INL/CON-20-61124-Revision-0

A Bayesian and HRA-Aided Method for the Novel Reliability Analysis of Software

November 2021

Tate H Shorthill, Han Bao, Hongbin Zhang, Heng Ban

INL/CON-20-61124-Revision-0

A Bayesian and HRA-Aided Method for the Novel Reliability Analysis of Software

Tate H Shorthill, Han Bao, Hongbin Zhang, Heng Ban

November 2021

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

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

A BAYESIAN AND HRA-AIDED METHOD FOR THE NOVEL RELIABILITY ANALYSIS OF SOFTWARE

Tate Shorthill and Heng Ban
Department of Mechanical Engineering and Material Science
University of Pittsburgh
3700 O'Hara Street, Pittsburgh, PA 15261
ths60@pitt.edu; heng.ban@pitt.edu

Han Bao and Hongbin Zhang
Idaho National Laboratory
2525 Fremont Avenue, Idaho Falls, ID 83415
han.bao@inl.gov; hongbin.zhang@inl.gov

dx.doi.org/10.13182/PSA21-34644

ABSTRACT

Technological advancements and nuclear power plant modernization has inspired considerable research in the areas of safety and reliability, yet there remains a lack of consensus for the reliability assessment of digital instrumentation and control (I&C) systems. Motivated by the lack of consensus for reliability analysis methods, this work employs a novel framework that incorporates Bayesian, human reliability, and common-cause failure (CCF) modeling techniques. The novel framework allows the use of state-of-the-art or classical modeling techniques when accounting for human and CCF effects on system reliability. The Bayesian and HRA-Aided Method for the Reliability Analysis of Software (BAHAMAS) is demonstrated by a case study for the quantification of software hazards found in a previous analysis of a digital reactor trip system. The results demonstrate the ability of BAHAMAS to account for human activities during the software development life cycle and their influence on software reliability. BAHAMAS is a flexible tool for extending the coverage of conventional probabilistic risk assessments to include modernized digital I&C systems.

Key Words: Digital I&C, Human Reliability Analysis, Bayesian Belief Networks

1 INTRODUCTION

The implementation of digital systems in future nuclear power plants is almost guaranteed, but technical challenges associated with the risks of digital systems remain. To assess digital designs, Idaho National Laboratory (INL) has initiated a project under the Risk-Informed Systems Analysis (RISA) Pathway of the U.S. Department of Energy's (DOE's) Light Water Reactor Sustainability (LWRS) program to develop a risk assessment technology for supporting digital I&C system upgrades and designs [1]. This integrated Risk Assessment technology for Digital I&C systems (IRADIC technology) has three phases: Phase 1—a hazard analysis—focuses on a qualitative identification of potential hazards; Phase 2—a reliability analysis—focuses on the quantification of identified hazards; and Phase 3 evaluates potential consequences based on the results of the hazard and reliability analyses [1].

Phase 1 developed and demonstrated the hazard analysis of highly redundant, safety-related digital I&C systems [2] [3] [4]. For a case study, a digital reactor trip system (RTS) was selected from the design documents associated with new certification applications to the U.S. Nuclear Regulatory Commission (NRC) [5]. While sufficient for a hazard analysis, these documents created a challenge for the continuation of the same study into Phase 2; there is no publicly available information regarding how the digital software code was implemented.

The goal of our present work is to quantify the software failures identified in Phase 1 by employing a novel approach to reliability analysis. Our motivation comes from the need to overcome a limited data scenario given that most quantitative methods require explicit software details or test data. Additionally, there is a lack of consensus in the risk assessment community; there is no accepted method for reliability analysis of digital I&C software [6]. The structure of the paper is as follows: Section 2 covers the technical background; Section 3 discusses our approach to meeting the needs and limitations identified in Section 2; Section 4 demonstrates the method; and Section 5 addresses the conclusions of the work.

