, will lead to a hazard" [38]. There are four types of UCAs in an STPA:

- Control action is not provided when it is needed.
- Control action is provided when it is not needed.
- Control action is provided when it is needed, but too early, too late, or in a wrong order.
- Control action lasts too long or stops too soon (only applicable to continuous control actions).

Each UCA should take the following format for consistency [38]:

$$\boldsymbol{UCA} = [\boldsymbol{source}] + [\boldsymbol{UCA\ type}] + [\boldsymbol{Control\ Action}] + [\boldsymbol{Context}] + [\boldsymbol{Link\ to\ System\ Hazards}] \quad (5)$$

The specification of the context for UCAs is important, usually words like "when," "while," or "during," are used to define the context. The UCA context should represent an actual or true condition that would make the control action unsafe, not a controller process model that may or may not be true.

Step 4: Construct an integrated FT by adding applicable UCAs as basic events.

In this step, applicable UCAs are selected and added into the hardware FT as the software failures. For a specific top event in the FT, some UCAs may be inapplicable. For example, if the top event of hardware FT is "ESFAS fails to actuate ESF components," Type 2 and 4 of UCAs are inapplicable since the control action of "sending actuation command" is needed, and not a continuous action. If the top event is "unexpected actuations by ESFAS," only Type 2 is applicable. Considering the hardware FT and redundancy-guided multilayer control structure are tightly connected and consistent with each other, these applicable UCAs (software failures) can be incorporated into the hardware FT in parallel with the respective hardware failures. Figure 6 provides an example of this structure.

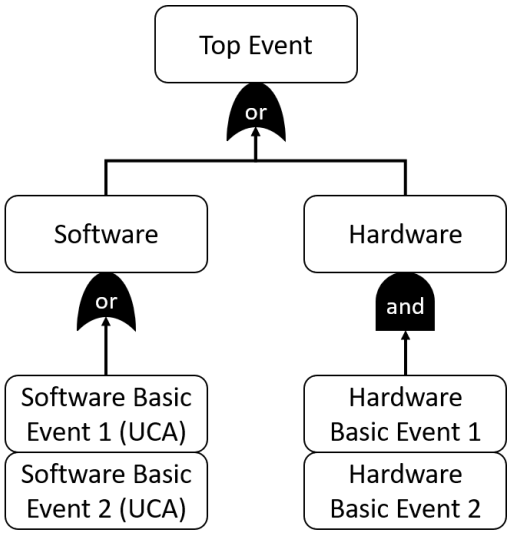


Figure 6. Example FT format for incorporating software- and hardware-based failures.

Step 5: Identify software CCFs from duplicate UCAs for redundant designs within the integrated FT.

After integrating UCAs into the hardware FT, the same types of UCAs, located in the same redundancy level, can be separated into independent failures and CCFs. Additionally, software CCFs can be classified into different types depending on the redundancy levels: (1) software CCFs occurring in all divisions; (2) software CCFs occurring in all of the units in one division; and (3) software occurring in all of the modules in one unit. The classification of software CCFs depends on the software diversity of the digital system. As one of the guidelines for the DiD analysis, software diversity should be considered. Software diversity is defined as, "the use of different programs designed and implemented by different development groups with different key personnel to accomplish the same safety goals; for example, using two separately designed programs to compute when a reactor should be tripped" [10]. Therefore, before the identification of software CCFs, the level of software diversity should be one of the key assumptions to guide the classification of software CCFs.

Step 6: Determine the minimal cut sets to discover the potential SPOFs.

As the main outcome of the systematic-theoretic hazard analysis, the minimal cut sets of the integrated FT should be calculated and evaluated to determine how many potential SPOFs have been added by considering the software failures. If the digital system has a low level of software diversity, the software CCF types occurring in all divisions could lead directly to the top event (e.g., the loss of function of the entire digital system), regardless of the contributions from other safety designs. As a part of risk analysis, hazard analysis directly provides evidence to evaluate the question, "Does the individual digital failure lead to the loss of function of the digital system?" If the individual digital failure is one of the SPOFs, a redesign request will be made for system designers and engineers based on the risk evaluation results.

Step 7: Identify and provide guidance to eliminate latent faults or triggers of CCFs.

A dormant fault does not affect safety before a triggering condition or event activates it to a failure. Triggers include plant transients, initiating events, external conditions, interactions among systems, human interactions, and internal states. Two main software faults identified by the NRC and EPRI were inconsistent with the system requirements specification [72], as well as the faults introduced during the detailed logic-design phases of software development due to interactions between some process logic inhibits and the test logic not being recognized by the designers or verifiers [73]. The NRC proposed two

design attributes to eliminate CCFs: diversity and 100% testability [74]. Diversity is applied to mitigate the potential for common faults and ensure safety using different or dissimilar means in technology, function, and implementation. With respect to 100% testability, the NRC stated, "If a portion or component of a system can be fully tested, then it can be considered not to have a potential for software-based CCF. Fully tested or 100% testing means that every possible combination of inputs and every possible sequence of device states are tested, and that all outputs are verified for every case" [74]. However, both design strategies have limitations. Diversity normally leads to higher costs, while potential CCF vulnerabilities will be more complicated and difficult to identify as system complexity increases. Applying 100% testing may reveal the presence of a fault, but not its absence, which means 100% testing does not fully eliminate software CCF concerns.

Therefore, this step focuses on identifying and providing guidance to eliminate the potential latent faults or triggers of CCFs and other independent failures based on the redundancy-guided STPA application in previous steps. The faults and triggers for hardware CCFs or independent failures can be identified in a straightforward manner. For software CCFs and independent failures, once the respective UCAs are obtained, their causal factors or latent faults can be placed into one of two categories: (1) unsafe controller behaviors (i.e., operator errors, power failure of digital controllers, or a pressurizer setpoint that is not correctly programmed in BPs) or (2) inadequate feedback or outside information (i.e., wrong or absent signals sent from the pressurizer to ESFAS). The triggers for software failures are defined as the contexts of the identified UCAs. The identification of causal factors should be interpreted by expert teams in system and software engineering, human reliability analyses, etc., and would be helpful to provide guidance for risk reduction and redesign of the digital systems.

The categories serve as a starting point for analyzing the results from previous steps. However, the impact of the results of Step 7 will depend largely on the knowledge and skill of the analysis team for determining the details of the causal factors. As each causal factor is identified, efforts can be made to evaluate or mitigate them by applying D3 or redesigning the system. The initial efforts in identifying causal factors may result in a resource bank of typical causal factors that can be used to expedite future analyses, thus reducing the cost of the RESHA over time.

3.2 Demonstration on Digital Reactor Trip System

This Section describes the hazard analysis of a four-division digital RTS, which has a similar structure to state-of-the-art digital systems in existing NPPs [75]. The analysis follows the seven-step process outlined in the preceding Section. The following is a list of assumptions built into the analysis:

- All RTS components are assumed to be digital, and are therefore susceptible to software failures. The reactor trip breakers (RTBs) are an exception; they are assumed to be analog, having only hardware failures.
- The functions of an RTB are controlled by two mechanisms: an undervoltage (UV) trip mechanism and a shunt (ST) trip mechanism; the reactor protection system (RPS) activates the UV while the diverse protection system (DPS) activates the ST. The main control room (MCR) and the reserve shutdown room (RSR) are assumed to have an alternate method that causes the RTBs to open.
- The MCR and the RSR only need to make a single control action to trip the reactor. They do not trip individual divisions.
- The controller in the MCR/RSR is an operator. The UCAs associated with the human controllers are added to the FT in the same manner as the UCAs associated with the software failures of digital controllers.
- The number of CCFs in the FT is limited to common failures within a single division, as well as CCFs across all divisions. No CCFs are provided for combinations of divisions. Limiting the combinations of CCFs, in the manner specified, simplified the model without removing the most

significant SPOFs caused by CCFs. Assuming failures had to occur across all divisions ensured that the analysis would capture a SPOF of the RPS due to CCFs. Including all other interdivisional CCF combinations (e.g., A-B, ABC, CBD, etc.) would provide an increase in cut sets, but is beyond the scope of this case study. It is assumed acceptable to ignore the SPOFs due to CCFs within sub-divisions or modules.

- RTS component and software diversity is ignored. All components that have identical functions are assumed to be identical components (i.e., there is no diversity provided in the system). Limiting diversity simplifies the identification of CCFs and helps to direct the use of safety features, such as diversity, to mitigate CCFs. It should be noted that if diversity is known, it can be included. However, the analyst should refer to NUREG/CR-6303, NUREG/CR-7007, NUREG-KM-0009, and NUREG/CR-5485, as necessary, to make informed decisions regarding the adequacy of diversity and where to include CCFs in the FT [9] [10] [14] [19].
- Each control action (e.g., trip signal) originates from an independent action within the controller rather than a single control action that is electrically or physically split to multiple destinations. This was done to follow the STPA process specifically, which assumes nothing regarding the physical connections between controllers. This provides the most unbiased controller requirements for the designers to work with. Decisions regarding how to distribute or duplicate control actions can be a point of defense in a system design. STPA allows the designer to make decisions how best to defend the system after the fact. The trip signal from the bistable processors (BPs) to individual logic processors (LPs) is the one notable exception. These connections, not shown in the control structure, are assumed to be physically split and distributed to each LP. This simplified the analysis from 512 potential UCAs associated with the BPs to 128.
- No bypassing function, maintenance work, or corrective actions are considered in the analysis.
- The hazard analysis assumes the RTS is monitoring an NPP operating normally at 100% power, with the control rods completely withdrawn and full power to the turbines prior to the anticipated operational occurrence (AOO).
- Individual sensor failures are grouped into one single basic event per division (e.g., division A sensor failure).

Step 1: Create a detailed representation of the SOI.

The RTS is responsible for controlling the automatic insertion of reactor control rods into the core to bring the nuclear reaction to a shutdown state. For the sake of this hazard analysis, the RTS contains four main controllers capable of causing reactor shutdown: (1) the MCR; (2) the RSR; (3) the DPS; and (4) the RPS. The MCR and the RSR are manual components, while the DPS and RPS are automatic. The automatic control elements of the RTS maintain four divisional redundancy, while the manual components (assumed for this analysis) have no divisional redundancy. The case study is simplified by narrowing the analysis to the module level detail of the RPS and divisional or higher resolution for the remaining three parts of the RTS, as shown in Figure 7.

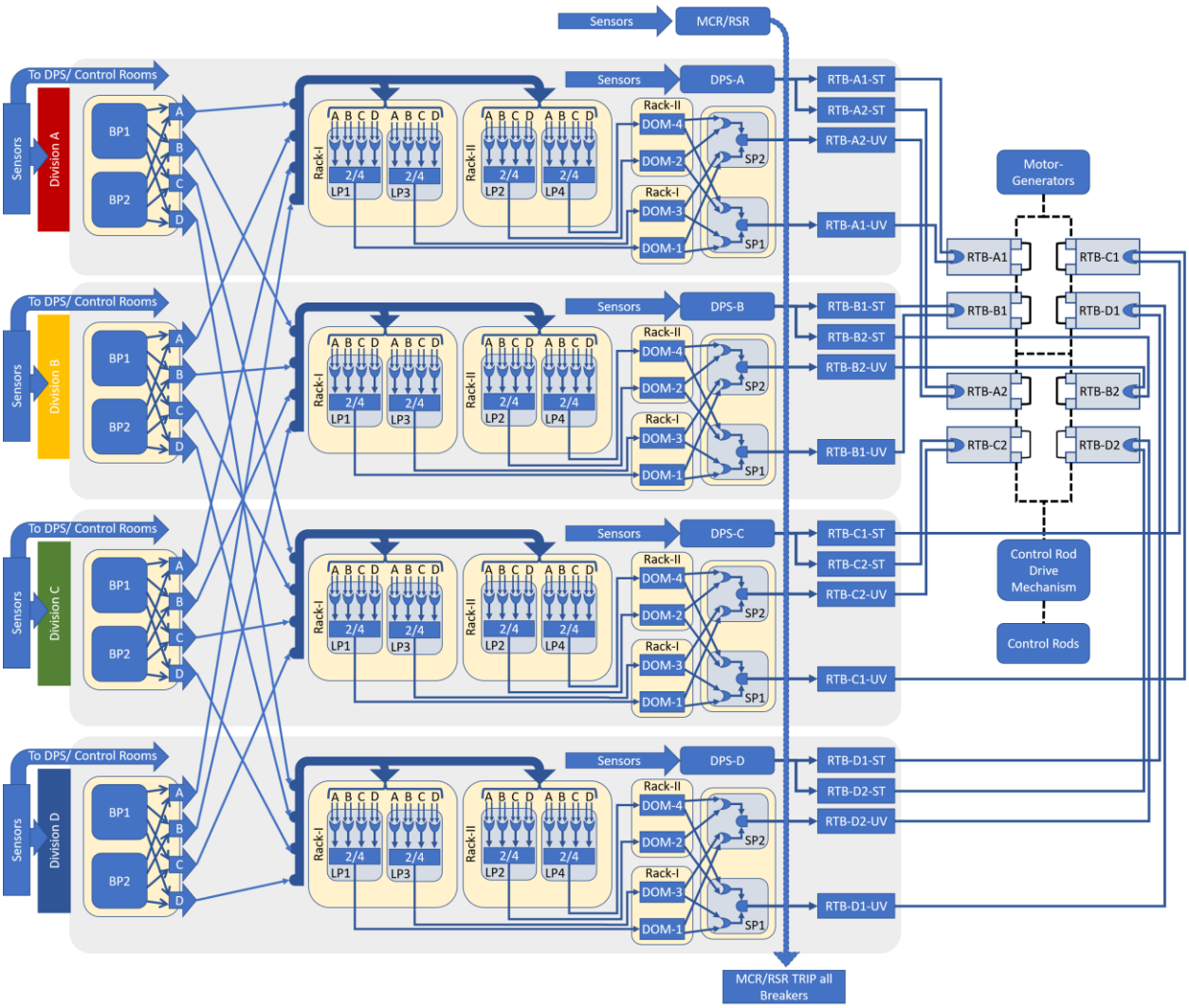


Figure 7. Detailed representation of the RTS.

Step 2: Develop a FT consisting of the hardware failures for a chosen function of the SOI.

Step 2A: The most significant function of the RTS is to trip or scram the reactor by rapidly inserting the control rods. The top event selected for the FT is therefore chosen to be that the RTS fails to trip the reactor during an AOO, which may be any event that can be anticipated for the reactor requiring shutdown via trip or scram.

Step 2B: The FT is assembled based only on the hardware failures of the RTS components, according to the process described previously. The FT should also include hardware CCF basic events for every common or redundant hardware component of the RTS. Figure 8 shows a portion of the hardware-based FT. Note that the structure also includes the logic indicated by the system sketch.

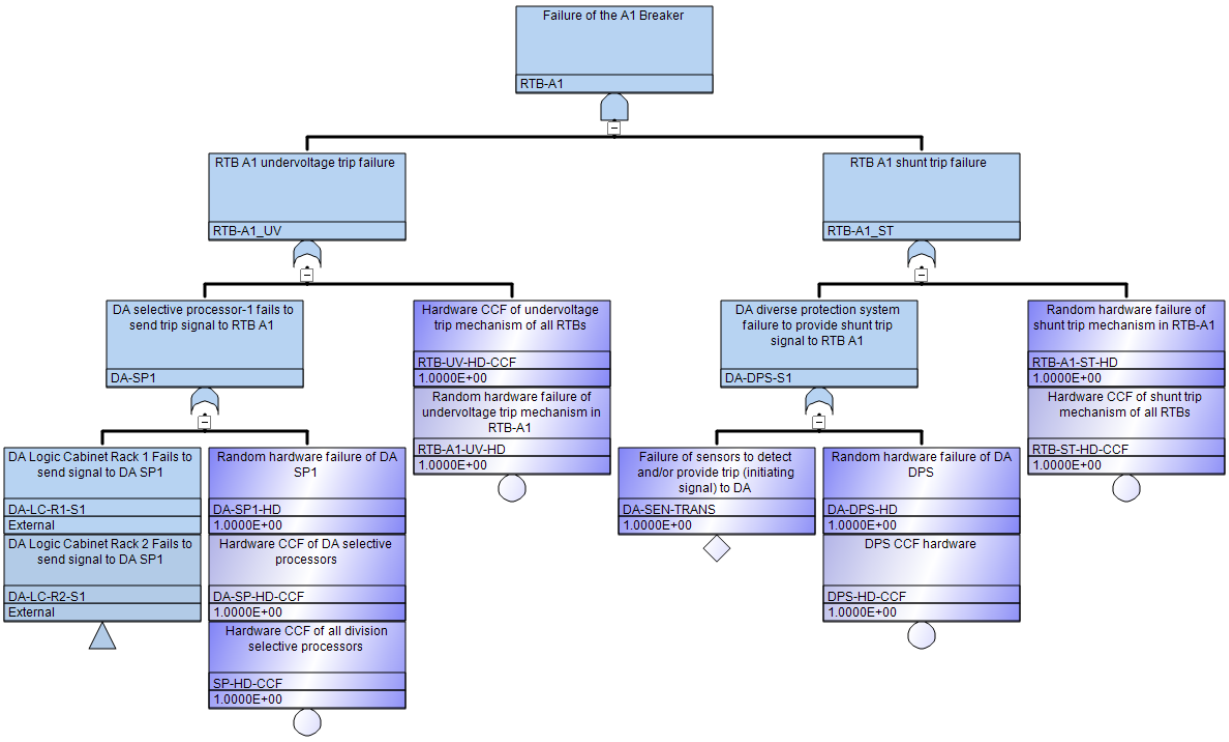


Figure 8. Portion of RTS FT showing hardware-type failures only. The U.S. NRC PRA software SAPHIRE was used to construct the FT [76].

Step 3: Determine UCAs based on a redundancy-guided application of STPA.

Step 3A: In order to gain a clearer understanding of the system, tables of losses and hazardous states that can lead to those losses are created. The RTS is designed to prevent these hazardous states from occurring and ultimately prevent the losses in the table. The hazardous states demonstrate system-level hazards and provide context for UCAs to be identified later. Table 1 and Table 2 present the losses and system-level hazards, respectively, for the RTS.

Table 1. Major losses to be prevented.

L1	Human injury or loss of life
L2	Environmental contamination
L3	Equipment damage
L4	Power generation
L5	Public perception

Table 2. Hazards that may lead to losses.

