INL/RPT-24-80888

Light Water Reactor Sustainability Program

An Integrated Framework for Risk Assessment of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants: Methodology Advancement and Application

September 2024

U.S. Department of Energy

Office of Nuclear Energy

INL/RPT-24-80888

Light Water Reactor Sustainability Program

An Integrated Framework for Risk Assessment of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants: Methodology Advancement and Application

Tate Shorthill[1], Edward Chen[1], Jisuk Kim[1], Congjian Wang[1], Paridhi Athe[2], Han Bao[1], Nam Dinh[2], Svetlana Lawrence[1]

September 2024

[1]Idaho National Laboratory
Idaho Falls, ID 83415
[2]North Carolina State University
Raleigh, NC 27695

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.lwrs.gov

Prepared for the
U.S. Department of Energy
Office of Nuclear Energy
Light Water Reactor Sustainability Program

EXECUTIVE SUMMARY

This report documents activities performed by Idaho National Laboratory (INL) during fiscal year (FY) 2024 for the U.S. Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program, Risk Informed Systems Analysis (RISA) Pathway, Digital Instrumentation and Control (DI&C) Risk Assessment project. The goal of the RISA Pathway is to optimize safety margins and minimize uncertainties to achieve economic efficiencies while maintaining high levels of safety. This is accomplished by providing the scientific basis to better represent safety margins and factors that contribute to cost and safety, and by developing new technologies that reduce operating costs.

In FY 2019, the RISA Pathway initiated a project to develop a technical basis to support effective and secure DI&C technologies for digital upgrades/designs. Based on this goal, INL has developed a risk assessment framework to (1) provide a risk-informed analysis process for the DI&C upgrades that integrates hazard analysis, reliability analysis, and consequence analysis; (2) incorporate risk-informed tools to address common cause failures (CCFs) and quantify corresponding failure probabilities for DI&C technologies; (3) evaluate the impact of digital failures at the component, system, and plant levels; and (4) enable insights and suggestions on designs so as to manage risks, thus supporting the development and deployment of advanced DI&C technologies in nuclear power plants.

The research efforts for FY 2024 encompass methodology advancement and application. They include: (1) implementation of a natural language processing (NLP) tool to expedite key aspects of the reliability analysis methods developed by INL; (2) advances to support intersystem CCF analysis by providing guidance for and identification of coupling mechanisms that may contribute to CCF; (3) investigation of how generative artificial intelligence (GAI) tools can aid in hazard analysis and diversity and defense in depth (i.e., D3) assessments; (4) industry collaboration, enabling demonstration of INL's ability to support risk assessment of DI&C systems at early and late stages of development; (5) a roadmap for the development of a software for each of INL's risk assessment tools; (6) development of a theory and manual for a risk quantification methodology; (7) and development of a reliability analysis for machine learning (ML)-integrated control systems.

This project provided the INL team with insights to improve and further develop the DI&C risk assessment framework to respond to the needs of the nuclear industry. This work demonstrates the successful benefits achievable through industry collaboration. The INL research team aims to continue such collaborations and provide science- and technology-based solutions and insights that support long-term operation of the existing nuclear fleet.

This report is primarily intended for DI&C designers, engineers, and probabilistic risk assessment practitioners. This includes stakeholders (e.g., the nuclear utilities and regulators who consider the deployment and upgrading of DI&C systems), DI&C software developers and reviewers, and cybersecurity specialists.

Note that the analyses performed with in this work are for public methodology demonstration and do not fully reflect the reality of any specific digital control system. The results were obtained based on limited design and testing information.

CONTENTS

FIGURES

TABLES

ACRONYMS

AEB	automatic emergency brake
AI	artificial intelligence
BAHAMAS	Bayesian and HRA-aided Method for the Reliability Analysis of Software
BBN	Bayesian belief network
CCCG	common cause component group
CCF	common cause failure
CCS	component control system
CDF	core damage frequency
CF	coverage factor
CPCS	core protection calculator system
D3	diversity and defense in depth
DCD	design control document
DET	dynamic event tree
DI&C	digital instrumentation and control
DOE	U.S. Department of Energy
DPS	diverse protection system
ESF	engineered safety feature
ESFAS	Engineered Safety Features Actuation System
ET	event tree
FM	foundation model
FT	fault tree
FTA	fault tree analysis
FY	fiscal year
GEH	GE-Hitachi Nuclear Energy
GPT	generative pre-trained transformer
HEP	human error probability
HRA	human reliability analysis
ID	in-distribution
IEC	International Electrotechnical Commission
INL	Idaho National Laboratory
LC	loop controller
LCL	local coincidence logic
LLM	large language model

LWRS	Light Water Reactor Sustainability
MBSE	model-based systems engineering
ML	machine learning
ODC	orthogonal-defect classification
ORCAS	Orthogonal-Defect Classification for Assessing Software Reliability
PPS	plant protection system
PWROG	Pressurized Water Reactor Owners Group
R&D	research and development
RAG	retrieval-augmented generation
RESHA	Redundancy-guided Systems-theoretic Hazard Analysis
RISA	Risk Informed Systems Analysis
RTS	reactor trip system
SDLC	software development life cycle
SHIELDS	Software for Hazard Identification and Evaluation of Digital Systems
SP	selective processor
SQA	software quality assurance
SSCs	systems, structures, and components
STPA	systems-theoretic process analysis
SysML	system modeling language
UCA	unsafe control action
UIF	unsafe information flow

AN INTEGRATED FRAMEWORK FOR RISK ASSESSMENT OF SAFETY-RELATED DIGITAL INSTRUMENTATION AND CONTROL SYSTEMS IN NUCLEAR POWER PLANTS: METHODOLOGY ADVANCEMENT AND APPLICATION

1. INTRODUCTION

This report documents the activities performed by Idaho National Laboratory (INL) during fiscal year (FY) 2023 for the U.S. Department of Energy (DOE) Light Water Reactor Sustainability (LWRS) Program, Risk Informed Systems Analysis (RISA) Pathway, Digital Instrumentation and Control (DI&C) Risk Assessment project. This work documents efforts to mature and apply methodologies developed and documented in previous reports, including [1], [2], [3], [4], [5]. The LWRS program, sponsored by DOE and coordinated through a variety of mechanisms and interactions with industry, vendors, suppliers, regulatory agencies, and other industry research and development (R&D) organizations, conducts research to develop technologies and other solutions to improve the economics and reliability, sustain the safety, and extend the operation of the U.S. fleet of nuclear power plants. The LWRS program has two objectives in maintaining the long-term operation of the existing fleet: (1) to provide industry with science- and technology-based solutions to exceed the performance of the current business model, and (2) to manage the aging of systems, structures, and components (SSCs) so that nuclear power plant lifetimes can be extended, and the plants can continue to operate safely, efficiently, and economically.

The RISA Pathway aims to support decision-making related to economics, reliability, and safety, providing integrated plant systems analysis solutions via collaborative demonstrations in order to enhance the economic competitiveness of the operating fleet. The goal of the RISA Pathway is to conduct R&D to optimize safety margins and minimize uncertainties in order to achieve economic efficiencies while still maintaining high levels of safety. This is accomplished in two ways: (1) by providing the scientific basis to better represent safety margins and factors that contribute to cost and safety, and (2) by developing new technologies that reduce operating costs.

One of the research efforts under the RISA Pathway is the DI&C Risk Assessment project, which was initiated in FY 2019 to develop a risk assessment strategy for delivering a strong technical basis to support effective, licensable, and secure DI&C technologies for digital upgrades and designs [1]. This aim is achieved via hazard, reliability, and consequence analyses, which provide metrics to improve the development of DI&C systems (see Figure 1). The efforts of this project have led to the development of the risk assessment framework shown in Figure 1, a framework that aims to:

- Provide a best-estimate, risk-informed capability to quantitatively and accurately estimate the safety margin obtained from plant modernization, especially for safety-related DI&C systems.
- Support and supplement existing advanced risk-informed DI&C design guides by providing quantitative risk information and evidence.
- Offer a capability for evaluating the design architectures of various DI&C systems in order to support system design decisions and diversity and redundancy applications.
- Ensure the long-term safety and reliability of safety-related DI&C systems.
- Reduce cost uncertainties and support integration of DI&C systems into nuclear power plants.

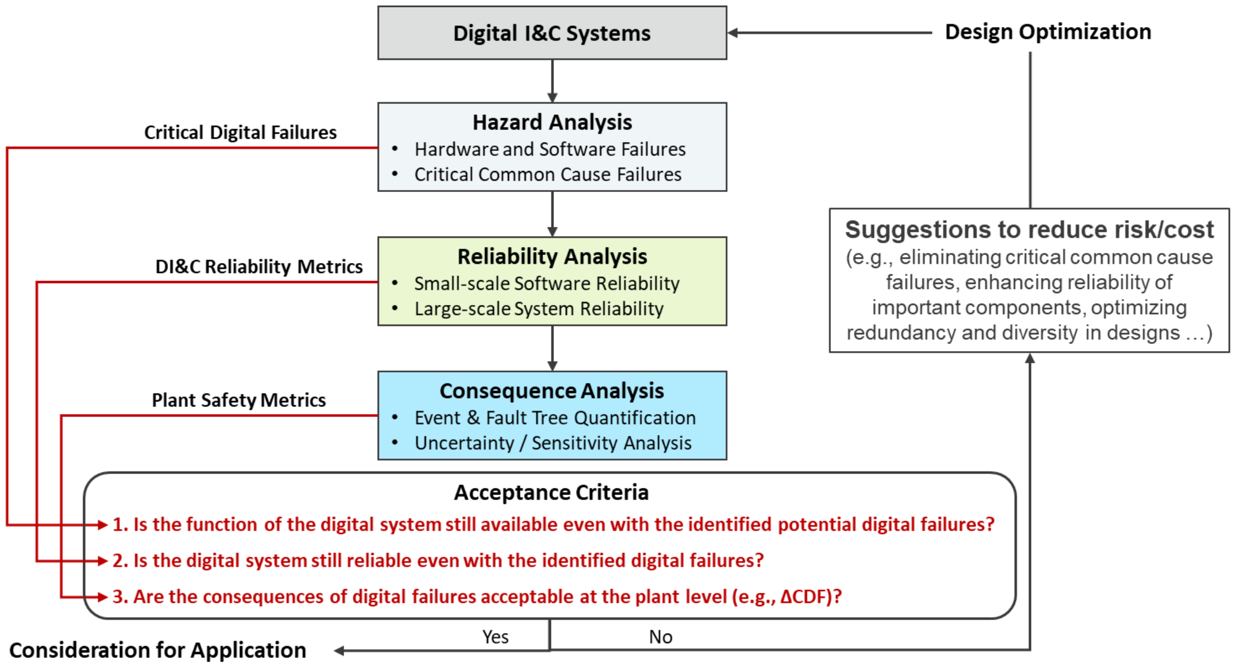


Figure 1. Schematic of the proposed risk assessment framework for safety-critical DI&C systems.

The DI&C risk assessment framework consists of several tools for supporting risk assessment [6]. Figure 2 shows a condensed view of the methods and tools in the framework. The four main tools developed by the INL team are:

- The Redundancy-guided System-theoretic Hazard Analysis (RESHA) is a tool for identifying hazards, with particular emphasis on identification of potential software CCFs within safety-related DI&C systems.
- The hybrid CCF model is a parametric CCF model similar to the beta-factor CCF model, but it has been tailored specifically for modeling software CCFs.
- The Bayesian and Human Reliability Analysis (HRA)-aided Method for the Reliability Analysis of Software (BAHAMAS) is a software failure quantification method for use during data-limited conditions, such as early design stages or when test data is unavailable.
- The Orthogonal Defect Classification for Assessing Software (ORCAS) is a software failure quantification method for use during the testing stages of software development. It tracks the occurrence of software defects during development to predict potential software failure modes.

The framework follows the format of identification (hazard analysis), quantification (reliability analysis, and evaluation (consequence analysis). RESHA helps I&C designers and engineers address both hardware and software CCFs and qualitatively analyze their effects on a system [7] [8]. The second part in risk analysis is the reliability analysis, which for this framework includes quantifying basic events for a fault tree (FT) analysis, including both software and hardware failures. For the proposed framework, two methods (i.e., BAHAMAS and ORCAS) were developed to support software failure quantification [9] [10]. Finally, consequence analysis is conducted to quantitatively evaluate the impact of digital failures on overall plant risks by assessing the affected behaviors and responses. Some digital-based failures may initiate an event or scenario never analyzed before (e.g., a failure mode only applicable to a digital system), potentially challenging plant safety. The framework provides a means for such events to be evaluated. Uncertainty and sensitivity analyses are performed to evaluate the safety and risk significance of target components and subsystems as they relate to DI&C system reliability and plant safety [11].