2 BACKGROUND

The numerous approaches to reliability analysis can largely be sorted into the following categories: test-based methods, software reliability growth models (SRGMs), rule-based methods, metric-based methods, and Bayesian belief networks (BBNs) [7]. In addition, many software reliability methods have been formulated to incorporate the timing of events via dynamic modeling [8]. Our selection of any of these methods will depend on their ability to work with limited data. The following is a discussion of each method. Immediately, test-based methods are eliminated as an option. SRGMs estimate software reliability by matching test data with empirical models [7]. Due to their need for test data, they too are not ideal for this work. The rule-based approach assumes a system reliability value can be assigned if that system was designed according to specified rules [7]. Because they do not necessarily depend on test data, the rule-based approach has potential for our work. Metric-based software reliability prediction relies on software attributes (e.g., bugs per line of code) to determine reliability [9]. The metric-based approach provides multiple ways to capture unique features of a software development process, but many metrics still require explicit software details. BBN methods rely on Bayes' theorem and acyclic graphical models to depict the influence of one event on another while also involving dependencies between events [7]. BBNs can incorporate disparate information and expert elicitation to perform assessments [7], which allows them to work well for the limited data scenario. Expert elicitation can be a source of model uncertainty. The dynamic methods will be saved for later research efforts, allowing the focus of our current work to remain on those methods that maintain compatibility and familiarity with conventional PRAs.

In 2018, Chu *et al.* demonstrated BBNs for quantifying the software failure of protection systems [10]. That work incorporated design requirements and software development life cycle (SDLC) processes but required expert elicitation in its demonstration. Like many BBN-based methods, a lack of data tends to drive the need for expert elicitation. However, due to its potential when compared to the other reliability analyses methods and due to the recent publications of Chu *et al.*, our work will continue the application of BBNs for reliability analysis.

3 TECHNICAL APPROACH

This section discusses a novel combination of BBNs, human reliability analysis (HRA), and CCF modeling techniques, for the quantification of software failure probabilities. The following details the foundation for the Bayesian and HRA-Aided Method for the Reliability Analysis of Software (BAHAMAS).

An investigation of software failure causal factors led to several important findings. First, software does not fail in the same manner as mechanical components [11]. Software performs exactly how it has been designed to perform; any unwanted action or behavior is due to faults in the software. Faults due to human errors in the SDLC are the main reason for unwanted software behavior [12]. Our approach investigates human errors, the quantification of which is the subject of human reliability studies or HRA

[13]. Extending the evaluation of software failures to their root human causes allows the quantification of software failure to be done by HRA methods while also minimizing the need for software performance data.

A BBN will be used to model the effects of root errors on software performance. Our approach takes a basic event from a fault tree (FT) (e.g., a child node) and models the influence of causal factors (e.g., parent nodes) on the child node. BBNs are used primarily for diagnostic (inference) or for predictive analyses [14]. This work will focus on predictive analysis. Additional concepts and applications of BBNs, such as diagnosis via Bayesian updating, are beyond the scope of this work. For the evaluation of the BBN, probabilities must be determined for the root causal factors, as well as for the conditional influence the causal factors have on a basic event.

HRA can be used to quantify the root nodes of the BBN that represent human tasks. HRA determines the probability that a human will perform a task incorrectly, but there is not good way to determine what human error has occurred or how many human errors exist (e.g., the HRA is not performed for every keystroke or hand calculation). We can only assume that the human error probability (HEP) indicates whether a task was performed incorrectly due to some unknown number of errors or "faults." Therefore, the HEP for a task can be considered the "probability of faults" for a root node of the BBN.

Both root nodes and the conditional relationships have uncertainties that must be quantified and, if possible, limited. To limit the uncertainty of the conditional relationships, the BBN is formatted in binary terms by using the concept of the "probability of faults." It is assumed that software failure is due to faults and if there are faults in a system or subsystem, then the system can be said to contain faults. Therefore, the root nodes represent fault sources, while the remaining BBN combines the results into "a probability of faults existing in the system," as shown in Fig. 1. As long as failures can be attributed to faults within a system, the BBN can be structured according to Fig. 1.