H1	Reactor temperature too high (L1, L2, L3, L4, L5)
H2	Equipment beyond limits (L1, L2, L3, L4, L5)
H3	Release of radioactive materials (L1, L2, L5)
H4	Reactor shutdown (L4, L5)

Step 3B: A redundancy-guided multilayer control structure is created. Each layer of redundancy in the RTS is used to create a control structure diagram. The first layer of the RTS contains the redundancy for the trip function across four diverse controlling subsystems: MCR, RSR, DPS, and RPS. The second layer of redundancy is found in the multiple divisions of the DPS and the RPS. The next layers of the control

structure come from the units and modules found in the RPS. The DPS sub-divisional redundancy is ignored to simplify the case study. Figure 9 shows the multilayer control structure for division A. The remaining divisions for the case study are identical in format. Each structure indicates control actions and feedback in the system. The unknown control actions (i.e., CAx in Figure 9) are populated as they are discovered by layer.

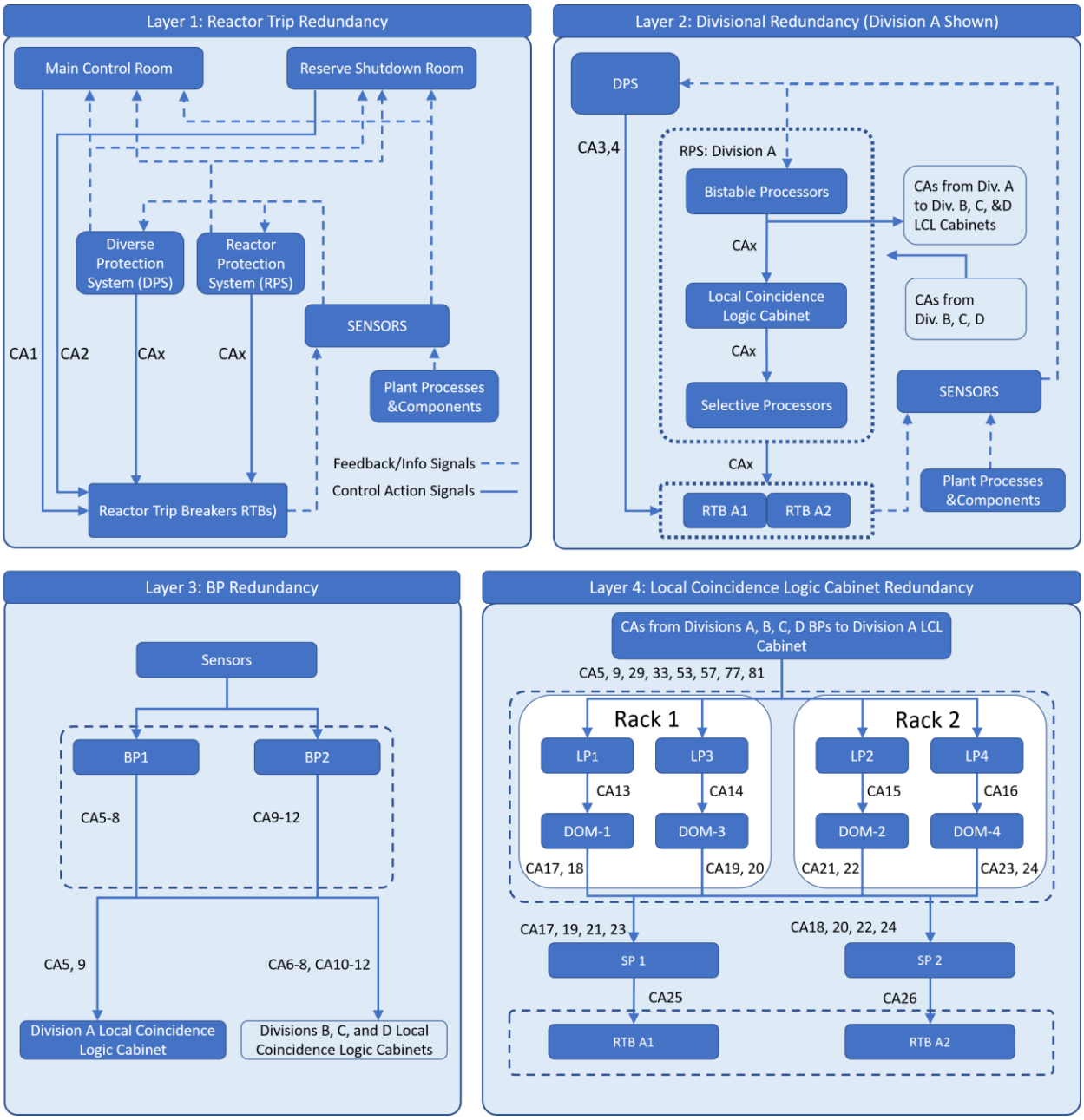


Figure 9. Redundancy-guided multilayer control structure.

Step 3C: The control actions found in each control structure are compiled in a table and analyzed according to the four UCA categories. Each UCA follows the format provided previously. Table 3 shows an example of the UCA table. It is assumed that Case D is not applicable, as a trip command applied too long should not matter during an AOO. Also, it is assumed that a trip command cannot be stopped too soon, as it is not a continuous controlling action.

Table 3. Examples of UCAs.

Control Action (CA)	**UCA-a: CA is needed, but not given**	**UCA-b: CA is given, but not needed**	**UCA-c: CA is given too early, too late, or in the wrong order**	**UCA-d: CA is applied too long or stopped too soon**
CA18: DOM-1 demands SP1 to trip the reactor	UCA18a: DOM-1 does not provide trip command to SP1 during AOO [H1, H2, H3].	UCA18b: DOM-1 provides trip command to SP1 when there is NO AOO [H4].	UCA18c: DOM-1 provides trip command to SP1 after AOO has existed for some time [H1, H2, H3].	UCA18d: Not applicable.
CA20: DOM-3 demands SP1 to trip the reactor	UCA20a: DOM-3 does not provide trip command to SP1 during AOO [H1, H2, H3].	UCA20b: DOM-3 provides trip command to SP1 when there is NO AOO [H4].	UCA20c: DOM-3 provides trip command to SP1 after AOO has existed for some time [H1, H2, H3].	UCA20d: Not applicable.
Note: AOO: Anticipated Operational Occurrence; DOM: Digital Output Module; SP: Selective Processor.				

Step 4: Construct an integrated FT by adding applicable UCAs as basic events.

The top event chosen for this case study is that the RTS fails to trip the reactor during an AOO. Based on this assumption, the appropriate UCAs are chosen (types: UCA-a and UCA-b) to populate the FT with software failures. Figure 10 shows the software failures added to the FT and the update from Figure 8.

Step 5: Identify potential software CCFs based on duplicate or redundant UCAs within the FT.

Within the FT, there are redundant or duplicate control actions that can be used to add additional software CCF basic events. The FT in Figure 10 shows the basic events of UCA Types A and C, as these were deemed appropriate to add to the FT in the previous step. These UCAs have commonality across redundant divisions or units. For example, Division A Selective Processor 1 from Figure 10 is the same component as Division A Selective Processor 2 (unit redundancy) and the same component as Division B,C, and D Selective Processors (divisional redundancy). Each of these common components can be seen in Figure 7. Assuming all of these components are from same manufacturer, with identical software and functionality, they have the potential for a CCF. In the FT, under each category for software failures, the newly identified CCF events are added. CCFs in this model are assumed to occur within a single division and across all divisions. There are no subsets of CCFs between divisions (e.g., A-B, B-C, C-D, etc.) Figure 10 shows the software-based CCFs that have been added to the FT.

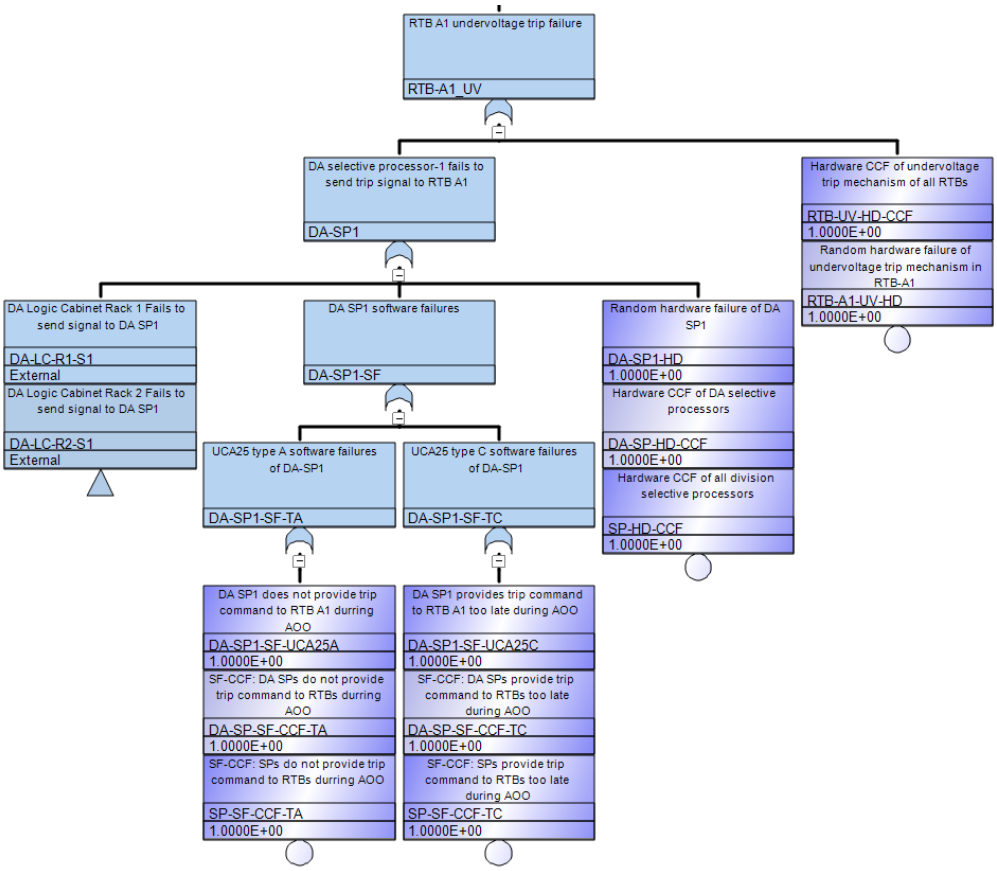


Figure 10. Portion of FT showing the UV trip failure of RTB A1 with software failures and CCFs added.

Step 6 Solve the FT for the minimal cut sets to determine potential SPOFs in the design.

The minimal cut sets were found using the SAPHIRE software program (e.g., the same program used to create the FT). To evaluate and provide a list of cut sets without having failure data, the cut sets were listed and truncated based on order rather than by probability. For this case study, the number of basic events contributing to the failure of the top event is limited to a truncation of six or fewer basic events. Table 4 contains results for the FT for varying conditions: (1) complete RTS; (2) hardware only failures; (3) automatic trip functions only; and (4) RPS only.

Table 4. Cut set results.

Truncation (order)	Full RTS model	RTS hardware only	Automatic trip only	RPS only
None	N/A	15234	N/A	N/A
6	1,184,652	-	4,583,568	N/A
5	85788	-	1,038,956	328,355
4	468	-	13,1628	54,899
3	0	-	9,532	15,283
2	0	-	52	1,203
1	0	-	0	13

Based on the information shown in Table 4, only the final column has any SPOFs. This result was obtained by ignoring the diversity usually provided by the DPS, RSR, and MCR. Each of these could have also been evaluated singularly for SPOFs, but the case study was simplified purposefully. Results for the RPS are given in Table 5. A total of 13 SPOFs exist—five that are hardware-based and eight that are software-based. All of them are due to CCFs (see footnote 5 in Section 3 regarding CCFs and SPOFs).

Table 5. First-order cut sets or SPOFs for the RPS system, UV trip only.

Number	Cut set	Description
1	SP-HD-CCF	Selective processor hardware CCF.
2	LC-DOM-HD-CCF	Logic cabinet digital output module hardware CCF.
3	RTB-UV-HD-CCF	Reactor trip breaker undervoltage hardware CCF.
4	LC-BP-HD-CCF	Logic bistable processor hardware CCF.
5	LC-LP-HD-CCF	Logic cabinet logic processor hardware CCF.
6	LC-LP-SF-CCF-TA	Logic cabinet logic processor software CCF Type A.
7	LC-LP-SF-CCF-TC	Logic cabinet logic processor software CCF Type C.
8	LC-DOM-SF-CCF-TA	Logic cabinet digital output module software CCF Type A.
9	LC-DOM-SF-CCF-TC	Logic cabinet digital output module software CCF Type C.
10	SP-SF-CCF-TA	Selective processor software CCF Type A.
11	SP-SF-CCF-TC	Selective processor software CCF Type C.
12	LC-BP-SF-CCF-TA	Logic cabinet bistable processor software CCF Type A.
13	LC-BP-SF-CCF-TC	Logic cabinet bistable processor software CCF Type C.

Step 7: Identify and provide guidance to eliminate triggers of critical failures in the design, including CCFs and SPOFs.

The focus of this step is the CCFs and SPOFs of the system. However, these techniques can be applied to any event of interest found in the cut set lists. The idea is to provide possible sources for their failure. The STPA Handbook indicates that the causes of UCAs can be grouped into two categories: (1) unsafe controller behaviors; and (2) inadequate feedback and/or other inputs [38]. Hardware-based failures are reasonably understood due largely because of industry experience [77]. Identifying causal factors for these failures is therefore based on historical failure rate data. For software-based failures, the analysis can be more challenging and requires the researcher to consider the possible causes of unsafe controller behavior or inadequate feedback. The following is an example of these two categories applied to one of the software CCFs found in Table 5. Specifically, the chosen basic event is LC-BP-SF-CCF-TC (i.e., Logic Cabinet Bistable Processor CCF of Software Type C) corresponding to the failure of all the BPs to provide a trip command to each of the logic cabinets of each division too late during an AOO:

- Causal factors due to category 1: unsafe controller behavior:

  Scenario: The nuclear reactor experiences an event meriting a reactor trip/scram. During this time, the BP should recognize the status of the plant and demand the reactor to trip. Processing delays within the BP result in the CA occurring too late. Software engineers should provide guidance as to the causes of processing delays because shared software can lead to a potential CCF.

- Causal factor due to category 2: inadequate feedback:

  Scenario: A BP may experience failure due to a lack of adequate feedback from the plant leading to the BP having an incorrect view of the status of the plant. For example, the BP relies on steam generator pressure information. This signal for steam generator pressure may be corrupt or incorrect, resulting in the BP failing to act appropriately. The BP may "think" the system is at an appropriate pressure and do nothing for some time before the pressure reaches a value corresponding to what the BP "thinks" is necessary to trip the reactor. Based on the assumption that all BPs have the same software, this would result in a CCF of the system. Of course, the CCF originates in faulty sensors, but also extends to the CA of the BPs and the subsequent UCA.

- Guidance for these two scenarios:

  Scenario: A BP may experience failure due to a lack of adequate feedback from the plant leading to the BP having an incorrect view of the status of the plant. For example, the BP relies on steam generator pressure information. This signal for steam generator pressure may be corrupt or incorrect, resulting in the BP failing to act appropriately. The BP may "think" the system is at an appropriate

pressure and do nothing for some time before the pressure reaches a value corresponding to what the BP "thinks" is necessary to trip the reactor. Based on the assumption that all BPs have the same software, this would result in a CCF of the system. Of course, the CCF originates in faulty sensors, but also extends to the CA of the BPs and the subsequent UCA.

For the second scenario, the proper use of D3 will help eliminate the common failure potential of shared software and hardware. In addition, the use of proper procedural testing and maintenance can ensure the sensors will provide correct information to the BP. Diversity in sensor measurements can also help to ensure safety by providing a backup source of feedback to the BP. Both processes are already commonly employed by the NRC and industry. See NUREG/CR-6303, NUREG/CR-7007, NUREG-KM-0009 and NUREG/CR-5485 to make informed decisions regarding the adequacy of diversity and where to include CCFs in the FT [9] [10] [14] [19].

3.3 Demonstration on the Digital Engineered Safety Features Actuation System

In this Section, the proposed RESHA approach was applied in the hazard analysis of a four-division digital ESFAS, which was modeled based on the digital ESFAS design for an advanced pressurized water reactor [75].

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

This four-division digital ESFAS includes the portion of the plant protection system (PPS) that activates the engineered safety features and their component control system (CCS). The safety instrumentation and controls of the ESF systems consist of the electrical and mechanical devices and circuitry from sensors to actuation-device input terminals, which are involved in generating signals that actuate the required ESF systems. The ESFAS portion of the PPS includes the following functions: (1) bistable logic; (2) local coincidence logic (LCL); (3) ESFAS initiation; and (4) testing. After receiving ESFAS initiation signals from the PPS, MCR operator console, or RSR shutdown console, ESF-CCS generates ESF actuation signals to ESF component interface modules (CIMs), which transmit signals to the final actuated device. ESF-CIMs also receive actuation signals from the DPS.

In each division, the ESFAS portion of PPS consists of four divisions. Each PPS division is located in an I&C equipment room and contains both an input and an output module, two BPs, two LCL function racks, and other hardware for interfacing with the other PPS divisions, as shown in Figure 11. The redundant BPs could generate ESF actuation signals to the LCL processors in the four redundant divisions if the process values exceed their respective setpoints. Each LCL rack contains two LPs. Initiation signals are provided to them via the ESF-CCS, which consists of four divisions of group-controller (GC) and loop-controller (LC) cabinets. Each GC supports component control and provides ESF actuation signals to the LC. Each LC has component control logic and a multiplexing function. Each ESF-CCS GC performs selective 2-out-of-4 coincidence logic, the output of the selective 2-out-of-4 logic is transmitted to the component control logic in the LC. The logic produces digital output signals to control the component through the CIM, which performs signal prioritization [29].