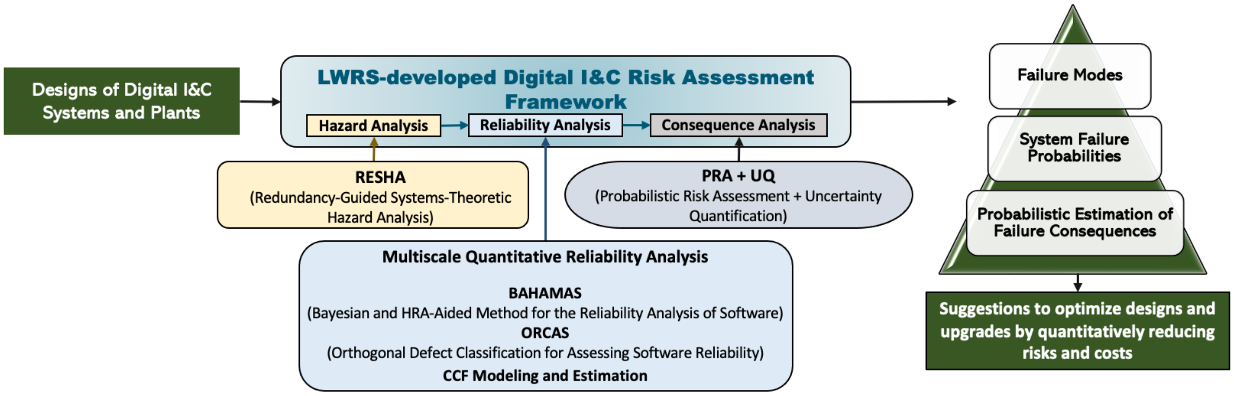


Figure 2. Flexible and modularized structure of the proposed risk assessment framework for safety-related DI&C systems.

This project's research efforts during FY 2019–2022 focused on methodology development and the demonstration of the proposed framework. In FY 2023, the framework was presented for technical peer review and stakeholder feedback. Peer review activities included coordinating the reviews performed by a group of industry stakeholders, documenting the peer review feedback, and providing resolutions and responses to the peer review comments. The comments from technical peer reviewers, along with the resolutions and responses to these comments, are outlined in the peer review report [12] published in March 2023. Additionally, collaborations with the nuclear industry were initiated to support the reliability and risk assessment of their DI&C systems via the proposed framework. The complete analysis and results are documented in [13].

This project's research efforts for FY 2024 emphasized methodology refinement and capability exploration. Specifically, these efforts improve the applicability, usability, and capabilities of the tools developed by INL to support industry needs for performing risk assessment. The remaining sections of this report are organized as follows. Section 2 describes the methodology refinement for ORCAS, including the use of natural language processing (NLP) techniques to relieve the elicitation burden felt by its developers and users. Section 3 provides updates and clarification for supporting intersystem CCF analysis. Sections 4 and 5 summarize collaborative R&D efforts with industry. Section 6 addresses current-year work concerning how GAI can aid in hazard analysis, including diversity and defense in depth assessments. Section 7 proposes a software development roadmap that will be a focus for future efforts. The goal of this roadmap is the development of the Software for Hazard Identification and Evaluation of Digital Systems (SHIELDS). Section 8 addresses the conclusions reached over the course of this work. Appendices A and B include the BAHAMAS theory and user guide, respectively. Finally, Appendix C addresses reliability analysis of machine learning (ML)-integrated control systems.

2. ORCAS METHODOLOGY REFINEMENT AND IMPROVEMENT

The INL team has developed two different software failure quantification methods, each for different application conditions: (1) BAHAMAS [9] for limited-data conditions and (2) ORCAS for data-rich analysis. Whereas BAHAMAS integrates HRA into a Bayesian belief network (BBN), ORCAS applies a white-box software invasive testing and modeling strategy to trace and identify the software defects that can potentially lead to software failures. The approach is a root-cause analysis methodology focused on comprehensive testing and utilizes orthogonal-defect classification (ODC) to correlate programmer development experience with software reliability. The failure data collected from testing strategies is then combined with software reliability growth models and linear reliability models to quantify the software failure probability of specific software failure events.

However, a limitation of the ORCAS methodology, as identified in an earlier milestone report [6], is its large database requirement used to adequately form correlation tables for probability quantification.

These correlation tables are critical, as they relate root cause defects discovered through the SDLC to the probable resultant software failure. As these correlation tables are generated from empirical data, aleatoric and epistemic uncertainty is a concern when the dataset is small. These uncertainties can reach or exceed 100% and can significantly limit the methodology, see Table 1 below (i.e., row 3, column 4). These uncertainties arise from the implicit uncertainty in human judgement when addressing software defects. As there is no unified solution encompassing every software defect, different developers may utilize different methods to address issues. Table 1 shows the correlation table highlighting the percent uncertainty. While this uncertainty cannot be removed, uncertainty bias from the data can be reduced by utilizing additional software development data. Note that in Table 1, not all rows sum to 1 due to rounding errors.

The processing of software defect data is resource exhaustive. To generate the data seen in Table 1, approximately 1,000 reports had to be manually inspected and classified by a researcher. Classification of each report took approximately 5–10 minutes. Furthermore, as these documents serve to describe software defects, human subjectivity on the part of the researcher also contributes to uncertainty in the correlation table. Given that the correlations shown in the table are believed to exist, it is hypothesized that an NLP model could also develop a similar correlation. In this case, to improve the ORCAS methodology, the NLP model Lbl2Vec is demonstrated to automate analyses of software defect data [14]. Utilization of an NLP can reduce human bias and subjectivity in document processing, while also improving consistency in document classification.

Table 1. ORCAS correlation table with percent uncertainty for software failure probability quantification.

	UCA/UIF-A	**UCA/UIF-B**	**UCA/UIF-C**	**UCA/UIF-D**
Algorithm	$0.22 \pm 23\%$	$0.53 \pm 13\%$	$0.12 \pm 36\%$	$0.13 \pm 14\%$
Assignment	$0.29 \pm 53\%$	$0.67 \pm 19\%$	$0.05 \pm 159\%$	0.00*
Checking	$0.22 \pm 34\%$	$0.54 \pm 21\%$	$0.10 \pm 71\%$	$0.14 \pm 91\%$
Function	$0.25 \pm 61\%$	$0.52 \pm 12\%$	$0.16 \pm 58\%$	$0.07 \pm 291\%$
Interface	$0.26 \pm 25\%$	$0.58 \pm 15\%$	$0.09 \pm 102\%$	$0.07 \pm 59\%$
Timing	$0.10 \pm 26\%$	$0.19 \pm 28\%$	$0.52 \pm 40\%$	$0.19 \pm 28\%$
*Note: no data				

2.1 Methodology

The overall methodology for dataset preprocessing, model construction, and performance evaluation is shown in Figure 3, in which the dashed boxes are the inputs and outputs of the methodology, the arrows are processes between steps, and the solid boxes are intermediate step outcomes. Each box has a label (A through H) describing what is required and completed by a process arrow. Boxes A and D are related to the preprocessing of defect reports and the construction of training datasets; boxes B, C, and E are related to model construction and development; and boxes G, F, and H are related to model efficacy evaluations. Processes are labeled via their start and end connecting boxes (e.g., AB), and describe what was completed between steps. The various boxes and processes are described in the following sections.

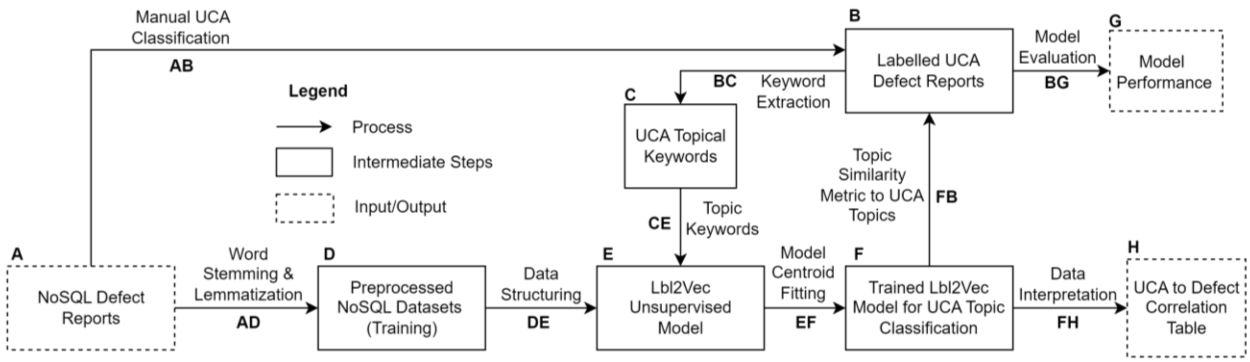


Figure 3. Overall model development and evaluation workflow.

2.1.1 Dataset Construction and Preprocessing

Four different datasets were used to evaluate and train the Lbl2Vec model. Three defect datasets were retrieved from [15], representing three popular NoSQL database management software systems: MongoDB, Cassandra, and HBase (Box A). The named datasets contain 1,618, 1,095, and 1,383 defect reports, respectively. These datasets were selected due to the data having already been preprocessed to a certain extent. Specifically, the defect reports are labeled based on (a) at which point during the software development life cycle (SDLC) each defect was detected; (b) which orthogonal defect class each defect most likely belongs to, based on expert interpretation; and (c) how the software function was impacted. An orthogonal defect class identifies which major category a defect belongs to, based on common defect characteristics, and is used by companies such as IBM [16] to conduct defect trend analysis. In this work, it permitted the development of a correlation between defect class and UCA class.

One additional dataset—namely, OpenPilot (https://github.com/commaai/openpilot)—was used to validate the trained language model (Box F). OpenPilot is a popular open-source vehicle autopilot software that automatically drives a vehicle by relying on the video imagining collected from a dashboard camera. Variation in software function across the three datasets is desirable, as one of our fundamental assumptions is that the developed defect-correlation table is function-agnostic. The dataset from OpenPilot was obtained via the Github Issue Crawler Tool (https://github.com/ibrr1/GitHub-Issues-Crawler-Tool), an open-source web crawler that automatically collects data by traversing issues from any Github repository, then processing them into text files. Each text file contains developer labels that identify the type of defect being reported (e.g., bug, enhancement, cleanup, docs, and controls). Unlike the defect reports from the NoSQL datasets, the collected OpenPilot defect reports do not contain orthogonal defect class labels and must be manually labeled. ODC applied to OpenPilot defect reports follows the work presented in [16] and [17]. Of the 2,274 closed issue reports, 107 contain the label "bug" that forms the testing dataset. Only bug-labeled reports are used, due to their clear relevance to software functions. Table 2 presents the distribution of orthogonal defect class per dataset. Column 2 shows the total number of defect reports per dataset, columns 3 through 7 show the orthogonal defect classes per dataset, and column 8 shows the number of defects that were either not classified or determined irrelevant.

Table 2. Distribution of orthogonal defect classes per dataset.

Dataset	# of Reports	Algorithm	Assignment	Function	Check	Timing	Other
Hbase	1383	855	51	193	110	11	163
MongoDB	1618	864	69	226	177	27	255
Cassandra	1095	645	49	124	120	4	153

OpenPilot	107	63	4	14	11	2	13

Technically, the raw defect reports may be used to train the language model, but will result in a poor topic classification, as overall word choice in the reports varies significantly. Stemming and lemmatization (Process AD) is a document preprocessing methodology that trims words to their roots by removing pluralization, tenses, and suffixes. For example, one common stemming conducted in the reports is the conversion of "correctly" to "correct." In this work, the Porter stemming algorithm is used [18]. Stemming and lemmatization does not fundamentally change the content or meaning of the defect report and is intended to improve model performance. This process is critical for relevant model development as developers may also use wordings that are synonymously equivalent but semantically different to describe what was wrong with the software.

After stemming and lemmatization, the data is fed into the Lbl2Vec model (Process DE) as a list of text files. White space characters (e.g., spaces, newline characters) are not stripped from the text files. All NoSQL defect reports are used in training the language model, but only the OpenPilot defect reports are used to test and evaluate its efficacy.

2.1.2 Lbl2Vec Language Model

Lbl2Vec [19] (Box E) is an unsupervised document clustering technique that works by creating and relating jointly embedded word, document, and label vectors to user-specified topics and keywords. Documents that share sufficient sematic similarity through embedded vectors are thus classified by topic. Since all learned embeddings share the same feature space, their distance can be considered to represent their semantic similarity. To learn a label embedding, all document embeddings that are close to the descriptive keyword embedding of a topic are clustered together to compute a topic centroid (Process EF of Figure 3) [19]. The distance from a topic centroid determines the similarity metric for a document, ranging from 0 to 1. Lbl2Vec uses the cosine similarity metric to determine how similar a document is to a topic centroid [19]. This method of automated document topic classification reduces the need for annotation costs and can expediate document processing [19]. Lbl2Vec is optimal for our use case, as defect reports are information-dense but require significant expert knowledge and time to interpret. In addition, each defect report describes a unique software defect that is laden with contextual and environmental information about the system.