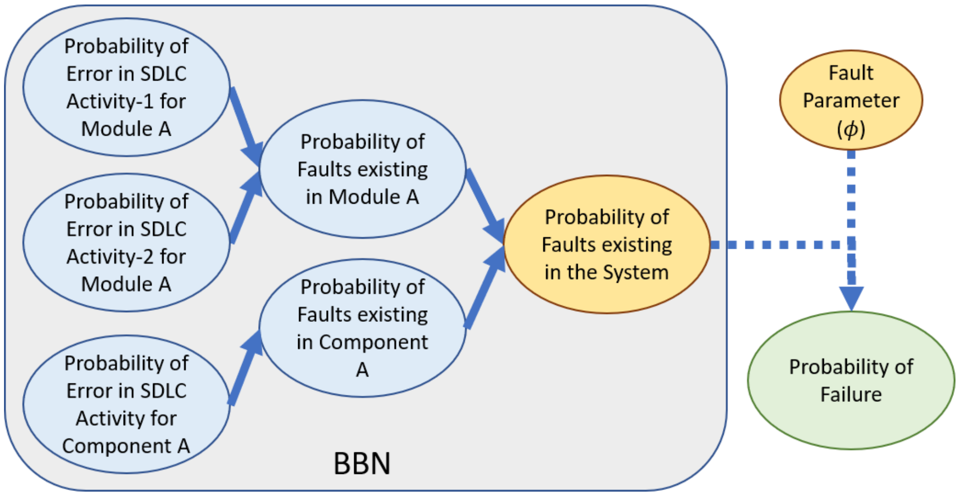
Probability of Error in SDLC Activity-1 for Module A
Probability of Error in SDLC Activity-2 for Module A
Probability of Error in SDLC Activity for Component A
Probability of Faults existing in Module A
Probability of Faults existing in Component A
Probability of Faults existing in the System
BBN
Fault Parameter (ϕ)
Probability of Failure

Figure 1. The general BAHAMAS structure. The BBN provides the probability of faults to be used with the fault parameter for determining the probability of failure. The root nodes are furthest left in the BBN.

Thus far, an approach has been provided to evaluated root causes of a software failure by looking at the probability of faults for the system. It is necessary to determine how to transform a probability of faults into a probability of software failure. Researchers have demonstrated transforming the number of faults into the probability of failure by assuming software failure probability (SFP) is proportional to the number of faults within a software [10] [15]. We assume a similar relationship, defined by fault parameter (ϕ), exists between the SFP and the probability of faults within a software. This relationship is shown in Eq. (1).

$$SFP = P(faults) * \phi \tag{1}$$

The application of Eq. (1) consists of the following. First, a generic SFP is determined based on representative software experience. Second, a BBN is created for the generic software case and evaluated for the generic $P(faults)$. Third, the ϕ is determined. Fourth, the BBN is assigned probabilities for the specific case and evaluated for the specific $P(faults)$. The chief differences between the generic and specific BBN will be associated with SDLC quality. Fifth, the specific $P(faults)$ and ϕ are used to solve for the specific SFP.

Consideration of CCF is an important part of reliability analysis. A CCF is the occurrence of two or more failure events “simultaneously” due to a shared cause [16].There can be a tremendous number of shared causes making explicit modeling infeasible; implicit CCF modeling provides a tractable alternative [16]. This work will rely on the most common implicit method—the beta-factor method [16], which assumes the total failure probability consists of individual and CCF probabilities, the proportions of each represented by a beta. For n components that share a common cause, the beta factor assumes that either a single failure occurs, or all n fail together. Both the individual and CCF are mutually exclusive events; therefore, the total probability represents the possibility of either event, which is found by the sum of the two probabilities. For the probability of failure for an event A:

$$P(A_{total}) = P\left(A_{single}\right) + P(A_{CCF}) \tag{2}$$

$$P\left(A_{single}\right) = (1-\beta)P(A_{total}) \tag{3}$$

$$P(A_{CCF}) = (\beta)P(A_{total}) \tag{4}$$

Based on our approach, the principal assumptions for BAHAMAS are: (1) software failures can be traced to human errors in the SDLC, which may be modeled with HRA methods; (2) in the absence of test data, the reliability of a system can be predicted based on how its SDLC quality compares with existing systems of similar design and purpose; and (3) so long as software failures can be attributed to faults within the system, the BBN in BAHAMAS can be structured as previously shown in Fig. 1. Assumption 1 is supported by examples where the root causes of software failure are associated with errors in requirements specifications, or design defects that can be attributed to SDLC activities [12] [11]. It is a natural extension to assume that design defects exist due to human errors. Assumption 2, which serves as the foundation for Eq. (1), is based on [10] and [15] where a proportional relationship for a software’s predicted performance was generated based on the differences between its SDLC quality and the SDLC quality for existing systems. Lastly, faults are commonly used as a metric for software failures [7] [12] [15].

4 CASE STUDY

This section demonstrates how BAHAMAS can be used to quantify hazards as part of an approach to the reliability analysis of digital systems. For brevity, the case study will demonstrate the quantification of a single software failure identified as part of the hazard analysis of an APR-1400-type four-division digital reactor trip system (RTS) [2]. In addition to the assumptions of the hazard analysis [2],the following are the assumptions of the present study:

1. The hardware failure causal factors that could lead to a software failure have been left out of the BBN because they are grouped with the hardware failures in the FT from Phase 1.
2. CCFs have been limited to all n/n identical components failing to employ the beta-factor method for quantifying CCFs and to simplify the analysis.
3. Generic SFP distributions gathered from operational data are assumed to consist of both individual and CCF sources.

4. The generic and specific models are assumed to have the same SDLC because specific data was unavailable.
5. The generic controller/processor is assumed to have four parts: (1) the input; (2) the memory; (3) the processor; and (4) the output.
6. Setpoint faults will likely cause delayed or spurious trip activation; therefore, setpoint faults are excluded for the failure-on-demand scenario.
7. All root causes are modeled with the same HRA values to simplify the demonstration of the case study. In general application, each root source should be individually evaluated.
8. The human errors associated with the root nodes for requirements development and software development are assumed independent in the BBN. This is because any errors in software development that could be attributed to the requirements should be considered requirements errors already. Thus, errors in the software development nodes are independent of the requirements nodes errors.

4.1 Step 1: Identify a Software Failure of Interest

The first step of BAHAMAS is to select an event to quantify. In this case, the software failure pertains to the bistable processors (BPs) in the RTS. The details of the basic event should include the component, context, and incorrect or erroneous action (e.g., action not provided; action provided and not needed; action provided too early, too late, or in the wrong order; and action provided for too long or stopped too early). In this case, the controller/component is the BP. The action is a failure to provide a trip signal to the logic cabinets. The context is during a plant condition for which a reactor trip is required.

4.2 Step 2: Identify Potential Causes of the Failure

The concepts of a control algorithm that represents a controller's internal beliefs and process model that specifies how a controller acts can help guide an analyst when searching for root causes [17]. When available, actual software code should be used to determine how the components of the system communicate and interact, including the control algorithm and process model. Software failures are due to faults within the process model and control algorithm that may arise from malfunctions in the hardware or software (including firmware). Hardware faults can be from damage associated with physical external events, while software faults are likely intrinsic problems (e.g., incorrect constraints). Further, root causes can be traced to human actions made during the SDLC. The BP is assumed to be a controller consisting of four components: (1) an input module; (2) an output module; (3) a central processing unit (CPU); and (4) the memory. The inputs to the process model are the sensor signals for reactor trip calculation. Outputs from the process model are state variables that are used by the control algorithm to determine whether to trip the reactor. The control algorithm relies on the output module, memory, and CPU for the performance of its functions.