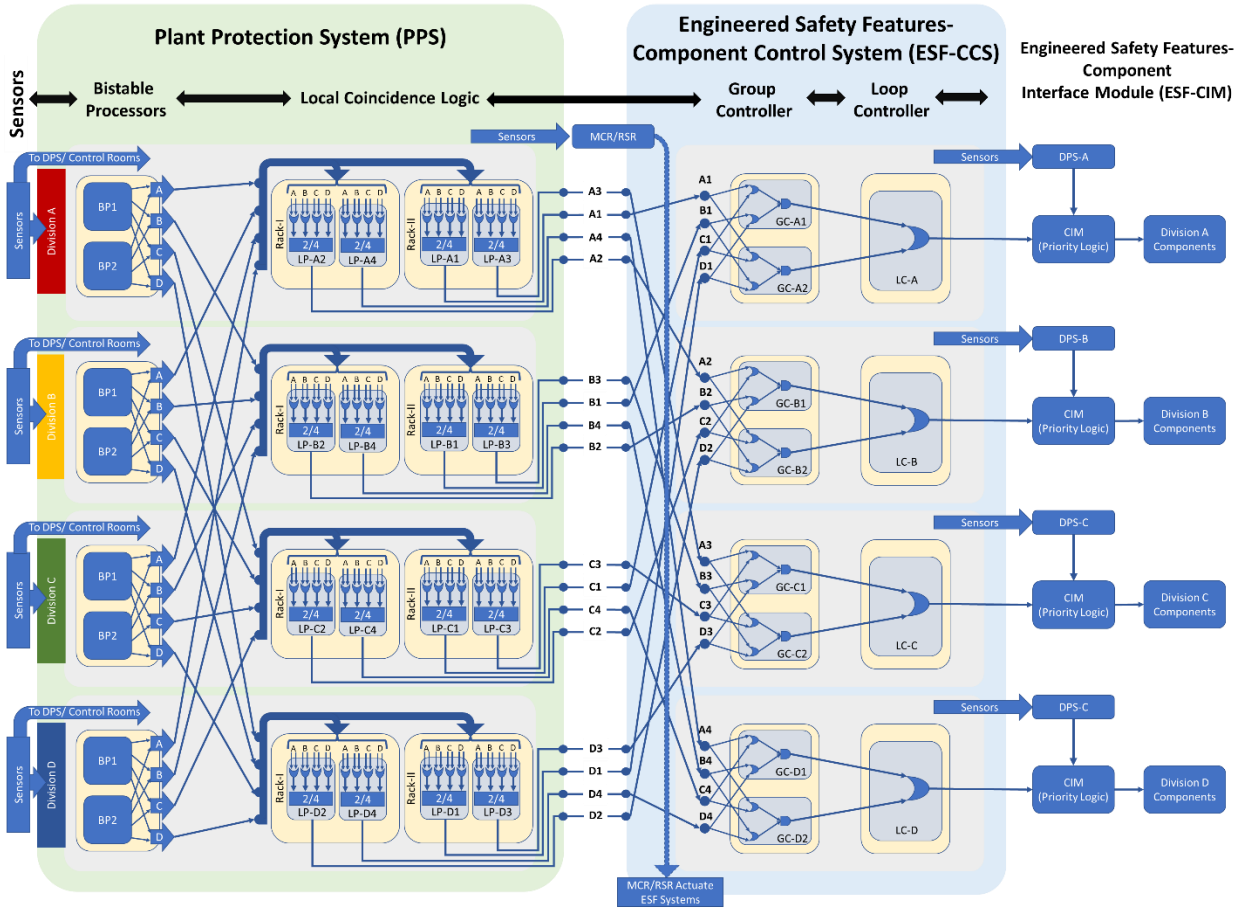


Figure 11. ESFAS functional logic.

Step 2: Develop a FT of hardware failures for a top event of interest of the digital system.

The top event for the FT was set as "ESFAS fails to actuate ESF systems." For different ETs, different relevant top events can be identified for ESFAS; for example, the top event of ESFAS could also be "ESFAS sends spurious signals to actuate ESF systems" when the actuation command is not actually needed. For hardware failures of ESFAS components, units, and modules, a hardware-based FT can be built. In this work, the PRA tool SAPHIRE [76] is used to construct the FT. Part of the hardware-based FT is shown in Figure 12. The top event for this portion of the FT is "LP-A1 fails to send actuation signals to GC-A1," where two conditions should be considered if software failures are not included: either an LP-A1 hardware failure or LP-A1 does not receive any signals from BPs.

For LP-A1 hardware failure, four basic events are included: (a) LP-A1 hardware random failure; (b) a hardware CCF of all LPs in Rack II of Division A; (c) hardware CCF of all LPs in Division A; and (d) hardware CCF of all LPs in all divisions. It is assumed that all basic units or modules that have identical function are identical. Both hardware and software diversity are ignored to simplify the process for CCF identification. It should be noted that diversity of the target digital safety system should be considered for a plant-specific hazard analysis. Therefore, three different CCFs are identified here according to the different levels of redundancy: division, unit, and module. In the following steps, the identification of software CCFs are also guided by the category of redundancy levels.

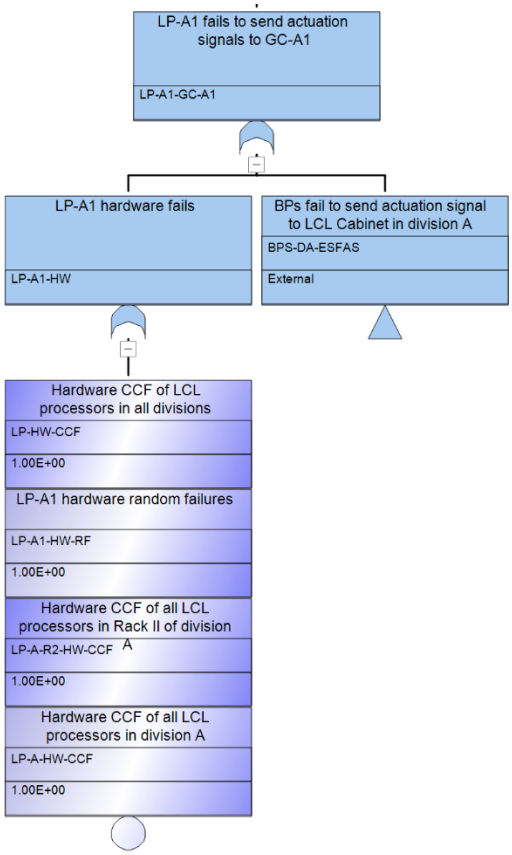


Figure 12. Portion of ESFAS FT showing hardware-type failures only (LP-A1 fails to send actuation signals to GC-A1). The values of failure probabilities are not assigned in this work.

Step 3: Determine UCAs based on a redundancy-guided application of STPA.

The first task is to build tables for losses that will be prevented and hazards that may lead to those losses. The major losses could be identified as human injury or loss of life, environmental contamination, equipment damage, and damage to public perception, while hazards could be core damage, release of radioactive materials, etc. Next, a redundancy-guided multilayer control structure is created for ESFAS, based on its functional logic and hardware structure, as shown in Figure 13. Figure 13 illustrates the different levels of redundancy in a digital ESFAS, shown previously in Figure 11. The top-level layer of redundancy is the four independent divisions to actuate ESF components (i.e., the division-level redundancy). The functioning of each ESF component is affected by a specific division. Signals from plant sensors are sent to all divisions to compare with the engineered set points. In each division, signals are received and sent by several independent LCL racks, where decisions are made as to whether to actuate ESF components. This is the second layer of redundancy: unit-level redundancy. Then, in each LCL rack, actuation signals are transmitted in redundant LPs, which is considered to be the third level of redundancy: the module level redundancy.

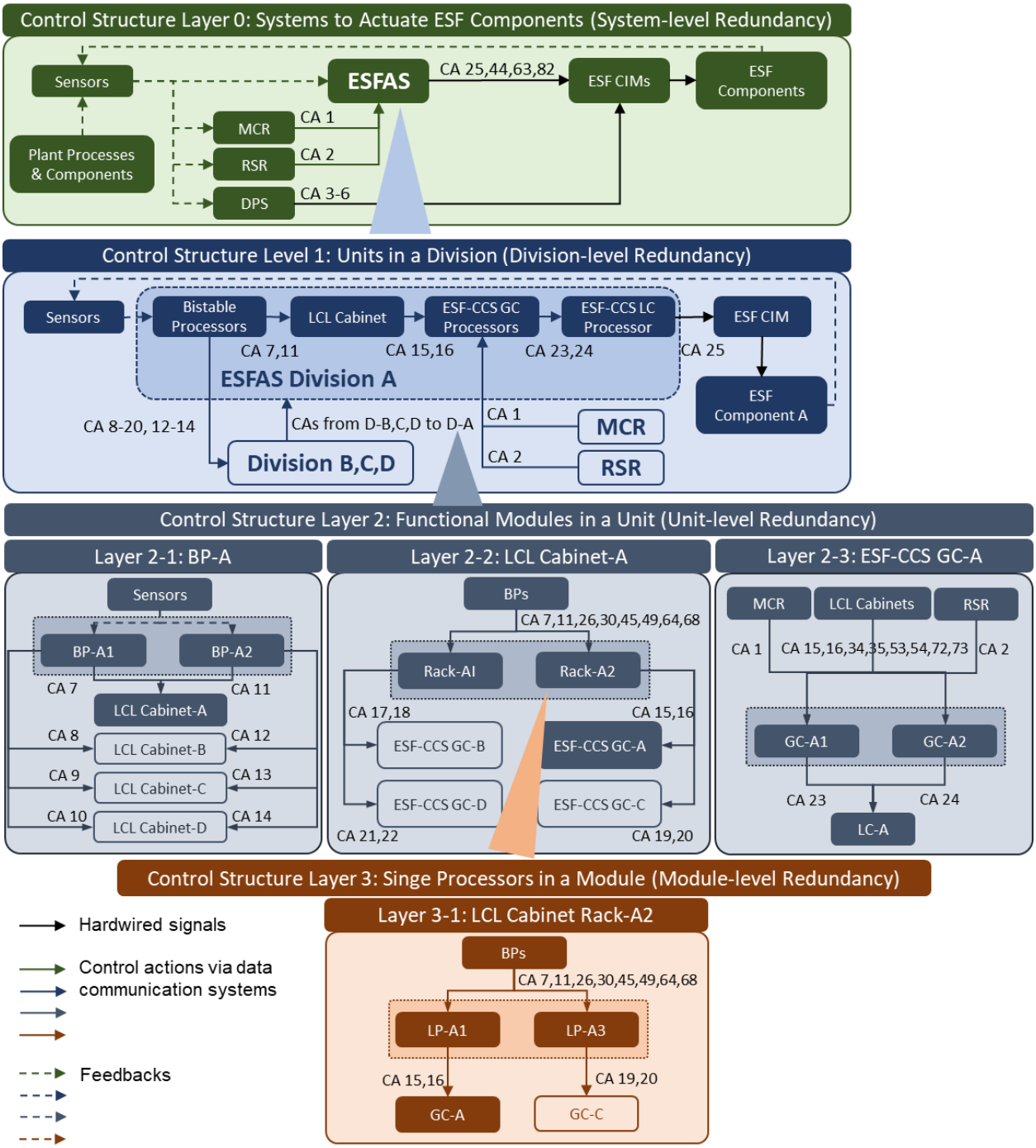


Figure 13. Redundancy-guided multilayer control structure for a digital ESFAS.

Based on the information contained in the multilayer control structure for ESFAS, there are 82 total CAs identified: one CA from MCR, one CA from RSR, four CAs from DPS, and four groups of 19 CAs, one from each ESFAS division—A, B, C, and D. Based on these CAs and the categories of UCAs described in the STPA Handbook, different UCAs can be defined. To deal with the complexity problem of redundancy and identify software CCFs effectively, the system-theoretic hazard analysis is proposed to integrate and reframe STPA process in a redundancy-guided way as a seven-step process, the key outcomes of which are an integrated FT, including software failures and hardware failures, identified

CCFs, and the minimal cut sets to discover the SPOFs leading to the loss of function of the entire digital system. SPOF refers to a situation in which a single part of a system fails, and the entire system loses function as a result. The proposed RESHA approach is illustrated in Figure 15. The steps of the RESHA approach are briefly described below. To deal with the complexity problem of redundancy and identify software CCFs effectively, the system-theoretic hazard analysis is proposed to integrate and reframe the STPA process in a redundancy-guided way as a seven-step process, the key outcomes of which are an integrated FT, including software failures and hardware failures, identified CCFs, and the minimal cut sets to discover the SPOFs leading to the loss of function of the entire digital system. SPOF refers to a situation in which a single part of a system fails, and the entire system loses function as a result. The proposed RESHA approach is illustrated in Figure 15. The steps of the RESHA approach are briefly described below. Table 6 lists the UCAs identified for LP-A1 software failures.

Table 6. UCAs identified for LP-A1 software failures.

Control Action (CA)	**UCA-a: CA is needed, but not given**	**UCA-b: CA is given, but not needed**	**UCA-c: CA is given in a wrong time or in the wrong order**	**UCA-d: CA is given too long or stopped too soon**
CA-15: LCL-Rack-A2-LP-A1 provides an actuation signal to GC-A1	UCA-15-a: LCL-Rack-A2-LP-A1 fails to provide an actuation signal to GC-A1 when it is needed	UCA-15-b: LCL-Rack-A2-LP-A1 provides an actuation signal to GC-A1, but it is not needed	UCA-15-c: LCL-Rack-A2-LP-A1 provides an actuation signal to GC-A1, but too late	UCA-15-d: LCL-Rack-A2-LP-A1 provides an actuation signal to GC-A1, but stops too soon

Step 4: Construct an integrated FT by adding applicable UCAs as basic events.

In this step, applicable UCAs are selected and added into the hardware FT as software failures. For a specific top event in the FT, some UCAs may be inapplicable. Considering the top event for the portion of FT in Figure 12 is "LP-A1 fails to send actuation signals to GC-A1," UCA-15-b and UCA-15-d are not applicable because sending an actuation command is required and is not a continuous action. Only UCA-a and UCA-c were considered in this case.

Step 5: Identify software CCFs from duplicate UCAs for redundant designs within the integrated FT.

After integrating UCAs into the hardware FT, the same types of UCAs located in the same redundancy level can be separated into independent failures and CCFs. According to the assumption that all basic units or modules that have identical function are identical, and software diversity is ignored to simplify the process for CCF identification, three different software CCFs, based on UCA-15-a or UCA-15-c, are classified depending on the redundancy levels. UCA-15-a provides an example: (1) all LPs in Rack II of Division A fail to provide an actuation command when it is needed; (2) all LPs in Division A fail to provide an actuation command when it is needed; and (3) all LPs in all divisions fail to provide an actuation command when it is needed, as shown in Figure 14.

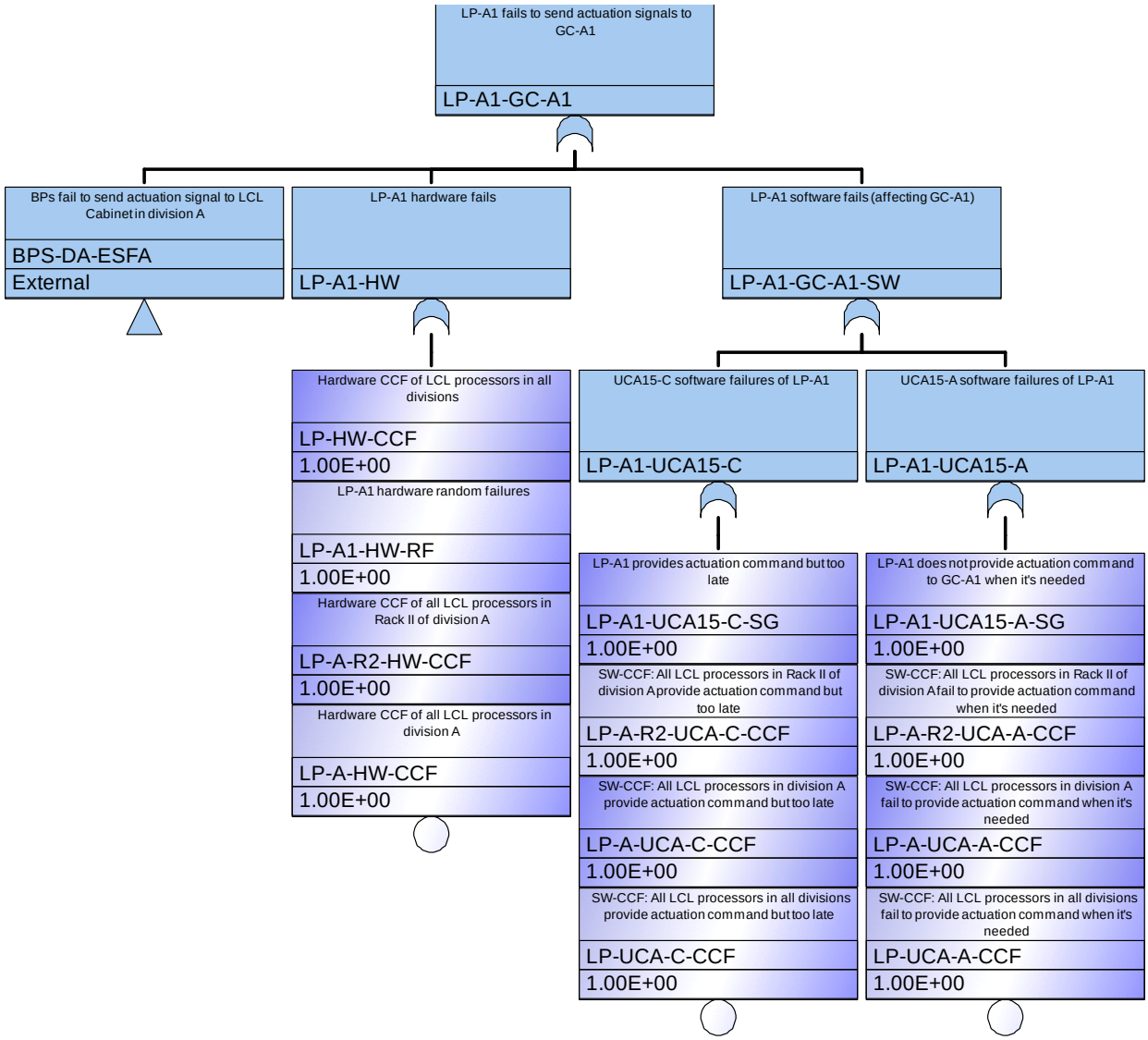


Figure 14. Integrated FT for "LP-A1 fails to send actuation signals to GC-A1," with relevant software failures added.

Step 6: Determine the minimal cut sets to discover the potential SPOFs.