The output of Lbl2Vec is an array of similarity metrics for each user-defined topic ranging from 0 to 1, where 0 represents no similarity, and 1 represents perfect similarity (Process FB of Figure 3). Table 3 shows a sample of the Lbl2Vec-generated similarity metric per topic. Column 1 reports the defect report number; column 2 shows the most likely topic as deduced by Lbl2Vec; and columns 3–6 show the similarity metric for each user-defined topic. Lbl2Vec decides the most likely topic by taking the maximum similarity across all topics. As such, certain outcomes that we have considered may be possible. First, for report 1 in Table 3, this is the optimal outcome where (a) the similarity difference between topics is significant enough that the model has not confused the document type with other topics, and (b) the confidence in a single topic category is high. For report 2 in Table 3, the similarity metrics represent another observed outcome. In this case, the model has high confidence in two conflicting topics (i.e., UCA-A and UCA-B). The choice for UCA-B was chosen by the model, as it has the highest value; however, we consider this result inadequate because the difference to the nearest topic is too small. For the last case, report 3, the topic classifications are all low, even though UCA-C seems to be the most likely topic and has a higher similarity metric than all other topics. This result is also unacceptable; although UCA-C has achieved a higher similarity metric than other topics, its similarity to the topic itself is inadequate. Therefore, in this work, a topic is considered of that class if it (a) has a similarity metric greater than 0.5, and (b) the difference between the selected topic and the nearest topic is greater than 0.25. The value 0.5 was selected as it indicates that the report is more likely than not to be part of that topic class. A difference of 0.25 was selected as it represents the coincidence probability that any class is selected and suggests a sufficient difference with other topics. The entire process for UCA topic classification by Lbl2Vec is as follows:

- Determine the topic similarity metric for all defect reports in the dataset.
- Eliminate all defect report classifications that have similarity metrics in all classes lower than 0.5.
- Eliminate all defect report classifications that have a difference in similarity lower than 0.25.
- Use the remaining defect report classifications for model evaluation.

Table 3. Sample similarity metrics derived from Lbl2Vec on NoSQL defect reports.

Report #	Likely Topic	UCA-A	UCA-B	UCA-C	UCA-D
1	UCA-A	0.853	0.263	0.216	0.332
2	UCA-B	0.563	0.610	0.362	0.253
3	UCA-C	0.032	0.015	0.361	0.086

2.1.3 STPA Classification of Orthogonal Defects and Keyword Extraction

While knowledgeable experts may be able to classify defects into UCA classes, this process is both time consuming and inefficient. Lbl2Vec automates the classification of defect reports into the four UCA classes most closely resembled. The following four UCA classes [20] are the target topic groups:

- UCA-A: Not providing a control action, leading to a hazard
- UCA-B: Providing a control action, leading to a hazard
- UCA-C: Providing a control action that is too early, too late, or in the wrong order
- UCA-D: Providing a control action that lasts too long or is stopped too soon.

For each UCA topic class, keyword selection is critical for correct clustering of defect reports within the model. Keyword selection is based on the word frequency discovered for each defect report. First, from the NoSQL defect database, 100 randomly selected reports were chosen. These were then reviewed by two separate researchers (referred to as R1 and R2) familiar with system-theoretic process analysis (STPA) and systems analysis (Process AB). Each researcher independently selected a UCA class they believed the defect had caused. UCA class selection is based on how the developer described the software failure and how it had impacted users, functional, or non-functional requirements. Of the 100 selected reports, 85 reports had concurrent UCA selections between the two researchers (Box B). We used Cohen's Kappa (k) to analyze inter-rater agreement [21]. Cohen's Kappa measures the agreement between two raters that classify items in mutually exclusive categories and is defined in Equation (1). Here, p_0 is the relative observed agreement between raters and p_c is the probability of agreement by chance. Cohen's Kappa may be interpreted as *poor* when $k < 0$, *slight* when $0 \leq k < 0.2$, *fair* when $0.2 \leq k < 0.4$, *moderate* when $0.4 \leq k < 0.6$, *substantial* when $0.6 \leq k < 0.8$, and *almost perfect* when $0.8 \leq k < 1$ [21]. A negative Kappa implies disagreement between classifications. Cohen's Kappa will also be used to interpret the efficacy of the Lbl2Vec language model at UCA topic classification. Table 4 presents the UCA classification results made by R1 and R2. The determined Kappa for each dataset is provided in column 2, while columns 3 through 6 show the number of matching UCA classifications made by both researchers. Column 7 shows the total number of matching classifications, whereas column 8 presents the number of defect reports from each dataset. All the Kappa values are *substantial*, further suggesting a consistent correlation between defects and UCA class exists.

$$k = \frac{p_0 - p_c}{1 - p_c} \tag{1}$$

Table 4. Distribution of defect report classifications per dataset.

Dataset Name	Kappa	UCA-A	UCA-B	UCA-C	UCA-D	Matching	Sample Size

Hbase	0.79	10	19	1	3	33	39
Cassandra	0.85	6	14	3	1	24	27
MongoDB	0.76	10	7	10	1	28	34
Total	0.80	26	40	14	5	85	100

From the matching UCA classifications of defect reports, we analyzed the word frequencies used by developers to extract keywords for Lbl2Vec (Process BC). Conjunctions, prepositions, and words shorter than two letters were filtered from the frequency analysis. Figure 4 shows a histogram of word frequencies per UCA class. Words that appeared only once were trimmed from the histogram, as they were deemed unlikely to be keywords. Words that appeared in multiple classifications were assigned as keywords to the class that had the highest occurrence frequency. "Nullpointexception" was assigned to UCA-B as a keyword, since it appears more frequently than for UCA-A. Ten words based on frequency of occurrence from each histogram were selected as UCA topic keywords (Process CE of Figure 3) and are shown in Table 5 (Box C).

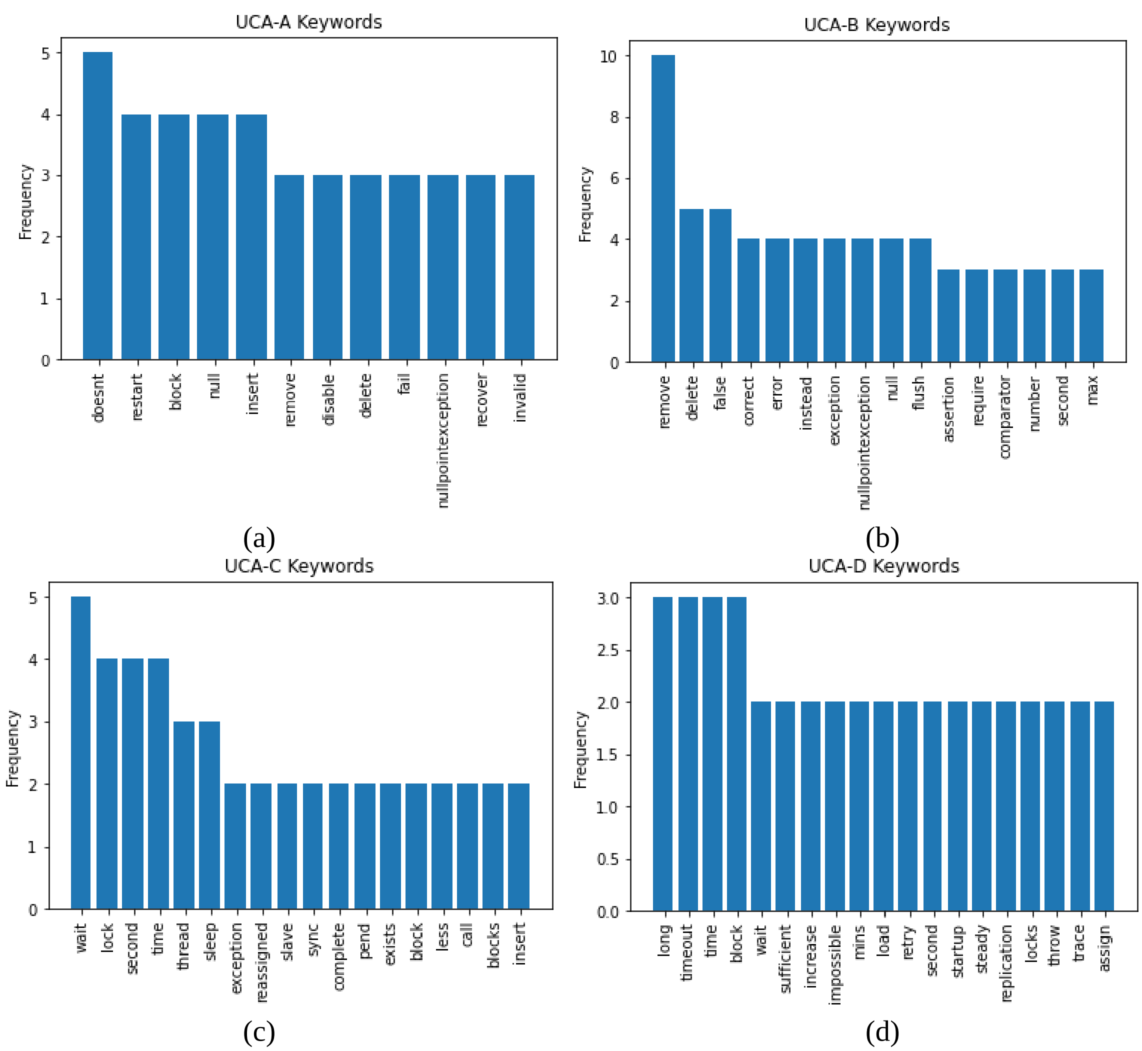


Figure 4. Defect description word usage frequency histograms pertaining to keyword extraction for the (a) UCA-A, (b) UCA-B, (c) UCA-C, and (d) UCA-D labels.

Table 5. UCA class topical keywords selected for Lbl2Vec model development.

Topic	Keywords
UCA-A	doesn't, restart, block, null, insert, disable, fail, recover, invalid, miss
UCA-B	remove, delete, false, correct, error, instead, exception, nullpointexception, flush, assertion
UCA-C	wait, lock, second, time, thread, sleep, exception, reassigned, slave, sync
UCA-D	long, timeout, block, sufficient, increase, impossible, load, retry, replication, trace

2.2 Results

2.2.1 Evaluation of Model Performance on Training and Testing Datasets

The trained Lbl2Vec was first applied to the NoSQL datasets presented in Table 6 to calculate a similarity metric for each defect report to the UCA topics. Table 6 compares the document topic classification to the classifications derived by researchers R1 and R2 with the Lbl2Vec classifications. Column 2 presents the Kappa value between the researchers and the Lbl2Vec. Columns 3–6 show the Lbl2Vec UCA classifications. Column 7 shows the number of matchings classifications made by Lbl2Vec to the total number of matching classifications made by researchers R1 and R2. The last row of the table shows the overall Kappa and the overall number of matching UCA classifications to the classifications made by the researchers. For instance, column 3, row 5, shows that 19 of the 26 matching UCA-A classifications made by the researchers were made by Lbl2Vec as well. While the Kappa value is lower than those made by the researchers, an overall Kappa value of 0.702 still implies *substantial* agreement, suggesting the model is somewhat capable of correctly determining UCA classes that are concurrent with human interpretation.

Table 6. Lbl2Vec UCA topic classifications and Kappa values.

Dataset Name	Kappa (*k*)	UCA-A	UCA-B	UCA-C	UCA-D	Matching/Total
Hbase	0.636	7	15	1	1	24/33
Cassandra	0.667	4	11	3	0	18/24
MongoDB	0.809	8	6	10	0	24/28
Total	0.702	19/26	32/40	14/14	1/5	66/85

In applying Lbl2Vec to the entire NoSQL dataset, we derived a UCA classification for all defect reports. Of the 4,096 defect reports, 570 were determined to be inapplicable as (a) the topic similarity metric was less than 0.5, or (b) the difference to the next closest class was less than 0.25. The remaining 3,526 classifications were split as follows: 1,673 defect reports were labeled UCA-A, 1,253 were labeled UCA-B, 419 were labeled UCA-C, and 181 were labeled UCA-D. For each UCA classification, we examined the frequency at which they were also labeled as a specific orthogonal defect class. In the case of the 1673 UCA-A labeled defect reports:

- 739 belonged to the algorithm defect class.
- 255 belonged to the assignment defect class.
- 800 belonged to the checking defect class.
- 562 belonged to the function defect class.
- 95 belonged to the timing defect class.

These values were then used to calculate the correlation between a UCA-A type failure given an algorithm defect exists in the software. This was applied to all orthogonal defect classes for all UCA classes. The final correlations are shown in Table 7 as the defect-UCA correlation table. A comparison is made between the classifications manually derived by the researchers and the Lbl2Vec classified UCA correlation table. For each UCA column, the two values are placed side by side. Of the 20 defect-to-UCA correlations, nine Lbl2Vec-derived correlations matched the researcher-derived correlations within a margin of ± 0.05, five were greater, and six were lower. While not all the values matched, this does not imply that Lbl2Vec is incapable and inaccurate. Rather, as the dataset used by Lbl2Vec to derive the correlations was larger than those used by the researchers, it simply implies that the Lbl2Vec correlations are more data-informed.

Table 7. Defect-UCA correlation table derived from manual vs. Lbl2Vec classification.