The case study excludes the details of environmental causes and other hardware-based causes as indicated in the assumptions. Therefore, software failure is left to those causes strictly associated with the software of the BP and itself. These software failures can be traced to human errors associated with the SDLC. The case study divides the SDLC into two activities: (1) the development of design requirements; and (2) the development of the application software (i.e., fulfillment of design requirements).

4.3 Step 3: Organize Potential Causes of the Failure into a Bayesian Network

The causes identified in Step 2 are placed into a BBN following the format shown in Fig. 1. The resultant BBN is shown below in Fig. 2. The intermediate nodes of the BBN will vary by application, but the final node will always represent the probability of faults existing in the system. As discussed in Step 2, only SDLC causal factors are used in this BBN. Currently, our approach does not differentiate the effects of human errors; thus, they may have either a single or common effect. Therefore, the output of the BBN represents the combined influence of single and common causes. Ultimately, this form of the BBN output matches well with the assumptions used in implicit CCF modeling methods (e.g., the beta-factor method).

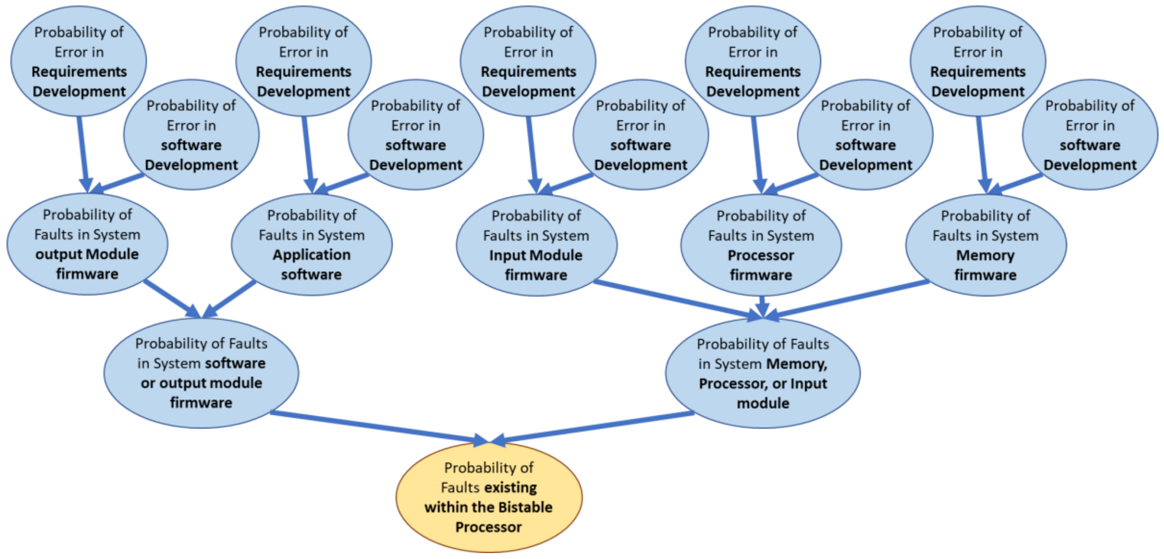
Probability of Error in Requirements Development
Probability of Error in software Development
Probability of Faults in System output Module firmware
Probability of Faults in System Application software
Probability of Faults in System Input Module firmware
Probability of Faults in System Processor firmware
Probability of Faults in System Memory firmware
Probability of Faults in System software or output module firmware
Probability of Faults in System Memory, Processor, or Input module
Probability of Faults existing within the Bistable Processor

Figure 2. The BBN for the BPs of the RTS.