SAPHIRE was used to calculate the cut sets of the integrated FT and to determine the potential SPOFs that might be added by considering the software failures. The cut sets are truncated based on order, rather than by probability, as listed in Table 7. The values of failure probabilities are not assigned in this work. For the fully integrated ESFAS FT model, there is only one first-order cut set that leads to the top event, which is "CIM hardware CCF." CIMs only receive hardwired signals from ESF-CCS and transmit signals to the final actuated devices. This basic event is also the only one first-order cut set for the FT model with hardware failures only and for the FT model without MCR/RSR operations. The latter is considered as a model for automatic actuation only. For the ESFAS FT model without diverse actuation systems (i.e., DPS and MCR/RSR), there are 13 first-order cut sets identified, as shown in Table 8. Four of these basic events are hardware CCFs, while others are software CCFs identified using redundancy-guided STPA. It should be noted that both hardware and software diversity are ignored to simplify the process for CCF identification in this work. Results should be different for plant-specific analysis once

diverse designs are considered. Compared to other cut sets, these identified ones could be potential key hazards that fail the whole digital ESFAS system if other diverse actuation systems are not in good working condition.

Table 7. Cut set calculations for different ESFAS models.

Truncation (Order)	Cut Set #			
	Full FT	FT with hardware only	FT w/o MCR or RSR (Automatic control)	FT w/o DPS, MCR or RSR
5	50714	570	127236	1096601
4	182	6	417	39834
3	19	3	91	139
2	19	3	37	31
1	1	1	1	13

Table 8. First-order cut set for the ESFAS FT model without diverse actuation systems (i.e., DPS and MCR/RSR).

#	Cut set / Basic event	Description
1	LC-BP-UCA-A-CCF	All BPs in logic cabinets fail to send actuation signals to LPs
2	LC-BP-UCA-C-CCF	All BPs in logic cabinets send actuation signals to LPs, but too late
3	LC-BP-HW-CCF	BP hardware fails in all divisions
4	LP-UCA-A-CCF	All LPs in logic cabinets fail to send actuation signals to ESF-CCS
5	LP-UCA-C-CCF	All LPs in logic cabinets send actuation signals to ESF-CCS, but too late
6	LP-HW-CCF	LP hardware fails in all divisions
7	ESF-CCS-GC-UCA-A-CCF	All GC processors in ESF-CCS fail to send actuation signals to ESF-CCS LC processors
8	ESF-CCS-GC-UCA-C-CCF	All GC processors in ESF-CCS send actuation signals to ESF-CCS LC processors, but too late
9	ESF-CCS-GC-HW-CCF	GC processor hardware fails in all ESF-CCS divisions
10	ESF-CCS-LC-UCA-A-CCF	All LC processors in ESF-CCS fail to send actuation signals to CIMs
11	ESF-CCS-LC-UCA-C-CCF	All LC processors in ESF-CCS send actuation signals to CIMs, but too late
12	ESF-CCS-LC-HW-CCF	LC processors hardware fails in all ESF-CCS divisions
13	CIM-HW-CCF	CIM hardware fails in all divisions

Step 7: Identify and provide guidance to eliminate latent faults or triggers of CCFs.

This step focuses on providing guidance to eliminate potential triggering conditions or events that activate dormant faults to the CCFs that were identified in previous steps. As mentioned in Section 4, for software CCFs or independent failures, all causal factors can be identified in two categories: (1) unsafe controller behaviors; and (2) inadequate feedback or outside information. The triggers of software failures are defined as the contexts of the identified UCAs. The step takes the CCF of UCA-15-a (#4. LP-UCA-A-CCF in Table 8) as an example to illustrate how to determine these causal factors. The identification of causal factors should cooperate with the expert teams in system and software engineering, HRA, etc. According to the contexts of the UCAs, different sub-causal factors can be defined for the two categories by using Bayesian networks. Figure 15 displays a simple Bayesian network; more details should be added via collaborations with different expert teams. In this way, reliability analysis can be performed based on these Bayesian networks and reliability models for quantifying the probabilities of identified CCFs in future work.

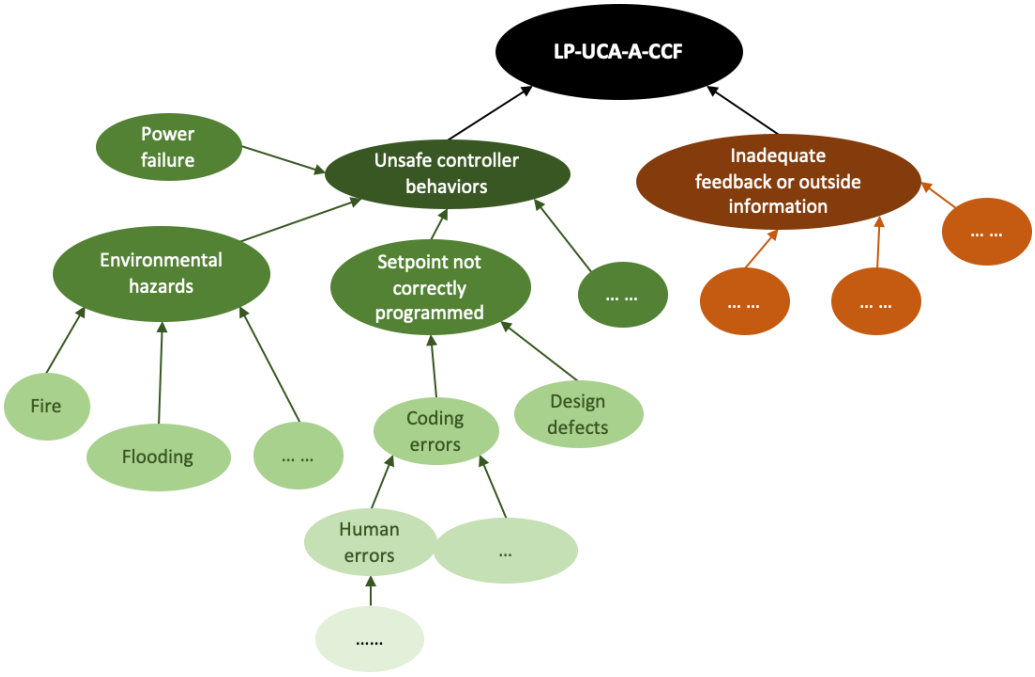


Figure 15. A simple Bayesian network for the identification of causal factors of a CCF.

4. INTEGRATED RELIABILITY ANALYSIS

This Section aims to propose a generic method for quantifying the probabilities of software failure that are defined as UCAs in Section 3. When applying RADIC for the specific plants with DI&C systems, relevant hardware and software information can be obtained for the plant-specific reliability analysis. This Section also demonstrates the proposed reliability analysis method based on a digital RTS design as described in Section 3.

4.1 Desirable Attributes for Quantitative Software Reliability Methods

In this Section, desirable attributes are discussed, as well as some common limitations found in the quantitative software reliability methods (QSRMs) described in the literature. A review of past works has provided some of the desirable QSRM attributes, which, when missing, can be a limitation. A QSRM should:

- have a clear method description
- have reasonable assumptions
- consider operational conditions
- consider the quality of lifecycle activities
- incorporate of testing and experience
- account for uncertainty
- be verified and validated
- be capable of demonstrating systems with high reliability
- consider CCFs
- not be based solely on previous experience
- account for dynamic interactions between: (a) the digital system and controlled processes; and (b) the components of the digital system itself.

Despite there being no method capable of satisfying all desirable attributes [5] [59], the list can be used to select the best methods for specific applications. Case-specific application of methods may help to minimize limitations and increase analysis coverage.

Specifically related to our previous work using RESHA is the need to quantify failures associated with subsections or modules of a software. Contrary to our need, nearly all "QSRMs consider the software system as a whole, and not as separate modules, or broken down by failure modes." Though some may be able to be applied to subsystems, this is not typically indicated. Consequently, finding an application capable of working with the results from RESHA is the primary challenge for our work.

4.2 Approach Description

The BAHAMAS workflow is discussed in this Section, where each of the main methods mentioned in the approach are incorporated for the reliability analysis of a software system. As discussed in Section 1, the risk assessment of digital systems has been divided into three phases. Phase 2 provides quantification for the results found in Phase 1. Although it is the intention in Phase 2 for BAHAMAS to be flexible, much of its formulation is based on the results of a RESHA-based Phase 1 hazard analysis. Consequently, the subsequent approach to Phase 2 is tailored best to hazards identified by the RESHA. BAHAMAS involves six main steps, as shown in Figure 16.

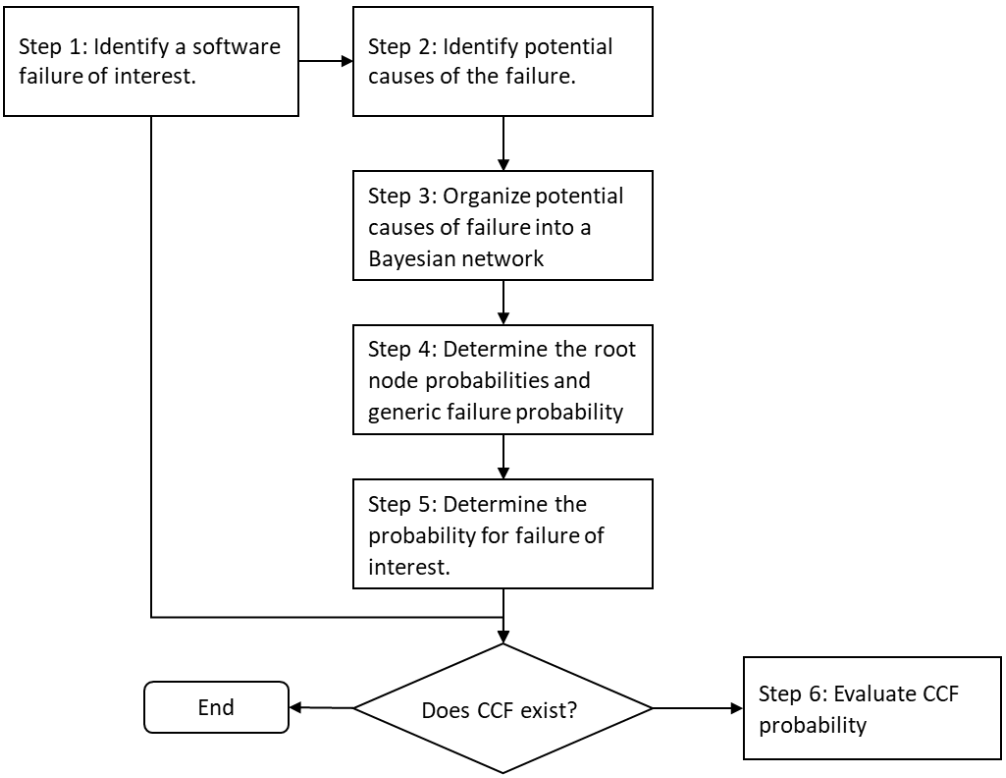


Figure 16. BAHAMAS workflow.

Step 1: Identify a software failure of interest.

The first step of BAHAMAS is to select an event needing to be quantified from a qualitative study. This step assumes that some previous hazard identification work has been completed. The event of interest should include details of the controller, CA, and context. The term controller can be thought of as a digital system or sub-part or module of a digital system (e.g., essentially anything required to perform some action). RESHA output falls into four main categories based on STPA principles: (1) action not provided; (2) action provided and not needed; (3) action provided too early, too late, or in the wrong order; and (4) actions provided for too long or stopped too early. These outputs form the basic sub-categories of failure to be quantified by BAHAMAS. The purpose of Step 1 is to select a basic event of interest and ensure it is adequately detailed for further analysis.

Step 2: Identify potential causes of the failure.

The purpose of Step 2 is to collect and organize information regarding the event of interest. In Step 1, the specific controller, action, and context of the event should have been identified. The analyst should then determine the dependencies, inputs, outputs, and components of the controller. Emphasis should be made to include all the components of the controller that have software or integrated software (firmware). However, it is ultimately the responsibility of the analyst to decide the desired resolution of the analysis. A simplified version of a controller is represented in Figure 17.

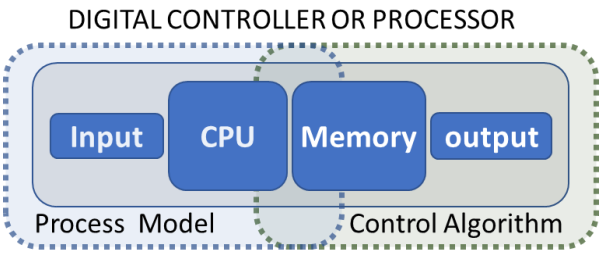


Figure 17. Simplified representation of the components of a digital controller.

It is indicated by STPA that a controller consists of two main parts: the process model and the control algorithm. The process model can be thought of as the diagnosis portion of a system, whereas the control algorithm provides actions based on the model's diagnosis [38]:

- Process model: A controller's internal beliefs used by the control algorithm to determine control actions. The process model may be updated in part by feedback used to observe the controlled process.
- Control algorithm: specifies how control actions are selected based on the controller's process model, previous control inputs and outputs, and other factors. Figure 18 provides an example of the relationship between the process model and control algorithm. Using the concepts of control algorithm and process model can help determine the potential causes that may interrupt or interfere with the correct behavior of the controller. It is a fundamental assumption that failures should have a root cause of human error. Consequently, root causes identified during the analysis of the controller, its components, process model, and control algorithm should be extracted to their human sources of error.

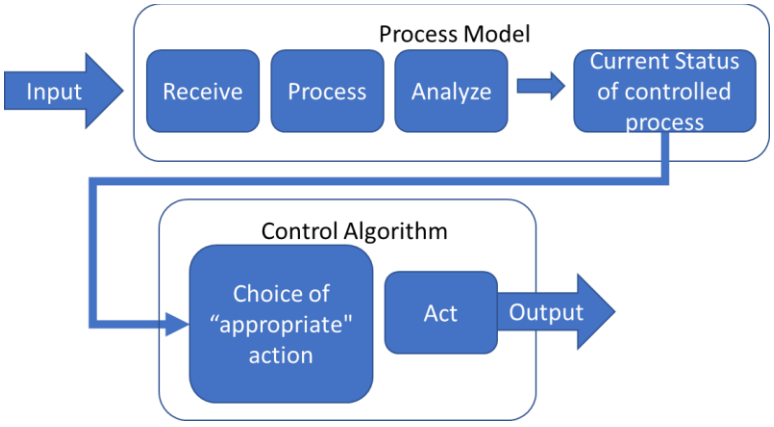


Figure 18. General relationship between a process model and control algorithm for a controller.

Step 3: Organize potential causes of failure into a Bayesian network.

This step is focused on creating a BBN based on the causes identified in Step 2. The main idea of the BBN is to create an acyclic (i.e., without feedback [4]) graphical network representing the relationships of interest. In this case, the relationships between root causes and probability of software failure. The reader can visit [78] or other sources for guidance on formulation or additional information regarding the BBN. For this step, each causal factor and root error source should be arranged as base or root nodes for the network and combined in a manner as follows.

It is assumed that the probability of error in the SDLC process are a direct result of human error. The BBN above is a simplified version. Each application will have variations, but BAHAMAS requires that the penultimate node should be the probability of a fault existing in the system.

Step 4: Determine the root node probabilities and generic failure probability.

This step adapts methods described in the approach for the determination of a necessary parameter for the evaluation of probability of failure given the probability of a fault in the system.

Step 4-A: Assign root node probabilities based on the SDLC quality of the generic software.

The purpose of this step is to provide numerical values for the root nodes of the BBN based on the average or general quality of the software. It is assumed the root source of failure can be sourced to human errors during the development of the software (i.e., the SDLC). As mentioned previously, HRA methods can be applied to quantify human errors. The analysist must select an HRA method based on the needs of the analysis. It is important the analysis be aware of the limitations and assumptions involved in the HRA method chosen. Also, when the SDLC is not clearly defined, the analyst must provide clear assumptions of the necessary activities and verification processes of the SDLC in order to apply HRA. Some applications are flexible, but will require assumptions to extend their application to tasks outside standard human responses to events within a NPP. CREAM attempts to incorporate aspects of the decision-making processes and cognitive concerns of human action into a reliability assessment [63]. SPAR-H provides the fast and convenient approach to analysis of human actions [62]. THERP is one of the first HRA methods. It should be noted that BAHAMAS is flexible and should produce consistent results regardless of selected HRA method. It is simply advisable that a more detailed HRA will provide greater coverage of SDLC attributes, and thus, produce a more comprehensive result.

Once the desired HRA method(s) is selected, the analyst will apply the method for a general SDLC. The HRA should be applied using assumptions to match the general quality of SDLC used for the software being analyzed (i.e., safety or non-safety software, etc.).

Step 4-B: Assign simple conditional probabilities for all except the final node.

To evaluate the probability of software failure, the BBN requires a conditional relationship between parent and child nodes. BAHAMAS requires the BBN to be structured in such a way as to limit additional uncertainties by minimizing the number of conditional relationships that must be provided by the analyst. The goal is to construct the network such that there are binary conditional relationships between each node, except for the final two. This conditional probability table for each node will be similar to the example shown in Table 9. Note that the BBN is structured so that conditional probability is either “0” or “1” for all the nodes, except the final node.

Table 9. Example conditional probability table.

Parent Nodes	**State of Node**		**State of Node**	
Error in SDLC Task 1 for module A	Y		N	
Error in SDLC Task 2 for module A	Y	N	Y	N
Child Node State	**Probability Given Parents**			
Fault in Module A (Yes)	1	1	1	0
Fault in Module A (No)	0	0	0	1

Step 4-C: Evaluate the BBN for the probability of faults for the generic software case.

Using the following equations or some other type of commercial software, the analyst should evaluate the marginal probability of faults for the generic software case. The following is a discussion on the determination of marginal probability within a BBN.

The marginal probability refers to the probability of a specific state within the joint probability of all other states [78]. The marginal probability incorporates both the conditional and node state probabilities for the calculation of a specific node prior to any observations of the system. The term "prior" means without having updated the information for any of the nodes. The concept of Bayesian updating is not necessary for the evaluation of the BBN of our current approach. The reader is advised to visit sources such as [78] to learn more about BBN updating, if desired. For the sake of the current approach, only the marginal probability determination is shown.