Orthogonal Defect Class $F(\mathbf{defect})$	**UCA Class Correlation, $P(UCA_X \mid \mathbf{defect})$**							
	UCA-A		**UCA-B**		**UCA-C**		**UCA-D**	
	Manual	Lbl2Vec	Manual	Lbl2Vec	Manual	Lbl2Vec	Manual	Lbl2Vec
Algorithm	0.331	0.442	0.137	0.098	0.339	0.321	0.194	0.138
Assignment	0.284	0.153	0.648	0.504	0.057	0.101	0.011	0.039
Checking	0.305	0.478	0.262	0.338	0.241	0.139	0.191	0.043
Function	0.385	0.336	0.240	0.423	0.240	0.109	0.135	0.132
Timing	0.189	0.057	0.054	0.083	0.622	0.759	0.135	0.101

Table 7 was analyzed as follows for UCA-A probabilities. First, given the occurrence probability of a specific defect class (e.g., algorithm) remaining in the software after the SDLC activities, the probability of a UCA-A is the matrix product between the defect class occurrence probability and the defect-UCA-A correlation. The full equation for deriving UCA-A event probabilities is given as Equation (2):

$$P(\mathrm{UCA_A}) = \begin{bmatrix} P(\mathrm{UCA}_A|\mathrm{Alg.}) \\ P(\mathrm{UCA}_A|\mathrm{Asi.}) \\ P(\mathrm{UCA}_A|\mathrm{Fnc.}) \\ P(\mathrm{UCA}_A|\mathrm{Chk.}) \\ P(\mathrm{UCA}_A|\mathrm{Tim.}) \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} F(Alg.) \\ F(Asi.) \\ F(Fnc.) \\ F(Chk.) \\ F(Tim.) \end{bmatrix} \quad (2)$$

Lbl2Vec was also applied to the OpenPilot dataset to validate that the correlations were consistent across different software functions. Note that, as the OpenPilot dataset only contains 107 defect reports, the results will have variations in a similar manner to the manually derived correlation table. In Table 8, the correlations derived for all three approaches are shown, with "Man" representing manual classifications by the researchers, "NoSQL" representing Lbl2Vec on the NoSQL datasets, and "OP" representing Lbl2Vec on the OpenPilot dataset. For the OpenPilot correlation columns, many of the correlations are zero, as Lbl2Vec did not classify any defects into those UCA classes. This does not imply that Lbl2Vec is incapable of UCA class identification, but only that the OpenPilot dataset may have been too small for those classes.

Table 8. Defect-UCA correlation table derived via manual interpretation, Lbl2Vec classification on the NoSQL datasets, and Lbl2Vec on the OpenPilot dataset.

Defect	**UCA Class Correlation**

Class	UCA-A			UCA-B			UCA-C			UCA-D		
	Man	NoSQL	OP	Man	NoSQL	OP	Man	NoSQL	OP	Man	NoSQL	OP
Alg.	0.33	0.44	0.37	0.14	0.10	0.05	0.34	0.32	0.3	0.19	0.14	0.28
Assi.	0.28	0.15	0.0	0.65	0.50	1.0	0.06	0.10	0.0	0.01	0.04	0.0
Check.	0.30	0.48	0.50	0.26	0.34	0.29	0.24	0.14	0.21	0.19	0.04	0.0
Funct.	0.38	0.34	0.51	0.24	0.42	0.14	0.24	0.11	0.0	0.14	0.13	0.07
Tim.	0.19	0.06	0.0	0.05	0.08	0.0	0.62	0.76	0.0	0.14	0.10	0.0

The 94 orthogonal-defect-labeled reports were manually classified by R1 into UCA classes and compared against the Lbl2Vec classifications. Of the 94 reports, only 86 could be clearly manually classified into a UCA class. The other eight lacked adequate descriptive information to sufficiently derive a UCA class. Lbl2Vec matching and mismatching classifications can be seen in Table 9. In columns 2–6, the matching to total manual UCA classification is shown. For instance, for column 2, only 26 of the 32 UCA-A classifications made by Lbl2Vec matched the researcher's classifications. Ultimately, the derived Kappa value suggests that Lbl2Vec is only *moderately* capable of correctly determining certain UCA classes—namely, UCA-A and UCA-B. The Kappa values for UCA-C and UCA-D suggest *fair* and *slight*, respectively, and are most likely attributable to the small sample size.

Table 9. Kappa values for Lbl2Vec on OpenPilot with number of classifications per UCA.

	UCA-A	UCA-B	UCA-C	UCA-D	Match/Total
Lbl2Vec/Manual	26/32	32/42	3/5	2/7	63/86
Kappa Value	0.75	0.68	0.47	0.05	0.64

2.3 Challenges and Opportunities

The research presented suggests that an alternative approach to software failure quantification may exist through root cause analysis correlations to specific UCA events. This would greatly improve upon the current approach of software basic event quantification, which uses the International Electrotechnical Commission (IEC) 61508 [22] safety integrity level (SIL) 4 bounding estimates [23]. While SIL 4 bounding estimates are currently agreed to be the necessary reliability for safety-critical digital systems, they also produce overly conservative estimates of failure probability in fault tree analysis (FTA) of software-based digital I&C systems in nuclear power plants [23]. By analyzing detected defects and their associated orthogonal defect classes throughout the SDLC, the occurrence probability of UCA events can be estimated. Application of Lbl2Vec for software defect mining also significantly reduces the amount of human effort required to develop the defect-UCA correlation table. By assessing an even larger dataset of software defects (such as in https://www.kaggle.com/datasets/davidshinn/github-issues), a global correlation table may be derived to assess the reliability of any software system regardless of function or usage.

However, there are several challenges that need to be addressed first. In terms of model development, parameter optimization unsurprisingly significantly impacts the final UCA topic classification. Parameters that require further examination include, but are not limited to, the number and choice of keywords. Although our selection of keywords was based on occurrence frequency within labeled reports, it is unknown whether technical jargon would result in better topic classification. Another challenge in this approach is developing consistent labeling of UCAs for keyword extraction. While R1 and R2 showed good Kappa agreement, this does not imply that all defect-UCA classifications were correct or consistent. For instance, for certain defect reports, it could not be agreed upon whether UCA-C or UCA-D was a better fit. Furthermore, while Lbl2Vec is an unsupervised methodology for topic classification, the choice of which defect reports to include in the model training impacts which word embeddings are associated with keywords, and may vary the final topic centroid, ultimately impacting UCA topic classification. Therefore, while the use of NLP (i.e., Lbl2Vec) is promising, a consistent methodology for identifying the correct and relevant training data is still needed. The phrase "garbage in, garbage out" is more relevant than ever for proper model development.

2.4 Summary of Work

FTA and event quantification play a crucial role in the licensing and modernization of existing and proposed nuclear digital I&C systems. Thus, this work proposes a method to automate root cause analyses of software defects in order to assist in quantifying UCAs that may be integrated in fault trees. The results suggest that a consistent correlation between software defects and UCAs exists, such that an alternative quantification method may be adopted in lieu of existing bounding estimate approaches. Future work will focus on comparing the results from other NLP models applied for root cause analysis to the results derived from Lbl2Vec.

3. INTER-SYSTEM COMMON CAUSE FAILURES

3.1 Motivation

DI&C systems are complex and may contain multiple functions and subsystems that may share resources, communication channels, power supplies, code, and displays. This complexity allows for a single digital system to perform multiple tasks. However, software can exhibit unpredictable failure modes and malfunctions that can affect other systems and potentially cause cascading effects. The interconnected nature of software means that errors can propagate through multiple parts of the system, making it difficult to pinpoint the exact cause and consequences of errors. Despite efforts to introduce diversity, the complex interactions within software and between systems can result in unexpected CCFs. Furthermore, since multiple digital systems can utilize similar algorithms, a single failure source can affect multiple functions or systems.

One consideration pertaining to software failure is functionality. Claes et al. [24] characterized software failure as a condition that manifests under specific conditions during the execution of a program. That is, the cause of the failure is not only a fault or defect that exists statically within the program, but also the certain conditions that occur. Thus, Claes et al. presented the relationship between terms, saying that an error during development results in a fault, whereas a fault causes a failure during execution. In addition, software defects may degrade a system without necessarily causing a failure. For example, in the case of an application with several functions, if one of the functions that necessitates access to a remote service is unable to establish a connection to the remote service, the function will cease. However, other functions of the application may continue to operate as expected. This helps demonstrate how the defects that influenced remote service connection did not impact other applications (i.e., some parts of the system were degraded, though others were not). Thus, software failures occur in combinations with a specific condition (e.g. input), emphasizing the importance of the functional requirements that activate them as well as the fault itself, see Figure 5.

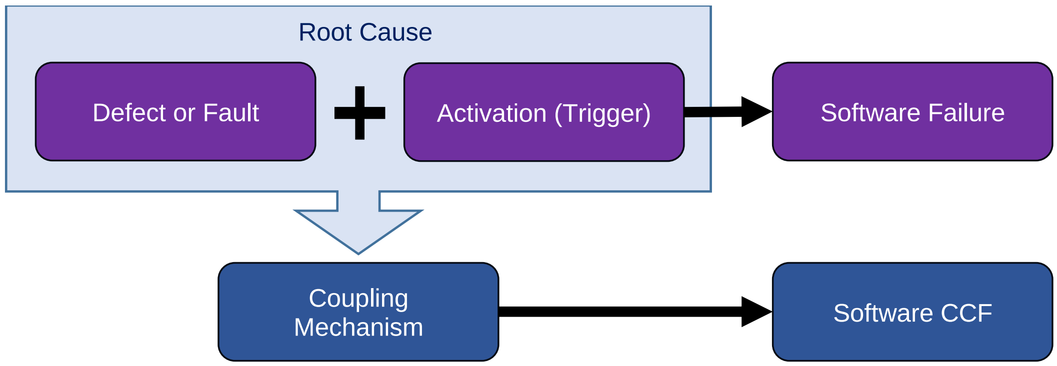


Figure 5. Definition of software failure.

Basu [25] noted that faults in a digital system can be propagated throughout the sub-systems, because a single subprogram/subroutine can be used multiple times throughout the program. Additionally, if there are redundant systems, the same defects might be repeated. Tero [26] mentioned that, in certain cases, CCFs between software from different vendors have been modeled, owing to the potential similarities that might exist in the software from these different suppliers. In 1990, mathematical programs such as MACSYMA, Maple, Mathematica, and Reduce—developed by different companies using different programming languages—produced the same incorrect results [8]. This highlights the possibility that two programs that seem different may exhibit the same incorrect behavior. In other words, CCFs can occur in software despite different manufacturers, programmers, programming languages, and algorithms. The Nuclear Regulatory Commission (NRC) suggests that the intra-system CCF be included if the same or similar software is used across multiple divisions or subsystems within a digital system, and that the inter-system CCF be included if the software is used across different systems. The inter-system CCF can include common support software that provides fundamental functions, such as communication with hardware, software platforms, controllers, and operating systems for CPUs [27].

The present research introduces an updated concept to address coupling mechanisms as part of the CCF analysis discussed in a previous study [5] that involved a digital reactor trip system (RTS) and a digital engineered safety features actuation system (ESFAS). (ESFAS is hereafter referred to as ESF in this report for the sake of convenience.) The RTS provides immediate shutdown of the reactor, either automatically or manually, whereas ESF provides mitigation systems (e.g., the core cooling and containment systems) to prevent and mitigate accidents. In a previous work, commonality was identified between the RTS and ESF, as these two systems each perform their respective functions by using two of the four processors in the same local coincidence logic rack. The previous study deemed identification of coupling mechanisms to be a challenging aspect of assessing inter-system CCFs. This research aims to improve on the existing CCF methodology.

3.2 The Current CCF Modeling Approach

The hybrid CCF modeling approach in the LWRS-developed framework was introduced in previous works [5] [6]. The hybrid CCF model was developed to support limited data and early development conditions—specifically by relying on qualitative features of the design to inform how well-defended a system is against CCF. The hybrid CCF model is parametric and is based on the beta factor model [28]. Like the beta factor model, the hybrid model treats CCFs as the complete failure of all components within a common cause component group (CCCG). This fits the theoretical basis for software CCFs, which is that they occur when operational conditions (e.g., common inputs) activate common software defects within a CCCG. Because software will perform exactly how it has been coded, so long as there are common defects the entire CCCG will fail together and experience a CCF. Subtle differences in coupling mechanisms may influence which components can be grouped and may fail together. Thus, a critical aspect of the hybrid CCF model—and one of its primary steps—is identifying the CCCGs. The general equations of the hybrid CCF model are listed below, followed by the main steps for performing an evaluation using the model.

$$Q_{CCF} = Q_{CCF_1} + Q_{CCF_2} + \cdots Q_{CCF_N} \tag{3}$$

$$Q_{CCF_n} = \phi_n Q_{CC_n} \tag{4}$$

$$Q_I = Q_T - Q_{CCF} \tag{5}$$

$$Q_T = \sum_{1}^{N} \phi_n Q_{CC_n} + Q_I \tag{6}$$

where Q_{CCF} is the total CCF failure probability for a given component. Q_{CCF_n} is the total CCF of a for a given CCCG. Q_{CC_n}represents the theoretical CCF probability for the CCCG, based on the similarity between the components of the CCCG (e.g., identical software requirements). Defenses may exist that prevent the CCFs from dominating the failure space for the system and otherwise reduce the CCF of the CCCG. These defenses are accounted for by ϕ_n.