4.4 Step 4: Determine the Fault Parameter

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

4.4.1 Step 4.1: Determine the Generic Software Failure Probability

The generic SFP is used to represent the operational experience of similar software systems. In this work, "similar" means software with a similar purpose (e.g., safety-related software), development practices, and expectations. The software failure experience of similar systems should match the failure of interest (i.e., failure-on-demand, spurious action, wrong timing, etc.) A hierarchal Bayesian analysis [18] was used to estimate the SFP for the generic population. The process involves making an estimation of the distribution of the probability of interest, called "a prior," and then updating the estimation using observations. Together, the prior and observations (e.g., operational data) are used to estimate a distribution for the generic SFP.

A prior distribution was selected based on the highest safety integrity level assigned by International Electrotechnical Commission (IEC) standard 61508 is SIL 4 [7], corresponding to a probability of failure on demand <1E-4 for safety systems. The prior distribution is assumed to have a peak value associated with 1E-4 and the values of the distribution should fall between zero and one. Based on these assumptions, a lognormal distribution was chosen with the σ-parameter = 1.8 and μ-parameter = -5.97.

Next, the update data was selected from operational experience of safety-related software protection systems recorded in NUREG/CR-7233. The data[1] showed 4457 demands for the safety-related systems, of which there were zero failures [10]. The lognormal prior was updated using the data from [10] resulting in the SFP distribution shown in Table 1. The calculation was performed using the INL/NRC reliability calculator website [19]. The results are shown in Table 1. Note that it is reasonable to assume that either single or common causes could lead to failures found in the sample safety systems used for the generic SFP.

[1] The data represents complete system failures and will likely result in an overprediction of our results. A better option would be to use data that corresponds with subcomponent or subsystem failures.

Thus, the SFP distribution, unless otherwise verified, is assumed to represent both CCF and single failure contributions.

Table 1. 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

4.4.2 Step 4.2: Assign Root Node Probabilities Based on the quality of the Generic Software

This step provides probabilities for the root nodes of the BBN based on a representative or generic safety software. This generic software system is intended to represent the general quality in verification, validation, and other SDLC activities associated with the set of software used to find the generic SFP. The technique for human error-rate prediction (THERP) was the HRA method selected for this analysis because it is one of the most widely known methods, having been in use for over 50 years [20]; however, other HRA methods could be used as well. The details of our application of THERP can be found in [3]. The HRA results for medium SDLC quality for requirements and software development are 3.04E-4 and 2.28E-3, respectively.

4.4.3 4.3: Assign Conditional Probabilities to the BBN

The BBN requires conditional relationships between parent and child nodes. The BBN is constructed such that there are binary, conditional relationships between each node, thereby limiting the uncertainty sources of the model. Table 2 shows the format of conditional probability tables used for the analysis.

Table 2. Example of a conditional probability table

Parent Nodes	State of Node		State of Node	
Error in SDLC Activity 1 for Module A	Y		N	
Error in SDLC Activity 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

4.4.4 Step 4.4: Evaluate the BBN for the Probability of Faults for the Generic Software

The BBN uses root nodes and conditional probabilities to find the probability of the final node of the BBN. The calculation consists of evaluating the marginal probability of each node of the BBN. The marginal probability refers to the probability of a specific state within the joint probability of all other states [21]. Eq. (5). provides the equation for marginal probability [21] as:

$$P(A = a_i) = \sum_{j=1}^{m} P(a_i, b_j) \tag{5}$$

Consider a BBN of two binary-state nodes: Node A (parent) and B (child). Each has two possible states: True or False (i.e., $A = \{a_1 = True, a_2 = \ False\}$ and $B = \{b_1 = True, b_2 = \ False\}$). In this example, the probability for each state of Node A is known, as well as the conditional probability of Node B given each state of Node A. Given these known values, Eq. (5) can be modified by the joint probability and commutative properties to find the marginal probability of B = b_1 = true.

$$
\begin{aligned}
(B = b_1 = True) &= \sum_{j=1}^{2} P(b_1, a_j) = P(a_1, b_1) + P(a_2, b_1) = P(a_1)P(b_1|a_1) + P(a_2)P(b_1|a_2) \\
&= P(A = True)P(B = True|A = True) + P(A = False)P(B = True|A = False)
\end{aligned}
\tag{6}
$$

The BBN can be evaluated based on the example above by starting with the first generation of nodes (i.e., those whose parents are the root nodes of the BBN). The marginal probability is evaluated for each child node of the first generation. Then, the marginal probability for each first-generation node is used as a parent probability for the next generation of marginal probability calculations. The process repeats until the final node marginal probability is determined. The probability of faults was found to be 1.285E-2.