Given the following properties:

- Joint probability, also called the intersection [78], of two events A and B is $P(A,B)$:

$$P(A,B) = P(A \cap B) = P(A) \times P(B|A) \tag{6}$$

- The commutative property [79] states that $P(A \cap B) = P(B \cap A)$:

$$P(B \cap A) = P(B) \times P(A|B) \tag{7}$$

The equation for marginal probability is:

$$P(A = a_i) = \sum_{j=1}^{m} P(a_i, b_j) \tag{8}$$

Consider a BBN of two binary state nodes: Node A (parent) and B (child). Each has two possible states: True or False (i.e., $A = \{True, False\}$ and $B = \{True, False\}$). Assume, in this example, the probability for each state of Node A is known, in addition to the conditional probability of Node B given each state of Node A. Using these known values, the marginal probability of B = true can be determined:

$$P(B = b_1) = \sum_{j=1}^{m} P(b_1, a_j) = P(b_1, a_1) + P(b_1, a_2) \tag{9}$$

Because $P(B|A)$ is known, it is beneficial to use the commutative property to arrive at:

$$\begin{aligned} &\sum_{j=1}^{m} P(b_1, a_j) = P(a_1, b_1) + P(a_2, b_1) = P(a_1)P(b_1|a_1) + P(a_2)P(b_1|a_2) \\ &= P(A = true)P(B = true|A = true) + P(A = False)P(B = true|A = False) \end{aligned} \tag{10}$$

The known probability values can then be used to evaluate the solution for B=true. This is the process by which the BBN can be evaluated by hand. Of course, for larger BBNs, it is the more convenient option to use commercial software.

Step 4-D: Determine the generic failure probability for failure of interest.

The SFP value must be determined first for a generic representation of software (e.g., control systems software or safety systems software). The idea is to gain a generic distribution for SFP based on representative software experience. This SFP should match the case of the chosen failure of interest from the categories listed in Step 1 (i.e., failure on demand, spurious action, wrong timing, etc.).

Kang et al. [65] collected operational data from nuclear safety-rated software failure on demand around the world. A hierarchal Bayesian analysis [80] was then used to estimate the SFP. This process involves making an estimation of the distribution of the probability of interest, called a "prior," and then updating the estimation using current observations. Together, the prior and observations (e.g., operational data) are used to estimate a distribution for the probability of interests.

The basic steps of this process are to first guess a prior SFP distribution. This can come from research, historical data, and expert opinions. Second, actual data should be collected for the SFP. Finally, an update of the prior distributions based on the observations should be made. Some key points mentioned in [80]:

- The prior distribution should be based on information other than the data collected (e.g., one should not use the same data for both the prior and the update).
- There are many possible distributions to select. The chosen distribution should match best with the case being analyzed.
- The most convenient prior distribution for probability is the beta distribution.

A convenient option for this form of Bayesian analysis is to use existing software. The NRC and Idaho National Laboratory (INL) developed a free online software called the reliability calculator website [81], which is a useful tool, but provides a limited number of prior distributions to work with.

It is important to note the following: Designers of safety systems often attempt to mitigate failure via redundancy, which limits or transitions system failure modes from single failures to the "hopefully" less likely CCFs. Thus, failures of safety systems may be a direct result of either single or CCF sources. In this Section, operational data is used to provide an estimate probability distribution for software failure. Without knowing the design for each of the sample systems, it is not possible to verify whether CCFs are a relevant concern. Therefore, it is reasonable to assume that either single or CCF causes could lead to failures of the sampled safety systems. Thus, the SFP distribution found here represents both CCF and single failure contributions.

Step 4-E: Determine the value for the "new parameter."

Now that both the generic SFP and Pr ($faults$) have been determined, the new parameter can be calculated:

$$new\ parameter = SFP/Pr\ (faults) \tag{11}$$

The new parameter is specific to the HRA methods and the type of software being analyzed. Any future applications using this new parameter must be done using the same HRA methods and for the same software types at the risk of incorrect results.

Step 5: Determine the probability for failure of interest.

Step 5-A: Determine the value for the "new parameter."

The root nodes probabilities should be modified to match the quality of SDLC for the specific software. The process is identical to Step 4-A. Once the nodes are modified, the analyst can then create a conditional probability table for the final BBN node. The conditional relationship between the final two nodes is defined by the new parameter found previously. The conditional probability table should appear, as shown in Table 10.

Table 10. Conditional probability table for the probability of the failure of interest.

Parent Nodes	**State of Node**	**State of Node**
Fault exists in the system	Y	N
Child Node State	**Probability Given Parents**	
Probability of Failure given parent node (Yes)	New parameter	0
Probability of Failure given parent node (No)	1-(new parameter)	1

Step 5-B: Evaluate the complete BBN for the specific software case.

This step focuses on the BBN evaluation to determine the probability of the failure of interest. The process mirrors the evaluation in Section 4.3 with the evaluation of marginal probability, including the BBN end node. The result of this step stems from the generic SFP found using operational data. The SFP represents the contributions of single and CCF sources. Step 6 is necessary to determine the proportions of single and CCF source to the total SFP for the specific case.

Step 6: Evaluate CCF probability.

BAHAMAS determines the probability that a failure will occur because of faults existing in the software. Step 6 relies on the beta factor method to determine the single and CCF contributions to the total SFP found in Step 5. This is done using the following equations for the failure of A:

$$P(A_{total}) = P\left(A_{single}\right) + P(A_{CCF}) \tag{12}$$

$$P\left(A_{single}\right) = (1-\beta)(A_{total}) \tag{13}$$

$$P(A_{CCF}) = (\beta)(A_{total}) \tag{14}$$

Total failure probability of the component or module consists of the individual and the CCF probabilities. Individual failure and CCF failure are mutually exclusive events; therefore, the total probability represents the possibility of either event, which is found by the sum of the two probabilities.

4.3 Case Study of Probability Estimation of Software Failures

This Section describes the reliability analysis of the four-division digital RTS, shown previously in Figure 5, which has a similar structure to state-of-the-art digital systems in existing NPPs [75]. The analysis follows the six-step process outlined in Section 4.2. The reliability analysis builds on the hazard analysis performed by RESHA; therefore, the reliability analysis adopts the initial hazard analysis assumptions in addition to the new assumptions made for this reliability case study.

Below are some assumptions of BAHAMAS, in addition to the assumptions for RESHA:

- The hardware failure causal factors that could lead to a software failure have been left out of the BBN because given hardware failure of software components (e.g., memory, DOM, DIM, etc.), it is assumed a hardware failure will ultimately cause failure of the specific component to perform its required function. And if grouped with the HD failures in the FT.
- The multiplicities of CCFs have been limited to all m/m identical components failing. This is the all or nothing approach that is common for analysis, which use the beta factor method for quantifying CCFs. This eliminates consideration of the single division or sub-module CCF and is done to simplify the case study.
- Generic SFP distributions are gathered from operational data, unless specified otherwise, are assumed to consist of both individual and CCF sources.
- Generic and specific models are assumed to have the same SDLC because a specific software is unavailable to analyze.
- Generic controller or processor is assumed to have four parts: the input, memory, processor, and output. Each having potential for firmware failures.
- Setpoint faults will not result in the probability of failure on demand for the RTS, as it is assumed that setpoint faults will more likely result in delayed or spurious trip activation.
- All root causes are modeled with the same HRA values as the application software. This is to simplify the demonstration of the case study.

- It is assumed that any HRA method can be used for the quantification of human root causes, though some may have better coverage. THERP was used for this case study. All assumptions related to the application of the THERP are included in Appendix A.

Step 1: Identify a software failure of interest.

Step 1 requires an event to be chosen from a hazard analysis. The previous work relied on RESHA to identify hazards associated a four-division digital RTS. One of the significant outputs from RESHA was the identification of potential SPOFs. CCFs were included in the category of SPOFs for the RESHA case study, so long as they were represented by a single basic event within the FT. The results of the hazard analysis are shown in Table 5. The reliability analysis case study will provide an example of the quantification for number 12 from Table 5. The basic event for number 12 can be seen in Figure 19. According to the reliability analysis methodology, the event of interest should include details for the controller, action, and context:

- Controller: RTS BPs.
- Action: Failure to provide trip signal to Logic Cabinets.
- Context: During an AOO.
- Additional Detail: The BP is responsible for evaluating the status of the NPP and initiating a trip during unsafe plant conditions. The trip signal is sent to the logic cabinets for each of the four RTS system divisions.

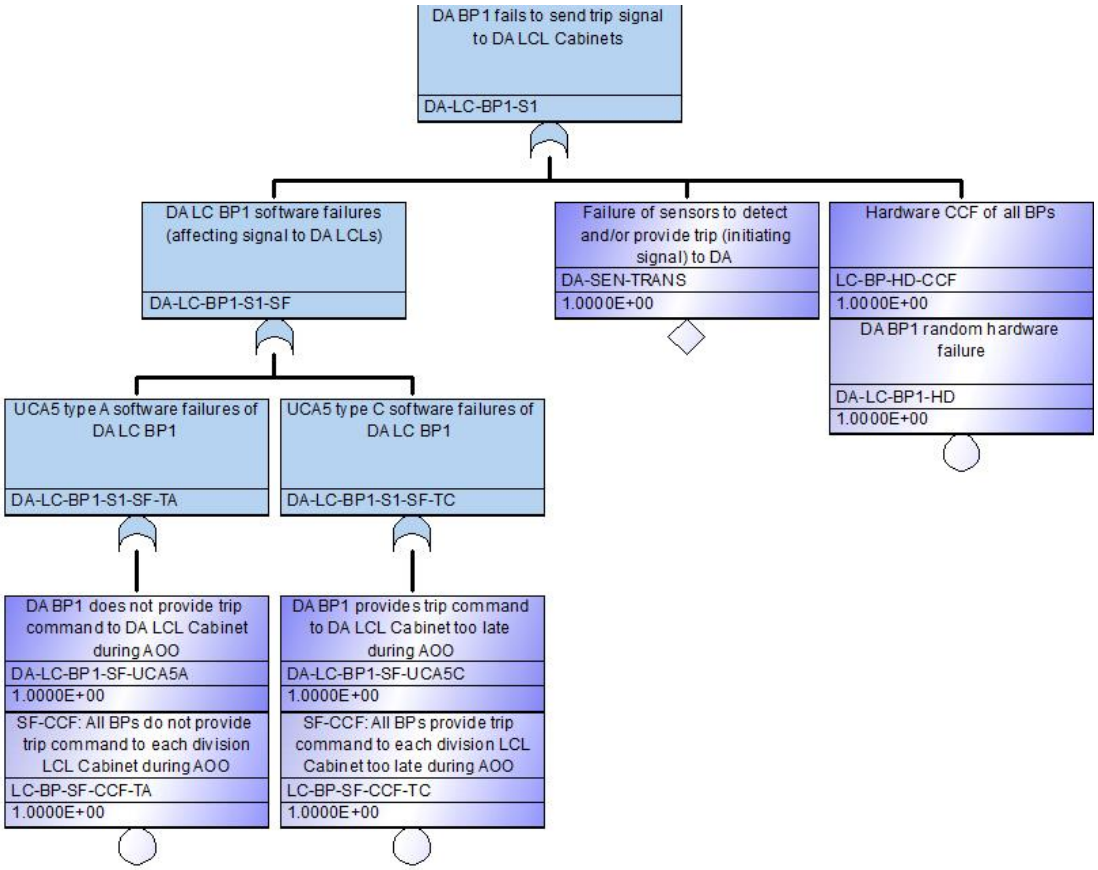


Figure 19. Failure event of interest (LC-BP-SF-CCF-TA), as shown within a FT.

Step 2: Identify potential causes of the failure.

The purpose of this step is to organize and collect information regarding the event of interest. Details for the event of interest may include dependencies, inputs, outputs, components of the controller, etc. First, the components of the controller are identified. The controller components should be assigned to their roles relating to the process model or control algorithm. The BP is assumed to be a controller consisting of the four components of a generic controller shown in Figure 17. Next, the inputs and outputs and dependencies are indicated.

The process model depends on the input module, central processing unit (CPU), and memory. The inputs to the process model are the sensor signals for reactor trip calculation. Outputs from the process model are nine state variables used to determine if a reactor trip is required:

- Over-power (reactor trips on high value)
- Local power density (reactor trips on high value)
- Logarithmic power (reactor trips on high value)
- Departure from nucleate boiling ratio (reactor trips on low value)
- Reactor coolant system flow rate (reactor trips on low value)
- Pressurizer pressure (reactor trips on high and low values)
- Steam generator water level (reactor trips on high and low values)
- Steam generator pressure (reactor trips on low value
- Containment pressure (trips on high).

The control algorithm relies on the output module, memory, and CPU for performance of its functions. The input to the control algorithm depends on the output from the process model. The output of the control algorithm is to send a trip signal or do nothing based on the 11 trip setpoints for each of the nine plant state variables. The BP will send a trip signal to each division of the RTS based on the state-of-the-plant.

The majority of the information regarding the process model and control algorithm will be used to aid in the analyst assess the quality of the SDLC. When available, actual software code can be used to determine how the components of the BP communicate and how they are used. Understanding individual components helps the analyst better assess the quality of software development processes; more detail leads to a better quantification in later steps.

Root causes of software failure are due to faults within the process model and control algorithm components. Faults can consist of malfunctions in hardware, or faults in the software (including firmware). As mentioned in the assumptions, hardware malfunctions have been grouped under the basic events in the FT for hardware. Therefore, environmental, electromagnetic, vibrational, and other potential causes whose influence on software comes via the hardware damage are excluded from this case study. Individual sensor failures (input to the process model) are also already included in the FT and are not needed for this case study.

The remaining root cause for faults within the software is due to human actions. These include human actions during the SDLC and during maintenance work, such as when a human worker installs setpoints for the RTS. For simplicity, the case study divides the SDLC into two categories: (1) the development of design requirements; and (2) the fulfillment of design requirements (i.e., software development). It is understood that faults are a result of human errors and a cause of software failure. Faults in the development of the application software used within the BP may result in failure of the BP to provide a trip signal. Faults may also lead to spurious or delayed trip signals. The question should be asked, “Can wrong setpoints in the software result in the failure of interest?” The simple answer is, “Possibly.” However, based on the structure and diversity in trip sensors and trip calculations, it might not be necessary to include setpoints in the BBN for the failure of interest.

Setpoints are used by the control algorithm of the BP to determine whether the reactor should trip. The plant state variables are all highly coupled by the physics of the reactor. A trip signal in one will manifest in the others as well, with subtle timing differences dependent on physics and type of AOO. For example: there is no doubt that an incorrect setpoint may result in the protection system failing to trip for low coolant flow. However, because of the highly coupled nature, the departure from nuclear boiling ratio (DNBR) trip would also activate for a low reactor control system (RCS) flow scenario. A complete failure to trip, whose cause is improper setpoints, would only occur should all of the trip setpoints be improperly set and in the "perfectly wrong" arrangement (e.g., some setpoints would have to be too high while the others, such as DNBR, would need to be too low). Ultimately, to get a failure to operate, all the setpoints would have to be skewed to such an extreme that it is unlikely such a scenario could ever happen.

It is important to address the other types of failures (e.g., timing-based failures, spurious actions, etc.). Delayed or spurious trip signal activation is significantly more likely to be caused by wrong setpoints because the event would not require a "perfect" combination of human errors. However, this creates a new challenge for modeling. It is highly impractical to consider all combinations for setpoint variability. Each of the 11 trip setpoints could be correct or incorrect, and for the incorrect cases, the setpoints might either be higher or lower than required. This provides $3^{11} = 177{,}147$ possible ways that the 11 setpoints could be arranged. Explicitly modeling each in a BBN is impractical for the current approach. An alternative solution would be to simplify the options for failure of all the setpoints together (either they are all set too high, too low, or are all correct).

In summary for this Section, the only contributors to software failure of interest modeled for the BP are those associated with the SDLC. Setpoint concerns are assumed negligible. Additionally, sensor input failures to the RPS along with hardware failures are also left out of the BBN because they were included with the original FT.

Step 3: Organize potential causes of failure into a Bayesian network.

This step requires that the potential causes of the failure of interest be placed into a BBN following the format shown in Figure 20. As discussed in Step 2, only SDLC causal factors used BBN for the UCA-A type failure of the BP.

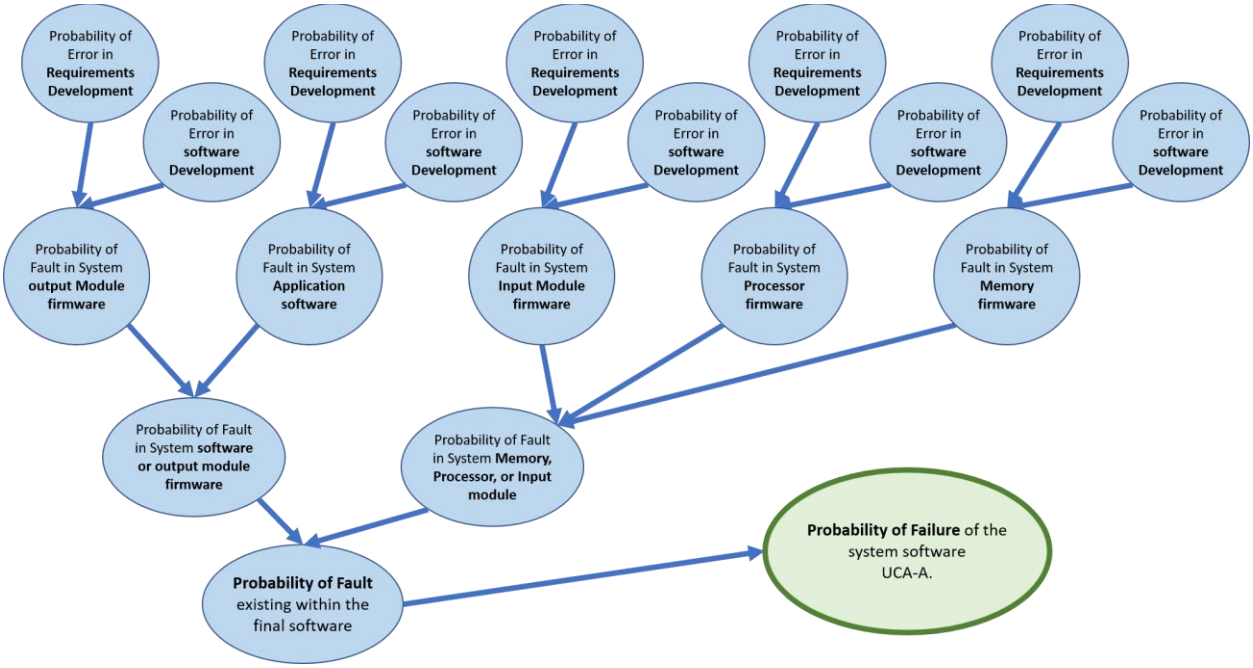


Figure 20. BBN for the UCA-A of BPs of the RTS.