The main steps of the CCF model are as follows:

1. Identify CCCGs. The CCCGs are identified based on the group of components (or for this work, software elements, modules, etc.) that share attributes which allow the group to be susceptible to CCF.
2. Define CCF defense parameters for each CCCG. The defense parameter, ϕ_n, represents the degree to which the CCCG is defended against CCF. The defense parameters are assessed for each of the seven subfactors shown in Table 10. The score consists of six levels, ranging from A (indicating weak defense) to E (indicating strong defense). Guidance for scoring the subfactor scoring tables was given in previous research [4]. All seven of the subfactors in Table 10 are areas where defense against CCF can be accomplished. As future work will serve to refine this process, the details are not included in the current report.
3. Determine the CCCG total failure probability. The goal here is to provide a quantitative value for each Q_{CC_n}.
4. Calculate the CCF for each CCCG.

Table 10. Subfactor table for software.

Subfactors	**A**	**B**	**C**	**D**	**E**
Information/Input Similarity	23976	4265	759	135	24
Understanding	7992	1422	253	45	8
Analysis, Feedback, & Awareness	7992	1422	253	45	8
Human-Machine Interface	11988	2132	379	67	12
Safety Culture & Training	6993	1244	221	39	7
Control	4995	888	158	28	5
Testing	11988	2132	379	67	12
$\phi_n = \sum(Subfactors)/d$ Where the denominator, $d = 100000$.					

The focus of this section is on Step 1, identify the CCCGs. The previous milestone report recognized the CCCG identification to be a particularly challenging and unstructured approach. This work provides a more direct guidance for this effort.

3.3 Consideration of CCCGs

This study considers functional requirements as one of coupling factors for identifying potential software CCF. First, CCCG identification is seen to become more complicated when considering inter-system CCFs between software that perform similar functions in different systems. For example, if there is only one coupling factor, the total number of CCCGs is 1. However, with an additional coupling factor, the number of CCCGs can increase to 3 (i.e., one CCCG for coupling factor 1, a second CCCG for coupling factor 2, and a third CCCG that includes both coupling factor 1 and coupling factor 2). The consideration of additional coupling factors or further increases the CCCGs to be considered. The increase in CCCGs heightens the complexity of CCF analysis.

Updated concepts for the CCF modeling approach

In digital systems, CCFs can be classified into four types, according to the location of the failure source and the controller design [29] as shown in Figure 6. Type 1 is when the failure source exists in the controller shared by two or more SSCs. Type 2 is when the failure source exists in the external shared resource of the controller. Type 3 is when each controller uses one shared resource, and the failure source exists in that resource. Type 4 is when a common or similar failure source exists in each controller. Classifying CCF types is important for understanding the types of coupling factors of each system.

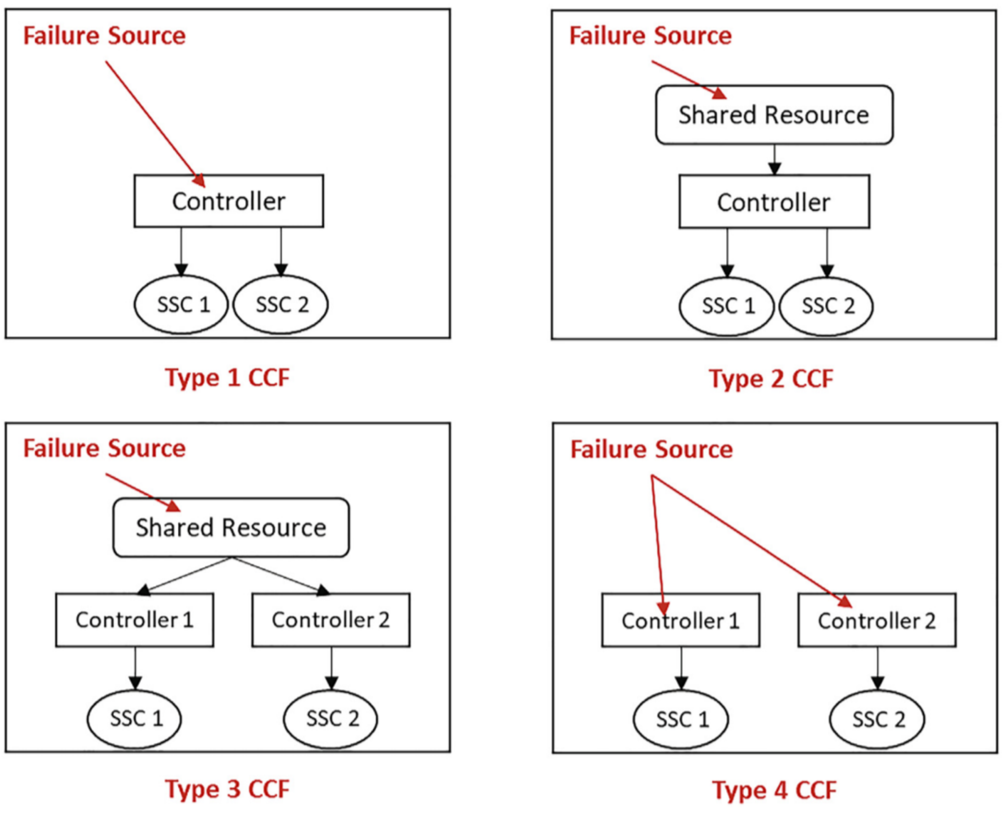


Figure 6. Types of software CCFs [29].

Categorizing the coupling factor type enables the number of possible CCCGs to be identified. For this work, common input, common software design, and common functional requirements are considered the coupling factor categories. For common input, an example of a common fault signal from a shared resource can lead to a Type 2 or Type 3 CCF. For common software design, an example of a common software defect as a failure source existing in multiple controllers can be given, potentially leading to a

Type 1 or Type 4 CCF. For common functional requirements, an example of a failure source existing in multiple controllers can be given, potentially leading to a Type 4 CCF.

If the number of coupling factor categories is known and the number of coupling factors within each category is confirmed, the number of CCCGs can be derived per Equation (7), where m is the number of coupling factor categories and N_i is the number of coupling factors for the i-th coupling factor category:

$$\begin{aligned} \textit{The number of CCCGs} &= \left(\sum_{j=1}^{N_1}\binom{N_1}{j}\right) \times \left(\sum_{j=1}^{N_2}\binom{N_2}{j}\right) \times \left(\sum_{j=1}^{N_3}\binom{N_3}{j}\right) \times \dots \times \left(\sum_{j=1}^{N_m}\binom{N_m}{j}\right) \\ &= \prod_{i=1}^{M}\left(\sum_{j=1}^{N_i}\binom{N_i}{j}\right) \end{aligned} \tag{7}$$

For example, if there is only one common input and two coupling factors for common functional requirements and common design categories, then m = 3, N_1 = 1, N_2 = 2, and N_3 = 2. In this case, the (nine) possible CCCGs are as shown in Table 11.

Table 11. Possible CCCGs for the case involving a common input, two common functional requirements, and two common designs.

CCCG	**Components**
CCCG 1	The components with same input (I)
CCCG 2	The components with same input (I), same design (D1)
CCCG 3	The components with same input (I), same design (D2)
CCCG 4	The components with same input (I), same functional requirement (F1)
CCCG 5	The components with same input (I), same functional requirement (F1), same design (D1)
CCCG 6	The components with same input (I), same functional requirement (F1), same design (D2)
CCCG 7	The components with same input (I), same functional requirement (F2)
CCCG 8	The components with same input (I), same functional requirement (F2), same design (D1)
CCCG 9	The components with same input (I), same functional requirement (F2), same design (D2)

Meanwhile, the number of basic events, including the CCF for a specific single component, is obtained using Equation (8). For example, for components with the same input (I), same functional requirement (F1), and same design (D1), five basic events exist, and they can be modeled as shown in Figure 7.

$$\begin{aligned}
&\textit{The number of basic events for a single component} \\
&= 1 \\
&+ \left(\sum_{j=1}^{N_1} \binom{N_1 - 1}{j - 1}\right) \times \left(\sum_{j=1}^{N_2} \binom{N_2 - 1}{j - 1}\right) \times \left(\sum_{j=1}^{N_3} \binom{N_3 - 1}{j - 1}\right) \times \dots \\
&\times \left(\sum_{j=1}^{N_m} \binom{N_m - 1}{j - 1}\right) = \prod_{i=1}^{M} \left(\sum_{j=1}^{N_i} \binom{N_i - 1}{j - 1}\right)
\end{aligned} \tag{8}$$

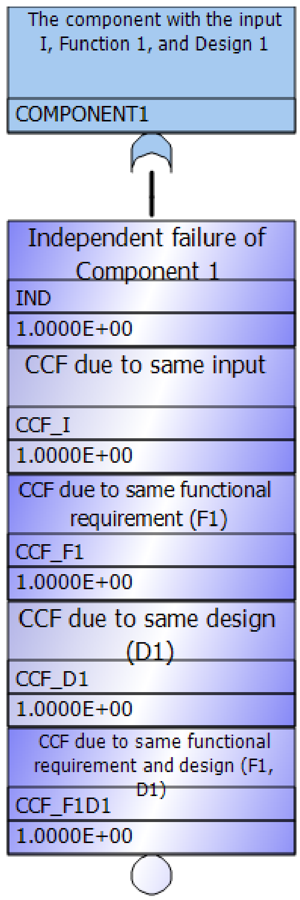


Figure 7. Modeling of a single component including CCF.

The existing equations of the hybrid CCF model should be clarified. The theoretical failure probability was introduced; however, the previously provided equations require additional clarification. This is given below to more clearly show the independent and CCF portions. Thus, when referring to theoretical CCF probability, previous work implied that there is no independent contributor to worry about. This is only true when ϕ_n is equal to one. Otherwise, defenses against CCF will indicate that the Q_{CCF_i} is not equal to Q_{CC_i}, consequently resulting in some additional independent failures that should be considered. These clarifications are expressed in the following equations:

$$Q_{cc_i} = Q_{ind_i} + Q_{CCF_i} \tag{9}$$

$$Q_{ind_i} = (1 - \phi_n)Q_{CC_i} \tag{10}$$

These clarifications also show that a more complete definition of Q_{cc_i} is given as the total probability of the i-th CCCG, which is composed of CCF probability (Q_{CCF_i}) and independent probability (Q_{ind_i}). This is in contrast to previous work that gave Q_{CC_i} as the theoretical failure probability; this is simply the limiting condition of its definition (i.e., only true when $\phi_n = 1$). Mathematically, ϕ_n represents the fractional proportion of CCF to the total probability of each CCCG, as defined in Equation (11). Conceptually, ϕ_n is defined in the hybrid model by how well-defended the CCCG is against CCF (1 equals no defense, 0 equals completely defended):

$$\phi_n = \frac{Q_{CCF_n}}{Q_{cc_n}} \tag{11}$$

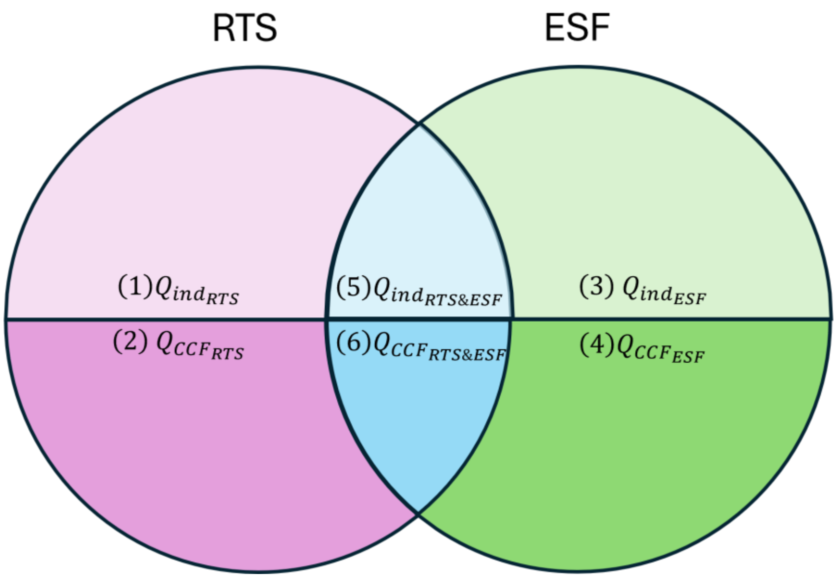


Figure 8. Example of CCCGs with two functions.