4.4.5 Step 4.5: Determine the Value for ϕ

The value of the fault parameter (ϕ) is found using Eq. (1). The case study used the mean value from the generic SFP distribution from Step 4.1 and $P(faults)$ from Step 4.4 to get ϕ = 1.59E-2.

4.5 Step 5: Determine the Probability for the Failure of Interest

The specific SFP is found using the same format as discussed in previous steps and modified root nodes. The root nodes were modified to reflect the case-specific SDLC quality, 1.62E-4 and 2.21E-3 for the requirements and software development, respectively. Then, the case-specific probability of faults is found to be 1.179E-2. Finally, using Eq. (1), the SFP can be found (e.g., SFP = 1.87E-4). As mentioned previously, the SFP represents the total software failure. In Step 6, individual contributions of single and CCF probabilities will be determined.

4.6 Step 6: Evaluate the Single and CCF Probability

This step relies on CCF modeling methods to determine the single and CCF probabilities. The beta-factor method is used for this case study, but other methods may also be used. A beta value of 0.05 is indicated for safety systems developed with good engineering practices [22]. The calculation of individual and CCF failure probability is performed using Eqs. (2), (3), and (4). The single and CCF failure probabilities are 1.8E-4 and 9.4E-6, respectively.

5 CONCLUSION

In 2019, INL initiated a project under the LWRS-RISA Pathway to develop a risk assessment technology (IRADIC) for supporting digital I&C system upgrades and designs [1]. The BAHAMAS method developed in this work aims to support the reliability analysis of digital I&C software systems by integrating BBN, HRA, and CCF modeling techniques. By using familiar concepts and a clearly defined structure, BAHAMAS is designed to provide convenient application for conventional PRAs. Additionally, this method provides a means of analyzing software systems with complex structures or where operational data may be limited. Instead of relying on testing data, BAHAMAS assumes that software failures can be traced to human errors in the SDLC and modeled with HRA. Therefore, BAHAMAS can be applied early in the SDLC to assist in design considerations, which may help reduce the software development timeline and ultimately save development costs. In conclusion, BAHAMAS provides a flexible and useful tool for the quantification of digital I&C software failures.

The case study shows the BP provides a single failure probability of 1.77E-4 and a CCF probability of 9.34E-6 for failure on demand. Though the results fall in the category of IEC 61508 SIL 4 (i.e., 1E-4 or less) [7], an uncertainty range for the results is still needed. It should be noted that each analysis is unique; therefore, the BBN and model parameters (i.e., ϕ) should be determined on a case-by-case basis. The CCF results are limited to the scenario of all n/n components failing. Future efforts will investigate options to incorporate combinations of CCFs into the reliability analysis. The case study presented relied on THERP for the quantification of faults in the SDLC. While a classic and well-used method, THERP is aging, and a different HRA method may prove to be better suited for the evaluation of SDLC. The primary assumptions 1 and 3 are largely justified by existing research as mentioned in Section 3. In contrast, the impact of assumption 2 remains a question of verification and validation. Additional investigations comparing BAHAMAS to another reliability method will clarify the impact of assumption 2. Future efforts will focus on refining BAHAMAS and investigating its uncertainty sources.

6 ACKNOWLEDGMENTS

This submitted manuscript was authored by a contractor of the U.S. Government under DOE Contract No. DE-AC07-05ID14517. Accordingly, the U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for U.S. Government purposes. This work was also supported by consultations and contributions from the following individuals: Ronald Boring, Thomas Ulrich, Jooyoung Park, Torrey Mortenson, and Zhegang Ma from INL; and Edward Chen from North Carolina State University.