Step 4: Determine the root node probabilities and generic failure probability.

Step 4-A: Assign root node probabilities based on the SDLC quality of the generic software.

The purpose of this step is to supply the probabilities to the root nodes of the BBN for a representative generic software. HRA will be used to assess the quality of generic software development. THERP was selected for this analysis because it is one of the most widely known methods having been around for 50 years [82]. The following is a brief discussion of the application of THERP for the determination of requirements.

THERP consists of several steps for the quantification of human error probability. The steps applicable for this case study are to [61]:

1. Define the system failures of interest:
    - The system failure of interest for this example is human error in the development of application software requirements.
2. List and analyze the related human operations (e.g., perform a task analysis):
    - The applicable points from the THERP description of the task analysis include: (1) identifying the man and machine interfaces and associating their influence on human performance; and (2) identifying problem areas in the design likely to cause human error (e.g., written procedures policies, practices, people skills).
3. Estimate relevant error probabilities:
    - The process involves using the HRA ET to determine the probability of human error. The ET limbs represent binary actions for "correct" or "incorrect," the probability of which are conditional [61]. The dependence between limbs of the ET are assigned conditional probabilities based on the relationships identified in the task analysis.

A summary of the application of THERP Steps 1–3:

A brief review of two international standards available from IEEE has provided a list of some of the key elements associated with the development of software requirements [83] [84] [85]. Key elements were selected from these sources and combined into a simplified task list for the development of requirements for the software of the BP:

- Define the scope
- Define the purpose
- Define the parameters of operation
- Define interfaces
- Identify applicable codes and standards
- Define the quality assurance and testing plan
- Create technical specifications
- Communicate design specifications.

Work assignments were assumed for each task. Additionally, dependencies were assigned between the tasks to account for the influence of one on another. Using Chapter 20 from THERP (NUREG 12) a nominal human error probability was assigned to each task. The quality of each task was then assigned a medium score value based on the principle assumptions discussed in Appendix A. Finally, these tasks were formed in an HRA ET for the evaluation of the probability of human error for the development of BP software requirements. Details of the HRA analysis using THERP can be found in Appendix A. The results of the HRA analysis for medium SDLC quality is shown in Table 11.

Table 11. THERP HRA for medium quality SDLC.

Probability of Human Error in:	Medium-level Quality SDLC
Requirements of Software	3.04E-4
Development of Software	2.28E-3

Step 4-B: Assign simple conditional probabilities for all, except the root and final nodes.

The conditional probabilities are assigned binary values to decrease the number of uncertainties in the model. The following shows the conditional probability tables used for analysis. Note that root nodes do not have conditional probability tables; hence, they are not included in Figure 21.

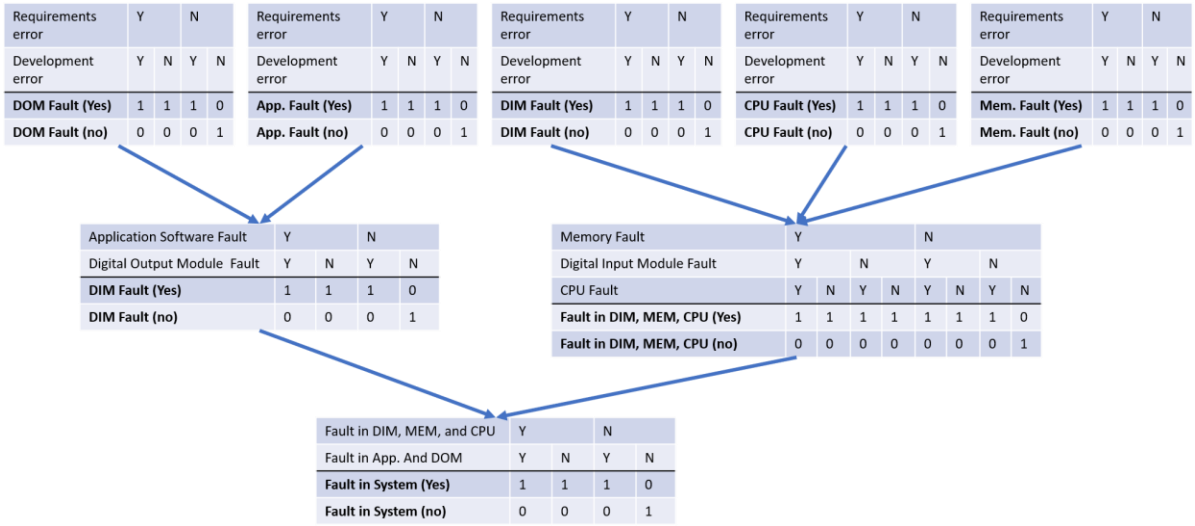


Figure 21. Conditional probability tables associate with the nodes of the BBN in Figure 20.

Step 4-C: Evaluate the BBN for the probability of faults for the generic software case.

Calculation of the BBN consists of evaluating the marginal probability of each to the nodes of the BBN up to and including the penultimate node. The evaluation was done by hand using the equations described in Section 4. The resulting probability of fault within the BP is 1.03E-2.

Step 4-D: Determine the generic failure probability for failure of interest.

The generic SFP for UCA-A of BP is selected based on the data gathered in [50]. The probability of failure on demand for generic safety system software based on operational experience is zero failures in 4457 demands [50].

To apply the process prescribed in Section 4.2, a prior distribution needs to be assumed. As mentioned in [4] and [86], the highest safety integrity level assigned by IEC standard 61508 is SIL 4, which has a corresponding probability of failure on demand at 1E-4 or less—the highest safety integrity assuming most safety systems should have a probability on the order of 1E-4 indicates that a mode for the distribution of the prior should be 1E-4.

Assumptions for the prior are:

- The probability density will concentrate at values peak (mode= 1E-4).
- The probability density near 1 should be very small.
- Values of the distribution should fit between 0 and 1 to correspond with probability.

Based on these assumptions, a lognormal distribution was chosen. By fixing the lognormal distribution node to 1E-4 and varying the value of the σ-parameter of the lognormal distribution, a distribution was chosen with σ-parameter=1.8 and μ-parameter=-5.97. Using the chosen lognormal prior and the operational experience data, the resulting distribution for SFP is found, as seen in Table 12.

Table 12. Generic SFP distribution.

Distribution	Parameter (μ)	Parameter (σ)	Mean of lognormal	Standard deviation of Lognormal	5th percentile of Lognormal	95th percentile of Lognormal
Prior (lognormal)	-5.97	1.80	1.29E-2	6.39E-2	1.32E-4	2.57E-2
Update (lognormal)	-8.8	0.78	2.04E-4	1.87E-4	2.25E-5	5.73E-4

Step 4-E: Determine the value for the "new parameter."

The value of the "new parameter" is calculated using the mean value of the generic SFP distribution (2.04E-4) using Eq (11):

$$new\ parameter = \frac{2.04 * 10^{-4}}{1.03 * 10^{-2}} = 0.0198 = 1.98 * 10^{-2} \quad (15)$$

Step 5: Determine the probability for the failure of interest.

Step 5-A: Modify the BBN to match the case-specific software probabilities.

The case-specific BBN results in a probability of faults within the BP software of 9.43E-3.

Step 5-B: Modify the BBN to match the case-specific software probabilities.

The "new parameter" is used as the conditional probability between the final two nodes of the BBN. Then, the network can be calculated as before to find the SFP = 1.87E-4. The calculation can also be shown to be equivalent to using Eq. (4), which results in the same value for SFP:

$$SFP = (1.98 * 10^{-2})(9.43 * 10^{-3}) = 1.87 * 10^{-4} \quad (16)$$

Step 6: (If required) Evaluate the CCF probability.

The SFP represents the total failure probability of the BP, which contains the contributions from the single and CCF failure sources. Consequently, to find the single and CCF failure, the beta factor method will be used. In this case, the main assumption is that the contribution of common cause and single failures can be represented using a proportionality constant (i.e., beta), as seen in Table 13.

Table 13. Common beta values from literature.

Application	Beta values
Electrical equipment	β =0.01 (best) 0.30 (worst)
NPP data	β =0.03-.22 with 0.10 (average)
Safety systems	β =.001-.05 (good engineering) to .25 (poor engineering)
Hardware failures	β =0.001-0.10
General applications	β =0.01-.1 (good engineering) 0.25 (poor engineering)

The value of β =0.05 is selected from the higher value for the safety systems without assuming poor engineering. The calculation of individual and CCF failure probability:

$$\begin{gathered} P(A_{total}) = P\big(A_{single}\big) + P(A_{CCF}) \\ P\big(A_{single}\big) = (1-\beta)(A_{total}) = (.95)(1.87 * 10^{-4}) = 1.77 * 10^{-4} \\ P(A_{CCF}) = (\beta)(A_{total}) = (.05)(1.87 * 10^{-4}) = 9.34 * 10^{-6} \end{gathered} \tag{17}$$

The case study for the analysis of UCA-A of the BP provides a single failure rate of 1.77E-4 and a CCF of 9.34E-6. These results fit well with the range of software failure probabilities for what has been cited from IEC 61508, "Safety Integrity Level 4" [4] [86]. It should be noted that the general SFP found by hierarchal Bayesian methods come from software systems operations experience representing overall software performance (akin to black box type testing). Using this particular SFP is likely an over prediction for the SFP of individual sub-components or subsystems of the RPS. Consequently, the values for SFP of the BP are likely conservative.

This current approach can be improved with more specific operational testing. White box data, found from testing or operational experience for the specific sub-components of the system, would result in more correct results for the predicted UCAs.

Based on the hazard analysis from RESHA, the single failure of any BP is unable to result in failure of the system; however, CCF can. The results of the reliability analysis show that the CCF probability is less likely than the single event. This does not absolve RPS from risk of failure involving the BP, combinations of failures with other components, including hardware failures, may drastically influence the perceived risk from single BP failures. Future consequence analysis work will investigate the various combinations of failures presenting the most significant risk to the RPS.

It should be noted that the analysis provides a "new parameter" based on software type and HRA methods used in quantifying fault probability. In order to apply the "new parameter" to other software systems, the software type must match, and so also must the methods used for fault quantification (i.e., HRA methods). For example, the fault quantification methods must match because different methods produce different values for the same scenario. Thus, if the methods do not match between the generic analysis and the specific analysis, then the "new parameter" will skew the results. Despite this challenge, the analysis is structured such that it is flexible, and likely can be used for any HRA method deemed appropriate by the analyst, as long as the aforementioned cautions are followed. Additionally, tabulated values of the "new parameter" could be created to streamline future analyses.

Regarding the calculation for CCFs using the beta factor method, this represents a limitation to our approach for reliability analysis because it does not account for the possibility of a multiplicity of CCFs. It is of interest to be able to account for combinations of failures that are not all or nothing. However, for this first version of BAHAMAS, the use of beta factor has been useful.

5. COLLABORATION

Under the RISA Pathway of the LWRS Program, the development and application of the proposed RADIC process—particularly the approaches for redundancy-guided system-theoretic hazard and reliability analysis—have been implemented with collaboration from the Plant Modernization Pathway of the LWRS Program, digital vendors and utilities, universities, and other ongoing initiatives on data collections, methodology development, plant-specific risk analysis, and cybersecurity. This research is being coordinated via ongoing interactions between these organizations to develop integrated research plans that will maximize the effective allocation of resources and support the development and deployment of DI&C technologies. The collaboration with multiple partnerships will formulate an integrated assembly line to support the development, licensing, and deployment of advanced digital technologies to NPPs from data collection, methodology, and tool development to the applications on specific plants. Specific coordination and collaboration research activities include:

- Coordination with the INL nuclear cybersecurity team on the establishment of a technical basis (e.g., PRA models and methods) for cybersecurity analysis of DI&C safety systems.
- Collaboration with universities (e.g., University of Pittsburgh, North Carolina State University, University of Tennessee–Knoxville) to perform risk analysis on artificial intelligence (AI)-guided DI&C system designs to improve their performance.
- Providing risk assessment capabilities for DiD and diversity applications to vendors for their DI&C systems.
- Engaging with Plant Modernization Pathway and utilities to support their pilot projects.

6. CONCLUSIONS AND FUTURE WORKS

6.1 Conclusions

This work has provided a means to overcome technical challenges faced by the nuclear industry for the implementation of DI&C systems. A modularized approach to conduct RESHA for DI&C systems has been developed and demonstrated based on an advanced digital RTS and ESFAS with multilevel redundancy designs. Systematic methods and risk-informed tools are incorporated to address both hardware and software CCFs, which provide a guidance to eliminate the causal factors of potential SPOFs in the design of digital safety systems in advanced plant designs. RESHA provided a means to identify software-based interactions and potential CCFs in highly redundant, state-of-the-art DI&C systems, by fully incorporating redundancy into the hazard analysis process.

Embracing redundancy in the analysis allowed RESHA to meet its objectives in three ways: (1) defining a step-by-step approach for the hazard analysis of digital systems that can help engineers efficiently make design and risk mitigation decisions by providing them a means to systematically identify the most critical CCFs and hazards of DI&C systems; (2) identifying the critical hazards of a system, thereby allowing utilities to effectively manage the cost of safety-rated DI&C by strategically eliminating unnecessary design features; and (3) providing a technical basis for reliability analysis by identifying crucial failure modes and qualitatively determining their effects on system vulnerability. Ultimately, RESHA helps improve the design of highly redundant DI&C through a detailed qualitative hazard analysis.

The method also provides a technical basis for implementing cybersecurity, reliability, and consequence analysis on unanalyzed sequences and optimizing the use of DiD analysis in a cost-effective way. The application of RESHA requires users having sufficient knowledge about relevant methods (e.g., FTA, STPA) and target systems. The identification of causal factors needs collaborations with relevant expert teams, such as system/software designers and engineers and human reliability analysts.

In addition, this work developed a novel method, BAHAMAS, for reliability analysis of DI&C systems. Software failure probabilities are quantified using an integrated approach that incorporates state-of-the-art BBN, HRA, CCF modeling techniques. BAHAMAS also provides a means for analyzing new software systems where operational data is rarely available, as well as flexibility allowing an analyst to employ appropriate HRA methods or incorporate new or advanced methods to capture the desired details of any SDLC. The case study relied on the use of THERP for the quantification of faults in the SDLC. THERP, while a classic and well used method, is aging. Other HRA methods may prove to be better suited for the evaluation of SDLC. CREAM's applications for cognitive aspects, as well as SPAR-H's rapid quantification abilities, also present attractive investigations for future research.

For CCF modeling, this approach applied the beta factor method. However, this approach is limited because it does not account for the multiplicity of CCF events. But the beta factor method does provide a useful and straightforward means to quantify highlighted concerns found from RESHA. Future work will investigate other options for improving CCF modeling techniques.

Section 1 discusses positive attributes for QSRMs. In this case, BAHAMAS has the potential of meeting many of these attributes. By providing a clear method and allowing for flexibility in the use of HRA, the door has been opened to allow for reasonable assumptions for case-specific analysis. The method accounts for lifecycle activities and provides consideration for CCFs. Despite coming up short for verification and uncertainty, the method has the potential to undergo such actions. In addition, reliance on previous experience is flexible (e.g., may not require significant testing) and validation efforts would help clarify how much previous test experience is really required. Finally, some of the assumptions for the case study precluded the consideration of operational conditions; however, BAHAMAS can certainly incorporate environmental and other fault contributors into the BBN. Additionally, operational considerations—particularly the interactions between the digital system and controlled processes—are

partially accounted by consideration of the process model and control algorithm. The process model and control algorithm account for relationships between various attributes of the control system and are used for the HRA analysis. In conclusion, BAHAMAS provides a flexible and useful tool for the quantification of DI&C system software failures and meets many of the desired attributes of an ideal QSRM.

6.2 Future Works

One area for future research in FY-21 deals with the risk analysis for the Human System Interface (HSI) in DI&C modernization of existing NPPs. The HSI is one of the key advanced design features applied for modern DI&C systems of NPPs. Normally, it is designed based on a compact workstation-based system in the control room. The compact workstation provides a convenient operating environment to facilitate the display of plant status information to the operator so that operability is enhanced by using advanced display, alarm, and procedure systems. The HSI should have sufficient diversity to demonstrate DiD protection against CCF of the safety system. However, the vulnerability of HSI is affected by many factors, human errors, cyber-attacks, software CCFs, etc. Therefore, this work aims to identify, evaluate, and reduce these system vulnerabilities to support the licensing, deployment, and operation of the HSI designs.

Main research activities include: (1) performing systematic hazard analysis to investigate system vulnerabilities (e.g., human errors, cyber-attacks, software CCFs) of HSI designs for DI&C upgrades; (2) conducting reliability studies of HSI systems to evaluate how individual failures affects the availability and reliability of HSI systems; and (3) identifying the accidental events that could be induced by system vulnerabilities, but not analyzed by previous risk analyses, and estimate their consequential impacts on key plant responses. Expected outcomes are: (1) an integrated risk analysis and evaluation for a representative advanced HSI design to support its licensing, deployment, and operation; and (2) a diversity and DiD analysis to provide risk-informed insights and suggestions on reduction and cost-efficient optimization of advanced HIS designs.

Another future area for research in FY-21 is the collaboration with industry partners and the LWRS Program Plant Modernization Pathway to complete reliability studies and perform consequence analysis for state-of-the-art DI&C systems. An integrated reliability study and risk-informed consequence analysis will be performed for a representative digital RTS and ESFAS of existing plants with software CCFs and plant responses considered. This work complements other approaches being developed for deploying DI&C technologies and provides risk-informed insights to facilitate the adoption and licensing of safety-related and non-safety-related DI&Cs.

Main research activities include: (1) coordinating and collaborating with industry partners and the LWRS Program Plant Modernization Pathway to perform hazard analyses and reliability studies for a state-of-the-art digital RPS and ESFAS; and (2) integrating digital RPS and ESFAS reliability models into plant level PRA models to evaluate plant level impacts of digital failures, especially software CCFs. Expected outcomes are reliability studies and consequence analysis for state-of-the-art safety-related DI&C systems (i.e., RTS and ESFAS).