Consider another example: we have three coupling factor categories, each containing the following coupling factors: input(0), function(2), and design(2). These can be expressed in Figure 8, in which the pink part corresponds to a CCCG formed from RTS functional attributes of a group of components, the green part corresponds to a CCCG formed from ESF functional attributes of a group of components, and the blue part corresponds to a CCCG formed from both RTS and ESF functional attributes of a group of components. The total failure probability of each i-th CCCG, Q_{cc_i}, is defined as follows:

$$Q_{cc_1} = Q_{ind_{RTS}} + Q_{CCF_{RTS}} \tag{12}$$

$$Q_{cc_2} = Q_{ind_{ESF}} + Q_{CCF_{ESF}} \tag{13}$$

$$Q_{cc_3} = Q_{ind_{RTS\&ESF}} + Q_{CCF_{RTS\&ESF}} \tag{14}$$

where: $Q_{ind_{RTS}}$ is the individual failure probability for a component due to its RTS functional attributes only (meaning there is no aspect of the ESF functional attribute), $Q_{CCF_{RTS}}$ is the CCF probability involving the RTS functional attributes (meaning the failure event does not involve any aspect of the ESF functional attribute), $Q_{ind_{ESF}}$ is the individual failure probability for a component due to its ESF functional attributes (meaning there is no aspect of the RTS functional attribute involved with failure), $Q_{CCF_{ESF}}$ is the CCF probability involving the ESF functional attribute (meaning there are no RTS functional attributes involved), $Q_{ind_{RTS\&ESF}}$ is the individual failure probability of a component involving the RTS and ESF functional attributes (e.g., a component has shared aspects with RTS and ESF functions, however only a single component fails rather than entire CCCG), and $Q_{CCF_{RTS\&ESF}}$ is the CCF probability involving both the RTS and ESF functional attributes of the components

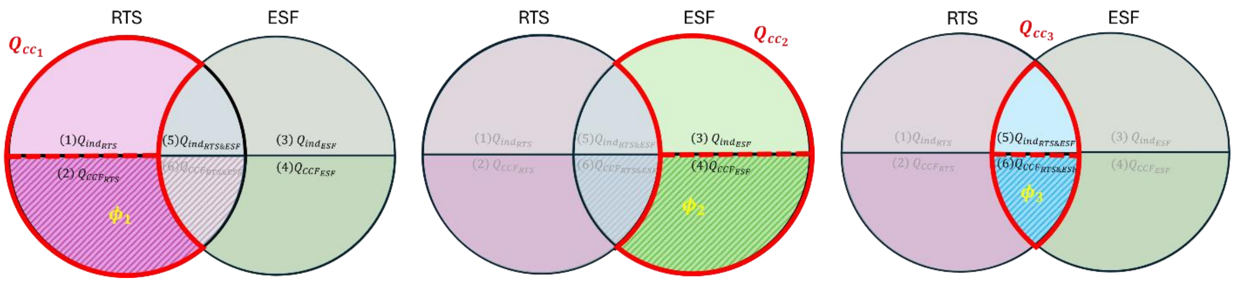


Figure 9. Example of CCCGs with highlighted sections to clarify parameters.

For each CCCG, ϕ_i is defined as follows:

$$\phi_1 = \frac{Q_{CCF_{RTS}}}{Q_{cc_1}} \tag{15}$$

$$\phi_2 = \frac{Q_{CCF_{ESF}}}{Q_{cc_2}} \tag{16}$$

$$\phi_3 = \frac{Q_{CCF_{RTS\&ESF}}}{Q_{cc_3}} \tag{17}$$

In this case, the total failure probability for RTS ($Q_{T_{RTS}}$) is calculated as the sum of the total independent failure probability and the total CCF probability.

$$Q_{T_{RTS}} = Q_{T_{I_{RTS}}} + Q_{T_{CCF_{RTS}}} \tag{18}$$

$$Q_{T_{I_{RTS}}} = Q_{ind_{RTS}} + Q_{ind_{RTS\&ESF}} \tag{19}$$

$$Q_{T_{CCF_{RTS}}} = Q_{CCF_{RTS}} + Q_{CCF_{RTS\&ESF}} \tag{20}$$

On the other hand, the total failure probability for a component that has the RTS function can also be expressed as the sum of Q_{cc_1} and Q_{cc_3}. Considering Equations (18), (19), and (20), the total independent failure probability for RTS can be organized as follows:

$$Q_{T_{I_{RTS}}} = Q_{T_{RTS}} - Q_{T_{CCF_{RTS}}} \tag{21}$$

$$= Q_{T_{RTS}} - \phi_1\left(Q_{T_{RTS}} - Q_{cc_3}\right) - \phi_3 Q_{cc_3}$$
$$= (1 - \phi_1)Q_{T_{RTS}} + (\phi_1 - \phi_3)Q_{cc_3}$$

In Equation (21), each probability is greater than or equal to 0, and ϕ has a value of between 0 and 1. The case in which the total independent failure probability for RTS ($Q_{T_{I_{RTS}}}$) has the minimum value is when ϕ_1 is less than ϕ_3. Assuming that ϕ_1 and ϕ_3 are 0 and 1, respectively, $Q_{T_{I_{RTS}}}$is determined via the following equation:

$$Q_{T_{I_{RTS}}} = Q_{T_{RTS}} - Q_{cc_3} \tag{22}$$

In this case, $Q_{T_{RTS}}$is always greater than Q_{cc_3}, so independent failure probability always has a positive value. Thus, negative individual probability cannot occur.

Another example is given below. Consider the following: the first coupling factor category is assumed to have one common input, the second category has two coupling factors (i.e., the RTS and ESF) as functional requirements, and the third category has software 1 (SW1) and software 2 (SW2) designs. In this case, there are nine possible CCCGs and five basic events for each single component (see Table 12).

Table 12. Definition of the failure probability and parameter of each CCCG for the example involving a common input, two common functional requirements, and two common designs.

<table>
<tr><th>CCCG</th><th colspan="2">Failure probability of each CCCF ($\boldsymbol{Q_{CC_i}}$)</th><th colspan="2">parameter for each CCCG</th></tr>
<tr><td rowspan="2">CCCG 1</td><td rowspan="2">Q_{CC_1}</td><td>$Q_{ind_{RTS\&ESF_{SW12}}}$</td><td rowspan="2">$\phi_1$</td><td rowspan="2">$\phi_1 = \frac{Q_{CCF_{RTS\&ESF_{SW12}}}}{Q_{CC_1}}$</td></tr>
<tr><td>$Q_{CCF_{RTS\&ESF_{SW12}}}$</td></tr>
<tr><td rowspan="2">CCCG 2</td><td rowspan="2">Q_{CC_2}</td><td>$Q_{ind_{RTS_{SW12}}}$</td><td rowspan="2">ϕ_2</td><td rowspan="2">$\phi_2 = \frac{Q_{CCF_{RTS_{SW12}}}}{Q_{CC_2}}$</td></tr>
<tr><td>$Q_{CCF_{RTS_{SW12}}}$</td></tr>
<tr><td rowspan="2">CCCG 3</td><td rowspan="2">Q_{CC_3}</td><td>$Q_{ind_{ESF_{SW12}}}$</td><td rowspan="2">ϕ_3</td><td rowspan="2">$\phi_3 = \frac{Q_{CCF_{ESF_{SW12}}}}{Q_{CC_3}}$</td></tr>
<tr><td>$Q_{CCF_{ESF_{SW12}}}$</td></tr>
<tr><td rowspan="2">CCCG 4</td><td rowspan="2">Q_{CC_4}</td><td>$Q_{ind_{RTS\&ESF_{SW1}}}$</td><td rowspan="2">ϕ_4</td><td rowspan="2">$\phi_4 = \frac{Q_{CCF_{RTS\&ESF_{SW1}}}}{Q_{CC_4}}$</td></tr>
<tr><td>$Q_{CCF_{RTS\&ESF_{SW1}}}$</td></tr>
<tr><td rowspan="2">CCCG 5</td><td rowspan="2">Q_{CC_5}</td><td>$Q_{ind_{RTS_{SW1}}}$</td><td rowspan="2">ϕ_5</td><td rowspan="2">$\phi_5 = \frac{Q_{CCF_{RTS_{SW1}}}}{Q_{CC_5}}$</td></tr>
<tr><td>$Q_{CCF_{RTS_{SW1}}}$</td></tr>
<tr><td rowspan="2">CCCG 6</td><td rowspan="2">Q_{CC_6}</td><td>$Q_{ind_{ESF_{SW1}}}$</td><td rowspan="2">ϕ_6</td><td rowspan="2">$\phi_6 = \frac{Q_{CCF_{ESF_{SW1}}}}{Q_{CC_6}}$</td></tr>
<tr><td>$Q_{CCF_{ESF_{SW1}}}$</td></tr>
<tr><td rowspan="2">CCCG 7</td><td rowspan="2">Q_{CC_7}</td><td>$Q_{ind_{RTS\&ESF_{SW2}}}$</td><td rowspan="2">ϕ_7</td><td rowspan="2">$\phi_7 = \frac{Q_{CCF_{RTS\&ESF_{SW2}}}}{Q_{CC_7}}$</td></tr>
<tr><td>$Q_{CCF_{RTS\&ESF_{SW2}}}$</td></tr>
</table>

CCCG 8	Q_{CC_8}	$Q_{ind_{RTS_{SW2}}}$ $Q_{CCF_{RTS_{SW2}}}$	ϕ_8	$\phi_8 = \frac{Q_{CCF_{RTS_{SW2}}}}{Q_{CC_8}}$
CCCG 9	Q_{CC_9}	$Q_{ind_{ESF_{SW2}}}$ $Q_{CCF_{ESF_{SW2}}}$	ϕ_9	$\phi_9 = \frac{Q_{CCF_{ESF_{SW2}}}}{Q_{CC_9}}$

Per the example indicated by Table 12, the total failure probability of RTS with SW 1 is calculated as:

$$\begin{aligned} Q_{T\,RTS_{SW1}} &= Q_{CC_1} + Q_{CC_2} + Q_{CC_4} + Q_{CC_5} \\ &= Q_{T_{I_{RTS_{SW1}}}} + \phi_1 Q_{CC_1} + \phi_2 Q_{CC_2} + \phi_4 Q_{CC_4} + \phi_5 Q_{CC_5} \end{aligned} \tag{23}$$

where $Q_{T_{I_{RTS_{SW1}}}}$ is the total independent failure probability of RTS with SW1, as expressed by Equation (24) which is confirmed to always have a positive value:

$$Q_{T_{I_{RTS_{SW1}}}} = (1-\phi_1)Q_{CC1} + (1-\phi_2)Q_{CC2} + (1-\phi_4)Q_{CC4} + (1-\phi_5)Q_{CC5} \tag{24}$$

As another example, take the case in which the input and functional requirements each have one coupling factor for their categories, whereas the design category has three (i.e., SW1, SW2, SW3). In this case, there are seven CCCGs, and five basic events can be derived for the single component.

Table 13. Definition of the failure probability and parameter of each CCCG for the example involving a common input, common functional requirements, and three common designs.

CCCG	**Failure probability of each CCCF (Q_{CC_i})**		**parameter for each CCCG**	
CCCG 1	Q_{CC_1}	$Q_{ind_{SW1}}$ $Q_{CCF_{SW1}}$	ϕ_1	$\phi_1 = \frac{Q_{CCF_{SW1}}}{Q_{CC_1}}$
CCCG 2	Q_{CC_2}	$Q_{ind_{SW2}}$ $Q_{CCF_{SW2}}$	ϕ_2	$\phi_2 = \frac{Q_{CCF_{SW2}}}{Q_{CC_2}}$
CCCG 3	Q_{CC_3}	$Q_{ind_{SW3}}$ $Q_{CCF_{SW3}}$	ϕ_3	$\phi_3 = \frac{Q_{CCF_{SW3}}}{Q_{CC_3}}$
CCCG 4	Q_{CC_4}	$Q_{ind_{SW12}}$ $Q_{CCF_{SW12}}$	ϕ_4	$\phi_4 = \frac{Q_{CCF_{SW12}}}{Q_{CC_4}}$
CCCG 5	Q_{CC_5}	$Q_{ind_{SW13}}$ $Q_{CCF_{SW13}}$	ϕ_5	$\phi_5 = \frac{Q_{CCF_{SW13}}}{Q_{CC_5}}$
CCCG 6	Q_{CC_6}	$Q_{ind_{SW23}}$ $Q_{CCF_{SW23}}$	ϕ_6	$\phi_6 = \frac{Q_{CCF_{SW23}}}{Q_{CC_6}}$
CCCG 7	Q_{CC_7}	$Q_{ind_{SW123}}$	ϕ_7	

		$Q_{CCF_{SW123}}$		$\phi_7 = \frac{Q_{CCF_{SW123}}}{Q_{CC_7}}$

For the example indicated by Table 13, the total failure probability of SW 1 is calculated as follows:

$$\begin{aligned} Q_{T_{SW1}} &= Q_{CC_1} + Q_{CC_4} + Q_{CC_5} + Q_{CC_7} \\ &= Q_{ind_{SW1}} + Q_{CCF_{SW1}} + Q_{ind_{SW12}} + Q_{CCF_{SW12}} + Q_{ind_{SW13}} + Q_{CCF_{SW13}} + Q_{ind_{SW123}} \\ &\qquad + Q_{CCF_{SW123}} \\ &= Q_{T_{I_{SW1}}} + Q_{CCF_{SW1}} + Q_{CCF_{SW12}} + Q_{CCF_{SW13}} + Q_{CCF_{SW123}} \\ &= Q_{T_{I_{SW1}}} + \phi_1 Q_{cc1} + \phi_4 Q_{cc4} + \phi_5 Q_{cc5} + \phi_7 Q_{cc7} \end{aligned} \tag{25}$$

where $Q_{T_{I_{SW1}}}$ is the total individual failure probability of SW1, and can be organized per the following equation, which is also confirmed to always have a positive value:

$$Q_{T_{I_{SW1}}} = (1-\phi_1)Q_{CC_1} + (1-\phi_4)Q_{CC_4} + (1-\phi_5)Q_{CC_5} + (1-\phi_7)Q_{CC_7} \tag{26}$$

3.4 Discussion

Per their characteristics, different software types can be considered in the same CCCG, depending on the same input or functional requirement. This coupling can result in numerous CCCGs. In previous research, as there was no standard, identifying software CCCGs could be challenging. However, in this research, more systematic confirmation is enabled by performing CCCG identification according to coupling factor category. Future studies could further discuss these coupling factor categories. Since the proposed method identifies all possible CCCGs that may occur for the given coupling factor categories, screening by expert judgment may also be required to rule out unlikely CCFs. In addition, future research will be conducted on how to quantify Q_T and Q_{cc_i}.