7 REFERENCES

1. H. Bao, H. Zhang, and K. Thomas, "An Integrated Risk Assessment Process for Digital Instrumentation and Control Upgrades of Nuclear Power Plants," INL/EXT-19-55219, Idaho National Laboratory, Idaho Falls, ID, USA (2019).
2. T. Shorthill, H. Bao, H. Zhang, and H. Ban, "A Redundancy-Guided Approach for the Hazard Analysis of Digital Instrumentation and Control Systems in Advanced Nuclear Power Plants," https://arxiv.org/abs/2005.02348 (2020).
3. H. Bao, T. Shorthill, and H. Zhang, "Redundancy-guided System-theoretic Hazard and Reliability Analysis of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants," INL/EXT-20-59550, Idaho National Laboratory, Idaho Falls, ID, USA (2020).
4. H. Bao, T. Shorthill, and H. Zhang, "Hazard analysis for identifying common cause failures of digital safety systems using a redundancy-guided systems-theoretic approach," *Ann. Nucl. Energy,* **vol. 148**, art. 107686 (2020).
5. Korea Electric Power Corporation, Korea Hydro & Nuclear Power Co., Ltd, "APR1400 Design Control Document Tier 2. Chapter 7: Instrumentation and Controls," U.S. Nuclear Regulatory Commission, Rockville, MD, USA (2018).

6. 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, Rockville, MD, USA (2013).

7. T.-L. Chu, M. Yue, G. Martinez-Guridi, and J. Lehner, “Review of Quantitative Software Reliability Methods,” BNL-94047-2010, Brookhaven National Laboratory, Upton, NY, USA (2010).

8. A. Mosleh, “PRA: A Perspective on Strengths, Current Limitations, and Possible Improvements,” *Nucl. Eng. Technol.,* **vol. 46**, no. 1, pp. 1–10 (2014).

9. Y. Shi, M. Li, S. Arndt, and C. Smidts, “Metric-based software reliability prediction approach and its application,” *Empir. Softw. Eng.,* **no. 22**, pp. 1579-1633 (2017).

10. 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 Commission, Rockville, MD, USA (2018).

11. Electric Power Research Institute (EPRI), “Methods for Assuring Safety and Dependability when Applying Digital Instrumentation and Control Systems,” 3002005326, Palo Alto, CA, USA (2016).

12. M. Muhlheim, and R. Wood, “Technical Basis for Evaluating Software-Related Common-Cause Failures,” ORNL/SR-2016/130, Oak Ridge National Laboratory, Oak Ridge, TN, USA (2016)

13. 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, Rockville, MD, USA (1983).

14. A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, “Comparing Fault Trees and Bayesian Networks for Dependability Analysis,” *Proceedings of the 18th International Conference on Computer Safety, Reliability, and Security (SAFECOMP99)*, Toulouse, France, September 27-29, LNCS 1698, pp 310-322 (1999).

15. 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,” *Ann. Nucl. Energy,* **vol. 120**, pp. 62-73 (2018).

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

17. N. G. Leveson and J. P. Thomas, “STPA Handbook,” MIT Partnership for Systems Approaches to Safety and Security (2018).

18. 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).

19. Idaho National Laboratory, "Reliability Calculator Web Site," U.S. Nuclear Regulatory Commission, https://nrcoe.inl.gov/radscalc/Default.aspx (2020).

20. R. Boring, "Fifty years of THERP and Human Reliability Analysis," *Joint Probabilistic Safety Assessment and Management and European Safety and Reliability Conference* (2012).

21. N. Fenton and M. Neil, *Risk Assessment and Decision Analysis with Bayesian Networks*, 2nd ed., CRC Press LLC (2018).

22. H. W. Jones, "Common Cause Failures and Ultra Reliability," *42nd International Conference on Environmental Systems*, San Diego, CA, July 15-19 (2012).