Finally, by integrating hazard analysis, reliability analysis, and consequence analysis together, the risk assessment strategy aims to: (1) help system designers and engineers to systematically address digital-based CCFs and quantitatively analyze their effects on digital system vulnerability and key plant responses; (2) improve existing PRA models for the industry by identifying and evaluating the risk associated with DI&C technologies; and (3) provide risk insights to address the licensing challenges facing DI&C upgrades.

7. REFERENCES

[1] K. Thomas and K. Scarola, "Strategy for Implementation of Safety-Related Digital I&C Systems," Idaho National Laboratory, Idaho Falls, 2018.

[2] National Research Council, Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues, Washington, DC: The National Academies Press, 1997.

[3] H. Hashemian, "Nuclear Power Plant Instrumentation and Control," in *Nuclear Power - Control, Reliability and Human Factors*, P. Tsvetkov, Ed., Intech, 2011, pp. 49-66.

[4] T.-L. Chu, M. Yue, G. Martinez-Guridi and J. Lehner, "Review of Quantitative Software Reliability Methods," Brookhaven National Laboratory, 2010.

[5] T. Aldemir, D. Miller, M. Stovsky, J. Kirschenbaum, P. Bucci, A. Fentiman and L. Mangan, "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, DC, 2006.

[6] J. Kirschenbaum, P. Bucci, M. Stovsky, D. Mandelli, T. Aldemir, M. Yau, S. Guarro, E. Ekici and S. A. Arndt, "A Benchmark System for Comparing Reliability Modeling Approaches for Digital Instrumentation and Control Systems," *Nuclear Technology,* vol. 165, no. 1, pp. 53-95, 2009.

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

[8] U.S. Nuclear Regulatory Commission, "A Defense-In-Depth and Diversity Assessment of the RESAR-414 Integrated Protection System," U.S. Nuclear Regulatory Commission, Washington, DC, 1979.

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

[10] G. G. Preckshot, "Method for Performing Diversity and Defense-in-Depth Analyses of Reactor Protection Systems," NUREG/CR-6303, U.S. Nuclear Regulatory Commission, Washington, DC, 1994.

[11] U.S.NRC, "Title 10 CFR (Code of Federal Regulations) Part 50, Appendix A: General Design Criteria for Nuclear Power Plants," U.S.NRC, Washingyon. D.C., 1995.

[12] U.S.NRC, "Digital Systems Software Requirements Guidelines - Volume 1: Guidelines," U.S. NRC, Washington, D.C., 2001.

[13] U.S.NRC, "Digital Systems Software Requirements Guidelines - Volume 2: Failure Descriptions," U.S. NRC, Washington, D.C., 2001.

[14] U.S.NRC, "Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems," U.S. NRC, Washington, D.C., 2008.

[15] U.S.NRC, "Common-Cause Failure Event Insights Volume 1: Emergency Diesel Generators," U.S NRC, Washington, D.C., 2008.

[16] U.S.NRC, "Common-Cause Failure Event Insights Volume 2: Motor-Operated Valves," U.S. NRC, Washington, D.C., 2008.

[17] U.S.NRC, "Common-Cause Failure Event Insights Volume 3: Pumps," U.S. NRC, Washington. D.C., 2008.

[18] U.S.NRC, "Common-Cause Failure Event Insights Volume 4: Circuit Breakers," U.S. NRC, Washington, D.C., 2008.

[19] U.S.NRC, "Historical Review and Observations of Defense-in-Depth," U.S. NRC, Washington, D.C., 2016.

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

[21] U.S.NRC, "Use of Probabilistic Risk Assessment Methods in Nuclear," U.S.NRC, Washington, D.C., 1995.

[22] S. A. Arndt and A. Kuritzky, "Lessons Learned from the U.S. Nuclear Regulatory Commission's Digital System Risk Research," *Nuclear Technology,* vol. 173, no. 1, pp. 2-7, 2010.

[23] U. NRC, "Plans for Addressing Potential Common Cause Failure in Digital Instrumentation and Controls," U.S. NRC, Washington, D.C., 2018.

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

[25] J. Gustafsson, "Reliability Analysis of Safety-related Digital Instrumentation and Control in a Nuclear Power Plant," Royal Institure of Technology, 2012.

[26] O. Backstrom, J.-E. Holmberg, M. Jockenhovel-Barttfeld, M. Porthin, A. Taurines and T. Tyrvainen, "Software reliability analysis for PSA: failure mode and data analysis," Nordic Nuclear Safety Research (NSK), Roskilde, Denmark, 2015.

[27] U.S.NRC, "Traditional Probabilistic Risk Assessment Methods for Digital Systems," U.S. NRC, Washington, D.C., 2008.

[28] U.S.NRC, "Dynamic Reliability Modeling of Digital Instrumentation and Control Systems for Nuclear Reactor Probabilistic Risk Assessments," U.S. NRC, Washington, D.C., 2006.

[29] U.S.NRC, "A Benchmark Implementation of Two Dynamic Methodologies for the Reliability Modeling of Digital Instrumentation and Control Systems," U.S. NRC, Washington, D.C., 2008.

[30] E. Zio and F. D. Maio, "Processing Dynamic Scenarios from a Reliability Analysis of a Nuclear Power Plant Digital Instrumentation and Control System," *Annals of Nuclear Energy,* vol. 36, no. 9, pp. 1386-1399, 2009.

[31] Z. Ma, h. Yoshikawa and M. Yang, "Module level reliability performance evaluation of digital reactor protection system considering the repair and common cause failure," *Annals of Nuclear Energy,* vol. 110, pp. 805-817, 2017.

[32] J. Guo, M. Yang, B. Zou, Y. Zhang, J. Yang and X. Dai, "Nuclear safety-critical Digital Instrumentation and Control system software: Reliability demonstration," *Annals of Nuclear Energy,* vol. 120, pp. 516-527, 2018.

[33] B. A. Gran and A. Helminen, "A Bayesian Belief Network for Reliability Assessment," in *Proceedings of International Conference on Computer Safety, Reliability, and Security*, Berlin, Germany, Sepetember 2001.

[34] C. A. Pollino and B. T. Hart, "Developing Bayesian network models within a Risk Assessment framework," in *Proceedings of the iEMSs Fourth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2008)*, Barcelona, Catalonia, July 2008.

[35] H. G. Kang, S. H. Lee, S. J. Lee, T.-L. Chu, A. Varuttamaseni, M. Yue, S. Yang, H. S. Eom, J. Cho and M. Li, "Development of a Bayesian belief network model for software reliability quantification of digital protection systems in nuclear power plants," *Annals of Nuclear Energy,* vol. 120, pp. 62-73, 2018.

[36] S. J. Lee, S. H. Lee, T.-L. Chu, A. Varuttamaseni, M. Yue, M. Li, J. Cho and H. G. Kang, "Bayesian Belief Network Model Quantification Using Distribution-Based Node Probability and Experienced Data Updates for Software Reliability Assessment," *IEEE Access,* vol. 6, 2018.

[37] H. Bao, H. Zhang and K. Thomas, "An Integrated Risk Assessment Process for Digital Instrumentation and Control Upgrades of Nuclear Power Plants," Idaho National Laboratory, Idaho Falls, ID, 2019.

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

[39] U.S. Nuclear Regulatory Commission, "Identification and Analysis of Failure Modes in Digital Instrumentation and Controls (DI&C) Safety Systems--Expert Clinic Findings, Part 2," RIL-1002, U.S. Nuclear Regulatory Commission, 2014.

[40] Electric Power Research Institute, "Hazard Analysis Methods for Digital Instrumentation and Control Systems," EPRI, Palo Alto, CA, 2013.

[41] A. J. Clark, A. D. Williams, A. Muna and M. Gibson, "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*, Orlando, Florida, USA, 2018.

[42] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick III and J. Railsback, "Fault Tree Handbook with Aerospace Applications Version 1.1," NASA, 2002.

[43] M. Jockenhovel-Barttfeld, S. Karg, C. Hessler and H. Bruneliere, "Reliability Analysis of Digital I&C Systems within the Verification and Validation Process," in *Probabilistic Safety Assessment and Management*, Los Angeles, CA, Septemmber 2018.

[44] N. Leveson, C. Wilkinson, C. Flemming, J. Thomas and I. Tracy, "A Comparison of STPA and the ARP 4761 Safety Assessment Process," Massachusetts Institute of Technology, Cambridge, MA, 2014.

[45] M. Rejzek and C. Hilbes, "Use of STPA as a diverse analysis method for optimization and design verification of digital instrumentation and control ssytems in nuclear power plants," *Nuclear Engineering and Design*, vol. 331, pp. 125-135, 2018.

[46] A. Mosleh, "PRA: A Perspective on Strengths, Current Limitations, and Possible Improvements," *Nuclear Engineering and Technology*, vol. 46, no. 1, February 2014.

[47] "IEEE Recommended Practice on Software Reliability," New York.

[48] Z. Jelinski and P. Moranda, "Software Reliability Research," in *Statistical Computer Performance Evaluation*, W. Freiberfer, Ed., New York, NY: Academic Press, 1972, pp. 465-484.

[49] J. D. Musa and K. Okumoto, "A logarithmic Poisson Execution Time Model for Software Reliability," in *Proceedings of the Seventh International Conference on Software Engineering*, Orlando, FL, 1984.

[50] T.-L. Chu, A. Varuttamaseni, M. Yue, S. J. Lee, H. G. Kang, J. Cho and S. Yang, "Developing a Bayesian Belief Network Model for Quantifying the Probability of Software Failure of a Protection System," NUREG/CR-7233, U.S. Nuclear Regulatory Commisison, 2018.

[51] LYU1996.

[52] "ISO/IED/IEEE International Standard - Systems and software engineering--Vocabulary," in *ISO/IEC/IEEE 24765:2017(E)*, Institute of Electrical and Electronics Engineers, 2017, pp. 1-541.

[53] "IEC standard 61508".

[54] Y. Shi, M. Li, S. Arndt and C. Smidts, "Metric-based software reliability prediction approach and its application," Springer, 2016.

[55] J. Kim, A. U. A. Shah and H. G. Kang, "Dynamic risk assessment with bayesian network and clustering analysis," *Reliability Engineering and System Safety*, vol. 201, 2020.

[56] N. Siu, "Dynamic PRA for Nuclear Power Plants: Not If But When?".

[57] T. Chu, M. Yue, G. Martinez-Guridi, K. Mernick, J. Lehner and A. Kuritzky, "Modeling a Digital Feedwater Control System Using Traditional Probabilistic Risk Assessment Methods," NUREG/CR-6997, U.S. Nuclear Regulatory Commission, 2009.

[58] K. Groth, C. Wang and A. Mosleh, "Hybrid causal mehtodology and software platform for probabilistic risk assessment and safety monitoring of socio-technical systems," *Reliability Engineering and System Safety,* vol. 95, no. 12, pp. 1276-1285, 2010.

[59] T.-L. Chu, Y. Meng, G. Martinez-Guridi and J. Lehner, "Development of Quantitative Software Reliability Models for Digital Protection Systems of Nuclear Power Plants," NUREG/CR-7044, U.S. Nuclear Regulatory Commission, 2013.

[60] EPRI, "Methods for Assuring Safety and Dependability when Applying Digital Instrumentation and Control Systems," EPRI, Palo Alto, CA, 2016.

[61] A. D. Swain and H. E. Guttmann, "Handbook of Human Reliability Analysis with Emphasis on Nuclear Power Plant Applications Final Report," NUREG/CR-1278. U.S. Nuclear Regulatory Commission, 1983.

[62] D. Gertman, H. Blackman, J. Marble, J. Byers and C. Smith, "The SPAR-H Human Reliability Analysis Method," NUREG/CR-6883, U.S. Nuclear Regulator Commission, Washington, DC, 2005.

[63] E. Hollnagel, Cognitive Reliability and Error Analysis Method (CREAM), Elsevier, 1998.

[64] J. Park, A. M. Arigi and J. Kim, "A Comparison of the quantification aspects of human reliability analysis methods in nuclear power plants," *Annals of Nuclear Energy,* vol. 133, pp. 297-312, 2019.

[65] H. G. Kang, S. H. Lee, S. J. Lee, T.-L. Chu, A. Varuttamaseni, M. Yue, S. Yang, H. S. Eom, J. Cho and M. Li, "Development of a Bayesian belief network model for software reliability quantification of digital protection systems in nuclear power plants," *Annals of Nuclear Energy,* vol. 120, pp. 62-73, 2018.

[66] S. H. Lee, S. J. Lee, S. R. Koo, A. Varuttamaseni, M. Yue, M. Li, J. Cho and H. G. Kang, "Optimization of software development life cycle quality for NPP safety software based on a risk-cost model," *Annals of Nuclear Energy,* vol. 135, 2020.

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

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

[69] G. W. Parry, "Common Cause Failure Analysis: A Critique and Some Suggestions," *Reliability Engineering and System Safety,* vol. 34, no. 3, pp. 309-326, 1991.

[70] H. M. Paula, D. J. Campbell and D. M. Rasmuson, "Qualitative cause-defense matrices: Engineering tools to support the analysis and prevention of common cause failures," *Reliability Engineering & System Safety,* vol. 34, no. 3, pp. 389-415, 1991.

[71] P. Hokstad and M. Rausand, "Common Cause Failure Modeling: Status and Trends," in *The Handbook of Performability Engineering*, K. B. Misra, Ed., London, Springer, 2008, pp. 621-640.

[72] U.S.NRC, "Technical Specification Required Shutdown Due to Core Protection," U.S.NRC, Washington, D.C., October 2005.

[73] U.S.NRC, "Design Defect in Safeguards Bus Sequencer Test Logic Places Both Units," U.S.NRC, Washington, D.C., July 1995.

[74] U.S.NRC, "Standard Review Plan: Guidance for Evaluation of Diversity and Defense-in-Depth in Digital Computer-based Instrumentation and Control Systems Review Responsibilities," U.S.NRC, Washington, D.C., August 2016.

[75] "APR1400 Desing Control Document Tier 2. Chapter 7: Instrumentation and Controls," Korea Electric Power Corporation; , Korea Hydro & Nuclear Power Co., Ltd;, South Korea, 2018.

[76] U.S. Nuclear Regulatory Commission, "Systems Analysis Programs for Hands-on Integrated Reliability Evaluations (SAPHIRE) Version 8.0," U.S. Nuclear Regulatory Commission, Washington, D.C., 2011.

[77] K. Korsah, M. D. Muhlheim, D. E. Holcomb, D. Taneja and K. Korsah, "Industry Survey of Digital I&C Failures," Oak Ridge National Laboratory, Oak Ridge, 2007.

[78] N. Fenton and M. Neil, Risk Assessment and Decision Analysis with Bayesian Networks, 2nd ed., CRC Press LLC, 2018.

[79] M. Holicky, Introduciton to Probability and Statistics for Engineers, Springer, 2013.

[80] C. L. Atwood, J. L. LaChance, H. F. Martz, D. J. Anderson, M. Englehardt, D. Whitehead and T. Wheeler, "Handbook of Parameter Estimation for Probabilistic Risk Assessment," NUREG/CR-6823, U.S. Nuclear Regulatory Commission, Washington, DC, 2003.

[81] Idaho National Laboratory, "Reliability Calculator Web Site," U.S. Nuclear Regulatory Commission, [Online]. Available: https://nrcoe.inl.gov/radscalc/Default.aspx. [Accessed August 2020].

[82] R. Boring, "Fifty years of THERP and Human Reliability Analysis," in *PSAM11*, 2012.

[83] "IEEE Guide for Developing System Requirements Specifications," in *IEEE Std 1233,* R2002 ed., IEEE, 1998, pp. 1-36.

[84] "ISO/IEC/IEEE International Standard - Systems and software engineering -- Software life cycle processes," in *ISO/IEC/IEEE 12207:2017(E)*, 1 ed., 2017, pp. 1-157.

[85] "ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering," in *ISO/IEC/IEEE 29148:2018(E)*, 2018, pp. 1-104.

[86] O. Backstrom, J.-E. Holmberg, M. Jockenhovel-Barttfeld, M. Porthin and A. Taurines, "Software reliability analysis for PSA," Nordic Nuclear Safety Research, Roskilde, Denmark, 2014.

[87] T.-L. Chu, M. Yue, G. Martinez-Guridi and J. Lehner, "Development of Quantitative Software Reliability Models for Digital Protection Systems of Nuclear Power Plants," NUREG/CR-7044, U.S. Nuclear Regulatory Commission, 2013.

[88] T. Shorthill, H. Bao, H. Zhang and H. Ban, "Redundancy-Guided System Theoretic Hazard Analysis," Arxive, 2020.

Appendix A

Application of THERP

The following is an application of the Technique for Human Error Rate Prediction (THERP) for the evaluation of root causal factors for the failure of software systems. The THERP Manual indicates that the "general approach used for HRA has been to identify, analyze, and estimate human error predictions (HEPs) for human tasks that system analysts and human reliability analysts determine could have major impact on system criteria of interest." The actions to identify, analyze, and estimate form the basis of steps used in this application of THERP:

- Define the system failures of interest
- List and analyze the related human operations (perform a task analysis)
- Estimate relevant error probabilities.

Assumptions:

- Because a specific system was unavailable to review, the same SDLC task list was assumed for both the specific RPS and generic safety system software. In contrast, a normal analysis would likely result in differences between the specific system SDLC tasks and the chosen generic SDLC task list.
- To simplify the analysis, the same SDLC task list was used for all software components and modules within the RPS system.
- Persons involved in the creation of requirements: stakeholders (client), engineer, engineering leader, and the project manager:
    - Group actions involve the engineer, engineering leader, and project manager.
- Each of the individuals involved have the required training and expertise to perform their tasks.
- The work environment is optimal.
- There is dependency between tasks.
- There is dependency between the person performing a task and the reviewer.
- More than two reviewers are not considered to provide recovery for a task:
    - The THERP Manual indicates that recovery is not credited to more than two checkers of a routine task [61]. Though some tasks may not be routine, we have chosen two to be the maximum assigned to simplify the analysis.
- The task of reviewing is assigned the same human error probability as the original task.
- Diagnosis actions made by the group are chosen from Table 20-3 in [61], which consists of a diagnosis of abnormal events in a nuclear reactor. It is assumed that the HEP values given for group efforts are appropriate for the creation of requirements, as both require diagnosis of a situation. It was assumed that a group meeting of an hour as a reasonable time to define a scope, purpose, or parameters for operation of a system. Hence, the HEP associated with a 60-minute diagnosis time was selected corresponding to a value of 1E-4.