4. COLLABORATIVE EFFORTS WITH PWROG

This section briefly summarizes the FY 2024 collaboration efforts between INL and members of the Pressurized Water Reactor Owners Group (PWROG) under the LWRS program, as carried out to support a pilot risk evaluation of a safety-related DI&C system in use at a nuclear utility. The complete analysis and results have been documented in a proprietary report.

Overview and Motivation

In 2022, a collaboration was initiated between the PWROG and the LWRS to support the risk evaluation of a DI&C safety actuation system in use at a nuclear utility. The purpose of the work [13] was to demonstrate the methodologies developed under the LWRS program to assess software CCF on a pilot DI&C safety-related system, and to address research topics identified by the PWROG, including: (1) identification of software failures, (2) quantification of software failures (especially given the lack of significant software-related failures in the historical record), and (3) addressing conservative assumptions in software CCF modeling, given that modeling assumptions can lead to overestimations of CCFs that dominate probabilistic risk assessment models.

In the spring of 2023, the LWRS team provided a preliminary demonstration [13] of two risk assessment pathways for quantifying software failures: (1) the data-sufficient pathway, which employs software testing data and relies on ORCAS; and (2) the data-limited pathway, which employs details on software development practices and relies on BAHAMAS. The LWRS report [13] demonstrated our capabilities to provide quantitative estimates of software failure. A commonly referenced quantitative estimate for software failure probability on demand is 1E-4 [30] and can be considered conservative. The LWRS report [13] provided statistical evidence of software failure probability per demand in the vicinity of the common industry estimate. However, several conservative assumptions employed for the data-limited pathway offered incentive to re-evaluate software CCF values. The primary deliverable for this year was a report documenting the recalculation enabled by the latest updates and refinements to the methods developed by under the LWRS program. A secondary motivation behind this year's collaboration was to reduce the level of effort to quantify DI&C software failures.

Summary

The primary focus for FY 24 was updating the data-limited pathway of the LWRS risk assessment framework by updating BAHAMAS to better support software failure quantification. Our previous work employed a screening-type assumption with BAHAMAS that set the defect introduction probability equal to 1.0, rather than evaluating every software development task individually. This reduced the burden for the analyst and can be useful for point estimate evaluations of software failure probability but create increases in uncertainty. In fact, the previous version of BAHAMAS did not provide clear estimates of the uncertainty associated with the quantification. Over the course of FY24, the LWRS team has provided guidance, methods, and a new software tool for performing the BAHAMAS calculation. These updates account for uncertainties and provide a more rigorous quantitative evaluation by examining the SDLC documentation in order to identify specific developmental tasks and quantifying their influence on defect introduction. The identified SDLC tasks are used as inputs for BAHAMAS. Guidance and details on performing a quantitative analysis with BAHAMAS are given in Appendix B.

The updates to BAHAMAS from FY24 allow for uncertainty quantification and discard the conservative assumption used in the previous evaluation for the PWROG. The added details led to the prediction of a reduced probability of defect introduction over the initial conservative value. Subsequently, the new defect introduction probability, coupled with adjusted model parameters, resulted in a significant reduction to the predicted mean software CCF probability from the previous year.

BAHAMAS was specifically developed to provide preliminary reliability insights for systems under development. Though conservative estimates may be useful for early screening assessments, the rigorous results from BAHAMAS serve as a useful metric for software reliability and for the efficacy of a selected SDLC. The results from BAHAMAS align with the conventional expectation that software-based systems should be at least as reliable as the analog systems they are set to replace. However, since BAHAMAS results are obtained without real-world testing, they should be supplemented with additional analyses as more data is acquired.

Ultimately, this collaborative project provided the LWRS team with insights for improving and further developing the LWRS-developed framework to respond to the needs of the nuclear industry. This work represents an example of a successful partnership and of the research benefits achievable through industry collaboration. It is a goal of the LWRS team to continue such collaborations and provide science- and technology-based solutions and insights that support the long-term operations of the existing nuclear fleet.

5. COLLABORATIVE EFFORTS WITH GEH

This section briefly summarizes the collaboration between LWRS and GE-Hitachi Nuclear Energy (GEH) to support a risk evaluation of a safety-related DI&C system in use at a nuclear utility. The names and functions of components and subsystems have been masked to protect GEH's proprietary technology.

In October of FY 2024, LWRS was invited to work with GEH to conduct risk analyses of GEH's early-design nuclear power plant safety systems. Three phases were identified as part of this project. Note that only phase 1 was completed, as phase 2 and phase 3 are planned for following FYs. In phase 1, qualitative analysis of the safety system was performed; in phase 2, quantitative and consequence analyses will be conducted; and in phase 3, a safety assurance case will be formed to justify the reliability of the safety system. The intent of phase 1 is to apply RESHA, a method developed through the LWRS RISA program, for early design support and qualitative risk assessment of the safety actuation system of a nuclear power plant. Due to the early design stage of the planned system, it was postulated that RESHA could qualitatively identify design inadequacies by comparing and contrasting different system architecture versions. A key focus area of this work was to identify these design inadequacies, which include but are not limited to software independent and CCF events and other DI&C-related failures.

In this work, a qualitative design analysis was conducted on the GEH safety actuation system. Two different dated versions of the GEH system were provided to the LWRS team for analysis. The difference between them lies in the addition of safety components, referred in this document as voting devices (VDs), to the later design version. The RESHA methodology was used to develop the qualitative fault trees (FTs) to assess these dated versions and to identify software-related failures. In regard to the change in design, FT analysis reveled that the addition of VDs in all redundant divisions of the GEH system indeed reduces the likelihood of failure. This was determined by observing the chain of events within a cutset, and by comparing the number of cutsets calculated from the two versions. Specifically, it was found that in the newer design version, independent failures (whether software- or hardware-related) of redundant divisions no longer impeded the safety function of the other divisions.

In addition to design comparison, a CCF analysis was also conducted in which two different total software CCFs were identified in relation to the VDs of the system. These software CCFs were identified based on their common cause component groups, where members consisted of duplicated component software systems in the redundant configuration of the system architecture. A comparison between the RESHA-developed FT and the one provided by GEH ensured that the FTs were aligned in terms of objective. Failure probabilities from the GEH FT were also adopted in the RESHA FT to qualitatively compare how the top event probability changes. In essence, the structures of the two FTs were well matched, and the top event probability for the two FTs was calculated to be the same magnitude, confirming the organization of the FT to be correct. Two improvements were made in the RESHA FT in contrast to the GEH FT: (1) refinement of how software CCFs can cause stakeholder losses by specifying the mode of software failure (as opposed to generic CCF events), and (2) identification of two software CCFs (as opposed to the single CCF proposed in the GEH FT). However, though potential software CCFs were identified, no further recommendations for addressing/mitigating these CCFs are currently available, as the system is too early in its development life cycle to evaluate defenses against software CCFs.

Other considerations were also discovered—namely, how peripheral bypass unit (PBU) and peripheral interface unit (PIU) failures can lead to unintended failures of the system. For the PBU, it was revealed that simultaneous bypasses can lead to failure to actuate the safety system. Such failure is contingent on the PBU failing and can be mitigated by implementing channel checking within the unit. Furthermore, it was identified that human-driven failures of the PIU can increase the likelihood of spurious trip. Spurious trip was not a top event assessed in this report but was included for completeness. PIU failures can also be mitigated by ensuring channel checking among divisions. Note that these events were not explicitly modeled in the RESHA FT, as the scope of analysis did not cover peripheral systems. However, these events are postulated based on the implicit evidence provided to INL.

As the GEH nuclear power plant safety system design evolves and grows, it is anticipated that the RISA methods presented in this work will be able to continuously assist the development team and will provide greater insights as the system design progresses. The final project objective being the safety assurance case to justify the reliability of the GEH safety system.

6. GAI APPLICATION ON DI&C

This section details work on the application of generative artificial intelligence (GAI) to support risk assessment. Specifically, this section shows how GAI can aid in the area of hazard analysis, including diversity and defense in depth assessments.

6.1 Introduction

Recently, developments related to GAI have spurred a major breakthrough in the field of AI. These developments are rapidly being adapted to further accelerate developments in different science and engineering areas. GAI models learn patterns from their training data to create new content that resembles the training data but remains distinct and novel. A major driving factor behind the recent GAI-related developments is the transformer architecture introduced by Vaswani et al. [31] in 2017. With proper selection of embedding techniques, the transformer architecture can be used with different data modalities (e.g., text, video, and image). The attention mechanism is the core of the transformer architecture. It allows the model to weigh the importance of different parts of the data, in a sequence, relative to each other. By computing attention scores, the model can capture long-range dependencies and complex relationships within the input data. With appropriate scaling, model size, and embedding techniques, the transformer-based models have led to the development of unimodal large language models (LLMs) and multimodal LLMs with advanced reasoning capabilities. These models have vast parametric memory that encapsulates large volumes of data. Notably, the LLMs developed by big corporations (e.g., generative pre-trained transformers [GPTs], as developed by OpenAI [32], and Llama, as developed by Meta AI [33]) particularly stand out. The transformers provide a highly scalable architecture that can support efficient transfer learning capabilities. Due to their scalability, efficiency, and multi-modal nature, transformer-based architectures have become an ideal choice for foundation models (FMs) [34] [31].

Industries are also rapidly adapting GAI technology to improve efficiency and streamline their workflow. SymphonyAI recently announced a series of GAI-based Copilots (i.e., Plant Performance Copilot, Digital Manufacturing Copilot, and the Connected Worker Copilot) to assist frontline workers in a variety of tasks so as to improve operating speed and efficiency [35]. Vision transformers were successfully used to build FMs for weather and climate applications [36]. Given these capabilities, we expect an increasing use of GAI in nuclear engineering to support innovation in the design, analysis, licensing, construction, operation, and maintenance of nuclear energy systems.

The field of generative AI is evolving rapidly thanks to rapid improvements in capability and performance. We can develop a GAI model (LLM) from scratch, or we can use the pre-trained version of these models and adapt them to nuclear engineering applications through fine-tuning, retrieval-augmented generation (RAG) [37], and prompt engineering. The pre-trained FM can be either open-source or proprietary, offering different types of models, capabilities, and user control. Training from scratch can be a laborious process that requires several iterations, optimization, and computing resources (GPU, TPU, etc.) For example, training GPT-3 (175 billion parameters) required significant computational resources and time. Here are some specific details on OpenAI:

- Computing Power: Approximately 10,000 V100 GPUs
- Data: 570 GB of text data
- Storage: Multiple petabytes, primarily for intermediate and final model checkpoints
- Energy: Estimates suggest around 1,287 MWh of electricity
- Duration: Training GPT-3 took several weeks (reportedly around 1–2 months).

For a unimodal LLM, during the initial training phase, the model is trained on massive datasets of texts and codes. The pre-training is self-supervised (i.e., the LLM does not require labeled data). The model learns grammar rules, syntax, patterns, and facts from the NLP during this phase. In the subsequent training phase, human supervision and feedback further improve the LLM response. The pre-trained LLM can be a FM for different downstream applications and can be used directly if it has the appropriate domain knowledge. Otherwise, the LLM must be customized for use in specific domains. For training from scratch, we must also address proper infrastructure needs related to different layers (see Figure 10 for an overview).