- THERP indicates when there is no appropriate HEP available, the nominal value of 3E-3 is assigned for errors of omission or for errors of commission (see page 20-13 in [61]). It is an assumption of the case study that for events not clearly omission or commission dominant, the union of the two will be used. Additionally, it is assumed that the omission and commission are independent. Therefore, the resulting value comes from Pr(A or B) = Pr(A)+Pr(B)-Pr(A)*Pr(B) = 5.991E-3.
- To simplify the case study, the difference between SDLC quality is based on the quality of the review processes. Hence, the difference between the specific and general HRA applications is that the reviewers are given moderate dependency for all cases for the general case.

THERP Step 1: Define the system failures of interest.

This step was completed previously in BAHAMAS. Generally, this process involves familiarization with a system, followed by a qualitative assessment (e.g., hazard analysis), in order to find areas where human actions may be a concern [61]. For the example provided here, the event of interest is the failure of the BP to send a trip signal to the logic cabinets of the RPS system.

THERP Step 2: List and analyze the related human operations (perform a task analysis).

The THERP Manual indicates that the task analysis should generally focus on identifying the man and machine interfaces and their associated influence on human performance. In addition, the task analysis should identify problem areas in the design likely to cause human error (e.g., written procedures policies, practices, and people skills). It is assumed that human actions in the SDLC can result in software failure. Hence, the task analysis will focus on the human tasks associated with the SDLC. The case study concerns an HRA analysis of the development of software requirements for the application software of the BPs in use in a representative RPS system:

THERP Step 2.1: Determination of SDLC tasks.

A brief review of past and present international standards available from IEEE has provided a list of some of the key elements associated with the development of software requirements [83] [84] [85]. A similar approach would be made to determine the generic SDLC task, but only the tasks should come from real applications, experience, interviews, and publications. The development of requirements can be grouped into actions associated with preparation, definition, analysis, and management:

- Preparation to define requirements involves:
  - Defining system boundaries and the project scope
  - Creating an organizational project strategy
  - Allocating or arranging for necessary resources
  - Clarifying the project scope with stakeholders.
- Definition of key aspects of the requirements should include:
  - Clarifying the purpose of the system
  - Establishing the modes of operation
  - Defining the system constraints
  - Providing justification for the requirements or specifications
  - Creating a database for the requirements
  - Defining necessary system interfaces
  - Defining system installation requirements
  - Creating and defining a system architecture.

- An analysis of developed requirements should include:
  - A review of system requirements
  - Clearly defined performance measures
  - Feedback from stakeholders
  - Resolution of any issues.
- Management of developed requirements should include:
  - A quality assurance plan via documentation.

The tasks above were used to generate a simplified task list to be used in the case study for both the generic and specific software applications. The simplified task list for the development of requirements for software systems is as follows:

1. Define the scope.
2. Define the purpose.
3. Define the parameters of operation.
4. Define interfaces.
5. Identify applicable codes and standards.
6. Define the quality assurance and testing plan.
7. Create technical specifications.
8. Communicate design specifications.

THERP Step 2.2: Analysis of tasks.

Details regarding the tasks should be clarified. Considerations for written procedures, policies, practices, and people skills are important. Additionally, aspects of the individuals involved in each task, their training, and dependencies should be considered. The idea is to clarify details for each task to make quantification using the THERP methodology easier.

The success or failure of a task may have conditional dependence on the previous tasks. For these cases, five levels of dependence can be assigned. The success of an example task will have zero, low, moderate, high, or complete dependence on the success or failure a prior task [61]. Dependency relationships are assessed for the eight tasks for the development of requirements and for reviewers of those tasks.

Example 1: A manager will be slow to trust assignments made to the new employee. If the manager is responsible of reviewing the work of the new employee, the manager will likely pay great attention to the review. This scenario would be assigned a low dependency relationship between the manager (as a reviewer) and the employee.

Example 2: For a critically important job task, a manager will be concerned that the task be completed correctly. A low dependence value may be assigned between the employee and the manager, despite the manager's complete confidence in the abilities of the employee.

When the task analysis and assumptions have been clarified, details regarding the HEP can be selected for each of the tasks being analyzed, which is the HEP probability of human error without consideration of shaping factors and dependency. Then, the HEP and task analysis results are used to determine the probability of human error using the HRA ET in Step 3 of THERP.

The following are the task details used for the case study:

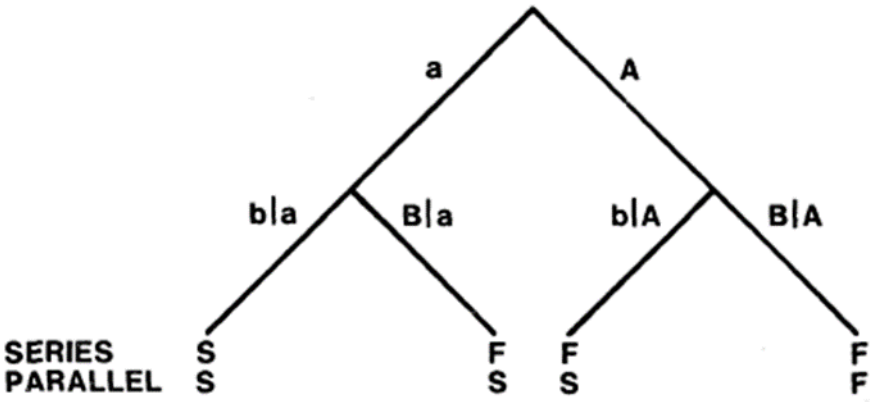
1. Define the scope:
    a. Dependency on previous task: **Zero Dependency**.
    b. Person(s) performing the task: Group.
    c. Person(s) reviewing the task: Stakeholders:
        i. Dependency on previous person(s): **Low Dependency**.
    d. Nominal HEP: **1E-4**.
        i. Assigned from Table 20-3 item #5 from [61] for a single event. (See Assumptions Section.)
2. Define the purpose:
    a. Dependency on previous task: **High Dependency**.
    b. Person(s) performing the task: Group.
    c. Person(s) reviewing the task: Stakeholders:
        i. Dependency on previous person(s): **Low Dependency**.
    d. Nominal HEP: **1E-4**.
        i. Assigned from Table 20-3 item #5 from [61] for a single event. (See Assumptions Section.)
3. Define the parameters of operation:
    a. Dependency on previous task: **High Dependency**.
    b. Person(s) performing the task: Group.
    c. Person(s) reviewing the task: Stakeholders:
        i. Dependency on previous person(s): **Low Dependency**.
    d. Nominal HEP: **1E-4**.
        i. Assigned from Table 20-3 item #5 from [61] for a single event. (See Assumptions Section.)
4. Define interfaces:
    a. Dependency on previous task: **High Dependency**.
    b. Person(s) performing the task: Group.
    c. Person(s) reviewing the task: Stakeholders:
        i. Dependency on previous person(s): **Low Dependency**.
    d. Nominal HEP: **1E-4**.
        i. Assigned from Table 20-3 item #5 from [61] for a single event. (See Assumptions Section.)
5. Identify applicable codes and standards:
    a. Dependency on previous task: **Moderate Dependency**.
    b. Person(s) performing the task: Group.
    c. Person(s) reviewing the task: Stakeholders:
        i. Dependency on previous person(s): **Moderate Dependency**.
    d. Nominal HEP: **1E-4**.
        i. Assigned from Table 20-3 item #5 from [61] for a single event. (See Assumptions Section.)

6. Define the quality assurance and testing plan:
    a. Dependency on previous task: **High Dependency**.
    b. Person(s) performing the task: Engineer.
    c. Person(s) reviewing the task: Engineering Leader; Project Manager:
        i. Dependency on previous person(s): **Moderate Dependency**.
        ii. Dependency on previous person(s): **Moderate Dependency**.
    d. Nominal HEP: **5.991E-3**.
        i. See Assumptions Section.
7. Create technical specifications:
    a. Dependency on previous task: **High Dependency**.
    b. Person(s) performing the task: Engineer.
    c. Person(s) reviewing the task: Engineering Leader; Project Manager:
        i. Dependency on previous person(s): **Low Dependency**.
        ii. Dependency on previous person(s): **Moderate Dependency**.
    d. Nominal HEP: **5.991E-3**.
        i. See Assumptions Section.
8. Communicate design specifications:
    a. Dependency on previous task: **Low Dependency**.
    b. Person(s) performing the task: Engineer.
    c. Person(s) reviewing the task: Engineering Leader; Project Manager:
        i. Dependency on previous person(s): **Low Dependency**.
        ii. Dependency on previous person(s): **Moderate Dependency**.
    d. Nominal HEP: **5.991E-3**.
        i. See Assumptions Section.

The difference in application for the general SDLC quality is that the reviewers are given moderate dependency for all cases.

THERP Step 3: Estimate the relevant error probabilities.

This Section is focused on the third THERP step. The ET limbs represent binary actions for "correct" or "incorrect," the probability for which are conditional [61]. The dependence between limbs of the ET are assigned conditional probabilities based on the relationships identified in the task analysis where the information from the task analysis is used in an HRA ET to determine the probability of human error. The basic tools used for evaluating the HRA ET of the case study are found in Figure 22 and Figure 23, as taken from [61].



TASK "A" = THE FIRST TASK
TASK "B" = THE SECOND TASK
a = PROBABILITY OF SUCCESSFUL PERFORMANCE OF TASK "A"
A = PROBABILITY OF UNSUCCESSFUL PERFORMANCE OF TASK "A"
b|a = PROBABILITY OF SUCCESSFUL PERFORMANCE OF TASK "B" GIVEN a
B|a = PROBABILITY OF UNSUCCESSFUL PERFORMANCE OF TASK "B" GIVEN a
b|A = PROBABILITY OF SUCCESSFUL PERFORMANCE OF TASK "B" GIVEN A
B|A = PROBABILITY OF UNSUCCESSFUL PERFORMANCE OF TASK "B" GIVEN A
FOR THE SERIES SYSTEM:
$Pr[S] = a(b|a)$
$Pr[F] = 1 - a(b|a) = a(B|a) + A(b|A) + A(B|A)$
FOR THE PARALLEL SYSTEM:
$Pr[S] = 1 - A(B|A) = a(b|a) + a(B|a) + A(b|A)$
$Pr[F] = A(B|A)$

Figure 22. HRA ET basics [61].

Level of Dependence	Success Equations	Equation No.	Failure Equations	Equation No.
ZD	$Pr[S_{"N"}\|S_{"N-1"}\|ZD] = n$	(10-9)	$Pr[F_{"N"}\|F_{"N-1"}\|ZD] = N$	(10-14)
LD	$Pr[S_{"N"}\|S_{"N-1"}\|LD] = \frac{1 + 19n}{20}$	(10-10)	$Pr[F_{"N"}\|F_{"N-1"}\|LD] = \frac{1 + 19N}{20}$	(10-15)
MD	$Pr[S_{"N"}\|S_{"N-1"}\|MD] = \frac{1 + 6n}{7}$	(10-11)	$Pr[F_{"N"}\|F_{"N-1"}\|MD] = \frac{1 + 6N}{7}$	(10-16)
HD	$Pr[S_{"N"}\|S_{"N-1"}\|HD] = \frac{1 + n}{2}$	(10-12)	$Pr[F_{"N"}\|F_{"N-1"}\|HD] = \frac{1 + N}{2}$	(10-17)
CD	$Pr[S_{"N"}\|S_{"N-1"}\|CD] = 1.0$	(10-13)	$Pr[F_{"N"}\|F_{"N-1"}\|CD] = 1.0$	(10-18)

Figure 23. Image of a table of equations used for determining conditional probabilities. The equations provide conditional probabilities of success and failure on Task "N," given success or failure on previous Task "N-1," for different levels of dependence [61].

Figure 22 provides options to evaluate a series of parallel systems. The assumption for the case study is that any un-recovered error will result in an error of the whole system (e.g., the series-type system). In addition, it is assumed that the failure of any task simply results in an undetected error in the requirements, but does not prevent the completion of all tasks of the system. The probability of failure can be evaluated using the equations shown in Figure 22. However, it is necessary first to clarify how the use of dependencies have been incorporated into the HRA ET. This is done using the conditional probability equations, shown in Figure 23. Together, Figure 22 and Figure 23 provide the necessary background for the calculation of the HRA ET of the case study, as seen in Figure 24. Figure 25 provides an example of the calculations involved for the first two tasks of the HRA ET in Figure 22. The HRA ET results for the case study are shown in Table 14.

Table 14. Example conditional probability table.

Probability of Human error in:	Medium	Specific	High
Development of Requirements	3.04E-4	1.62E-4	5.31E-5
Development of Software	2.28E-3	2.21E-3	8.88E-4

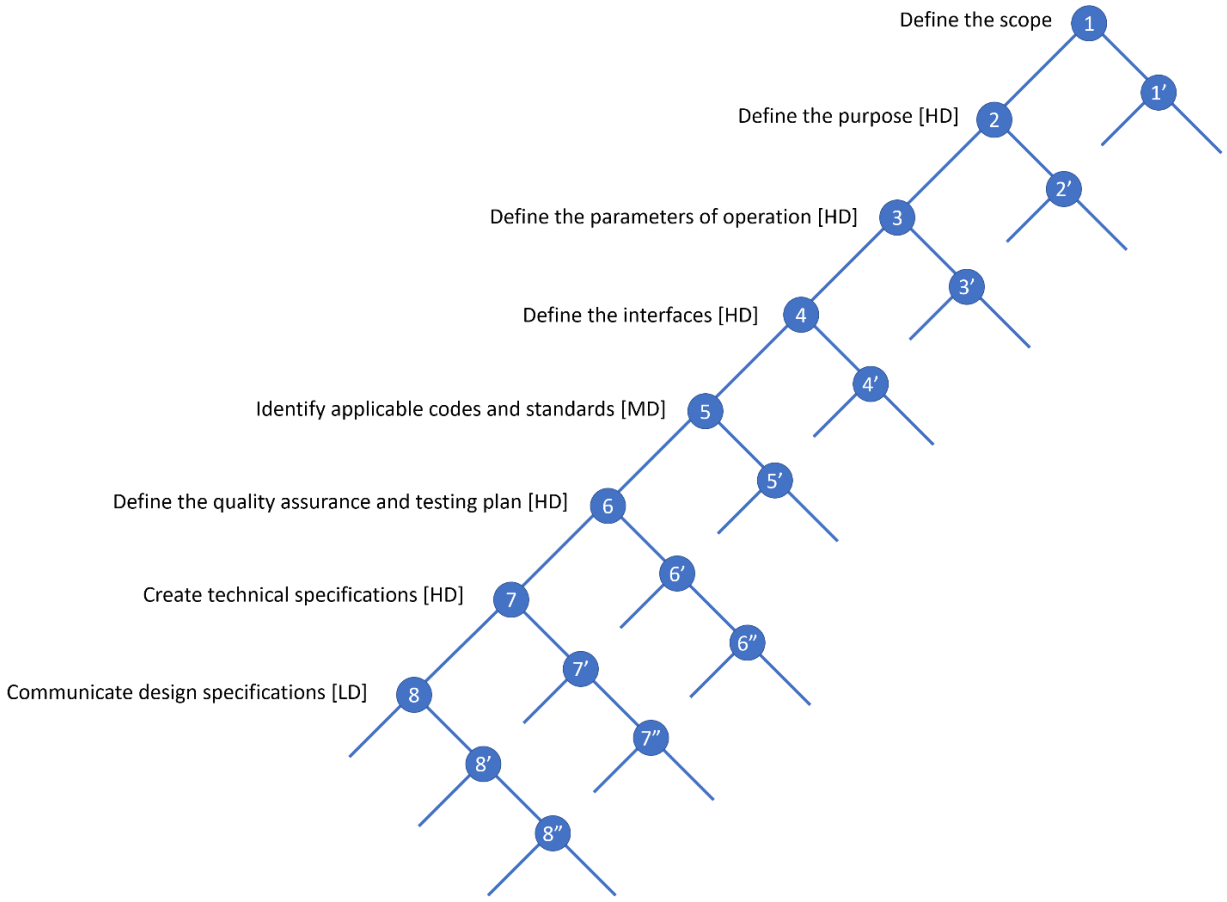
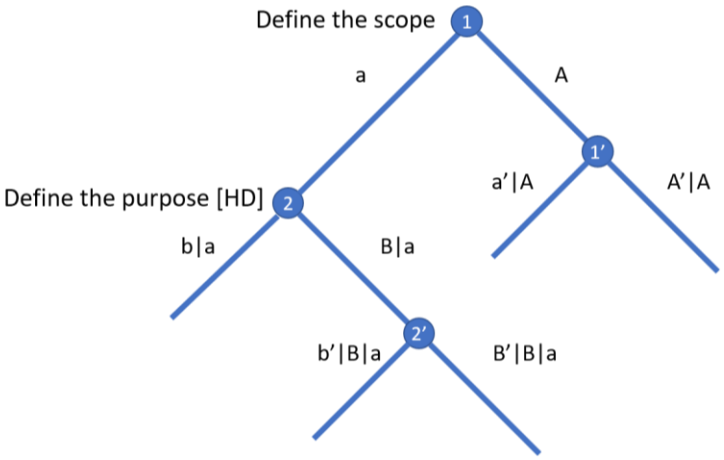


Figure 24. HRA ET for the development of software requirements. Primes and double primes account for the reviewers of tasks.



Task 1:

- Performed by: Group
- Nominal HEP (NHEP): 0.0001
- Dependency on Previous step: not applicable
- Reviewer Dependency: Low Dependency (LD)

A = NHEP	0.0001
a = (1-A) =	0.9999
A'|A (LD)= (1+ A'*19)/20 =	0.050095
a'|A (LD) = 1-A'|A =	0.949905

Task 2:

- Performed by: Group
- Nominal HEP (NHEP): 0.0001
- Nominal Human success probability (NHSP) = 1-NHEP
- Dependency on Previous step: not applicable
- Reviewer Dependency: Low Dependency (LD)

b|a (HD) = (1+NHSP)/2=	0.99995
B|a (HD) = 1-b|a =	0.00005
B'|B|a (LD) = (1+ NHEP*19)/20 =	0.050095
b'|B|a (LD) = (1-B'|B|a) =	0.949905

Figure 25. Example calculations for the evaluation of the probability of human error in the development of software requirements.