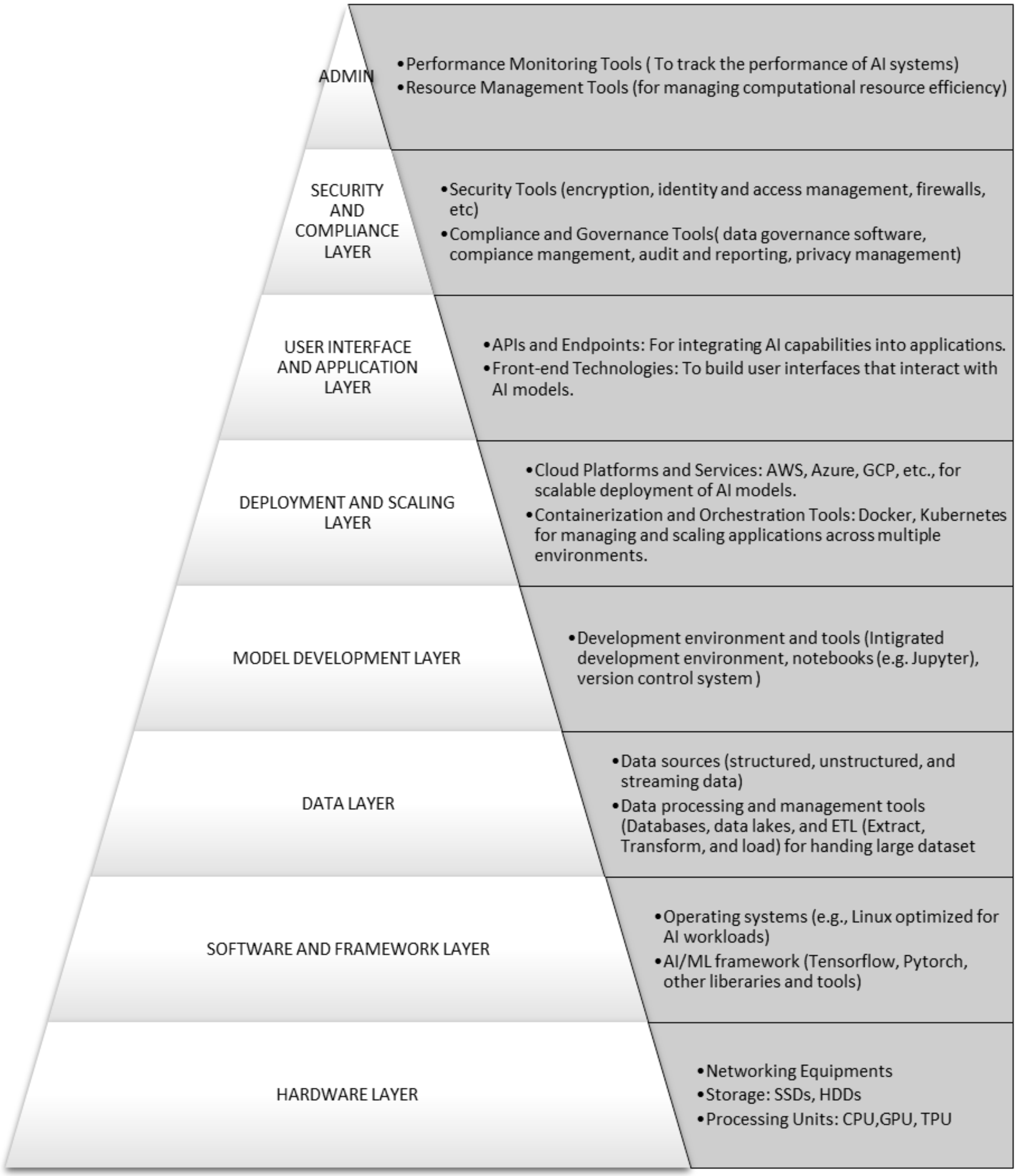


Figure 10. Overview of GAI infrastructure needs and requirements.

Fine-tuning involves adjusting the parameters of the pre-trained LLM so as to adapt it to the new data. Note that, during fine-tuning, only part of the model is available for weight adjustment. Usually the initial layers that capture general language patterns, features, rules, and facts are frozen to keep the base capabilities of the model intact. Pre-training involves self-supervised learning. However, since fine-tuning involves supervised learning, we need labeled data. Thus, data preparation is an important part of fine-tuning, which is an iterative process that requires careful optimization in consideration of training parameters and data quality and quantity.

RAG does not involve any sort of additional LLM training. Instead, it uses an external information retrieval system to augment the knowledge of the LLM with external sources of information. In this context, the pre-trained LLM is often considered a parametric memory (embedded within the LLM), while the external source is considered a nonparametric memory that LLM accesses via the information retrieval system.

The LLM provides a response based on the user's questions. The quality of the response is significantly affected by the instruction and contextual information provided in the user prompt. Prompt engineering, which can also enhance LLM performance, is the process of designing and creating prompts that can efficiently guide the LLM in generating relevant responses. It involves providing clear instructions, format, context, and relevant information within the user prompt itself [38].

For this work, we focused on the frontier multimodal LLM called the generative pre-trained transformer (GPT 4 and GPT 4o), developed by OpenAI. GPT is trained on vast amounts of general domain data obtained from a variety of sources such as textbooks and web pages). It can be adapted for specific nuclear engineering domain applications via fine-tuning, RAG, and prompt engineering. In this work, we focused on RAG and prompt engineering approaches. In our previous work [39], we built a custom RAG pipeline using GPT 3.5 (see Figure 11 for reference). However, in the present work, we used the inbuilt multi-modal RAG capability available on the OpenAI web interface for GPT 4 and GPT 4o. Unlike fine-tuning, RAG does not involve additional training of the LLM. Instead, it uses an informational retrieval system to create a vector database from the external knowledge source. A similarity search is conducted between the user query and the vector database, and the matching information is passed to the LLM, which then uses this information to respond to the user query.

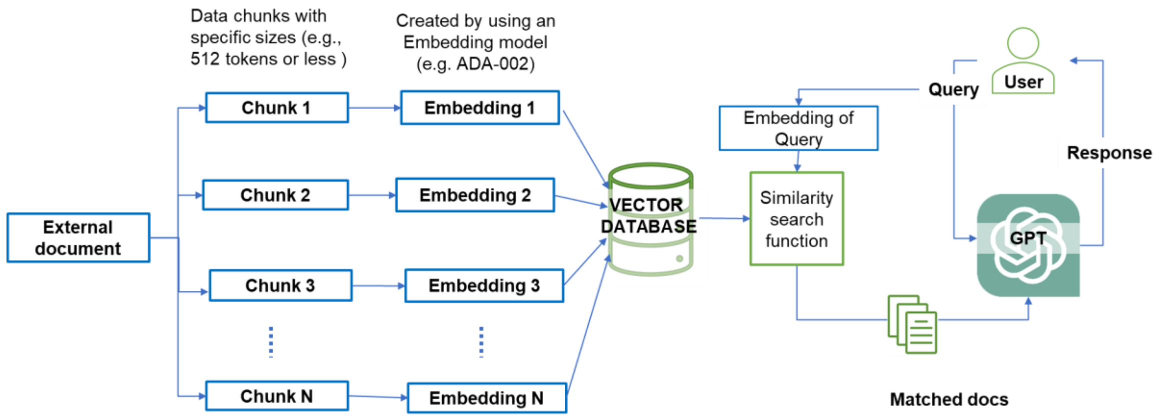


Figure 11. RAG using GPT 3.5 [39].

We focused on use of the GAI model for the risk assessment and design optimization of safety-critical DI&C systems. Two target applications were considered in this work:

- Safety assurance and design optimization of DI&C systems.
- Diversity and defense in depth (D3) applications.

This work summarizes three case studies related to the two applications (i.e., D3 and safety assurance and design optimization) where GAI may be used as a proxy expert to support various tasks that involve:

- Knowledge abstraction and reasoning (e.g., hazard identification, scenario analysis, and causal and counterfactual reasoning for accident scenarios)
- Information retrieval and extraction (e.g., extracting information related to control structure)
- Instruction-based knowledge management.

The three case studies presented in this work focus on domain specific knowledge pertaining to DI&C of light water reactor nuclear power plants, and problem-solving capabilities of GPT, customized by RAG and prompt engineering using selected industry datasets and references. These case studies are as follows:

- Case A: STPA-Basics. Testing and analysis of GPT capabilities for basic functions in STPA.
- Case B: STPA-DI&C. Testing and analysis of GPT capabilities for STPA of DI&C systems.
- Case C: D3-DI&C: Testing and analysis of GPT capabilities for D3 application in DI&C systems.

The objectives of our case studies are centered around exploring the potential use of GAI in DI&C, assessing the significance of its impact, and identifying key issues and challenges pertaining to:

1. Required data sufficiency vs. existing/identified data deficiencies and gaps.
2. Required algorithm maturity vs. existing/identified algorithmic deficiencies and gaps.
3. Possible failure modes, related risks, and consequences.

The following section summarizes the results and key findings related to the three case studies considered in this analysis.

6.2 Multimodal LLM for STPA (Case Studies A and B)

Hazard analysis is an important part of the reliability and safety assessment of nuclear reactor systems. Among different methodologies for hazard analysis are FTA, event tree analysis, failure modes and effects criticality analysis, hazard and operability analysis, STPA [20], etc. Among these, STPA is significant for DI&C applications, as it provides causal analysis of accidents in consideration of all the various factors, including human factors and software. It not only focuses on component reliability but also identifies possible unsafe interactions between system components. From the perspective of complex system analysis, it delves into the unknown-unknown quadrant of the Rumsfeld matrix. STPA is based on the System Theoretic Accident Model and Process (STAMP). It works on complexity reduction using the top-down approach and involves accident causality analysis using system theory. The key steps in STPA are [20]: (1) define the purpose of the analysis, (2) model the control structure, (3) identify the unsafe control actions (UCAs), and (4) identify the loss scenario. One of the most crucial steps in the STPA, from the perspective of DI&C systems, is identifying the model of the control structure, as this requires detailed knowledge of the control architecture and the interactions between different components. Such information is documented in various reports, manuals, and literature related to the control system. The required information is present in different modalities (i.e., textual description as well as figures and block diagrams of the control system and its various components). Identifying the model of the control structure requires expert input and review of various documentation. Thus, this process requires significant resources and time. We place a higher emphasis on this step of STPA, and use the multimodal abilities of GPT to help extract the control structure. STPA is conducted in two parts. The first (Case A) focuses on testing and analysis of GPT for basic functions in STPA. The second part (case B) focuses on testing and analysis of GPT for STPA of DI&C systems. The inherent knowledge base of GPT does not cover specific information related to the DI&C system, so we supplement the GPT with external knowledge for RAG. This external knowledge base consists of: (1) information related to standard procedure and guidelines (e.g., Guide on STPA Process), (2) information related to the system for which we need to perform STPA (e.g., the specific DI&C system). Additionally, prompt engineering approaches such as prompt changing and chain-of-thought are used to enhance and improve the response of GPT.

6.2.1 Case A: STPA-Basics

For the basic STPA, we first performed domain testing to examine the inherent knowledge base of GPT for STPA. Our testing indicates that GPT has adequate knowledge of STPA. In the initial queries, GPT identified all the key steps of STPA. However, it missed some specific details. Therefore, we used the RAG with STPA handbook as a reference external source to update GPT with any specific information. Table 14 describes the key steps in STPA, as obtained by GPT 4o.

Table 14. Key steps in STPA (generated by GPT 4o using RAG).

Aspect	**Description**
What is STPA?	STPA is a hazard analysis technique based on systems theory, focusing on identifying unsafe interactions within a system rather than just component failures. It was developed to address the limitations of traditional safety analysis methods, especially in complex, software-intensive systems.
Key Steps in STPA	**Description**
Step 1: Define the Purpose of the Analysis	Establish the system's goals, boundaries, and the context in which it operates. Identify the key safety constraints that must be maintained to ensure safe operation. This step includes identifying losses, system-level hazards, and system-level constraints.
Step 2: Model the Control Structure	Create a control structure diagram that represents the system, including controllers, actuators, sensors, and controlled processes. The control structure captures functional relationships and interactions within the system by modeling feedback control loops.
Step 3: Identify UCAs	Analyze the control actions within the system to identify any that could lead to a hazardous state. UCAs are those that, when provided, not provided, provided at the wrong time, or provided in the wrong order, could cause an unsafe condition. This step helps to develop functional requirements and constraints for the system.
Step 4: Identify Loss Scenarios	Develop scenarios that describe how the UCAs identified in Step 3 could lead to a loss (accident or incident). This step involves considering environmental conditions, component failures, human errors, and other factors that could contribute to the scenario. The goal is to understand how these UCAs could occur in the system.
Step 5: Develop Mitigation Strategies	Propose measures or design changes that can mitigate or eliminate the loss scenarios identified in Step 4. These could include changes to the control structure, additional safety constraints, or improvements to human-machine interfaces. The focus is on preventing hazards and ensuring safe operation.
Step 6: Document and Verify	Compile the findings and proposed mitigations into a comprehensive report. Verification involves checking that the proposed mitigations effectively address the identified hazards and that no new hazards have been introduced. This step also ensures traceability from identified losses to specific control actions and mitigations.

Note that this work does not represent the first attempt to use an LLM to conduct STPA. In Qi et al. [40], ChatGPT (GPT 3.5) was used to perform STPA for automatic emergency brake (AEB) and electricity demand side management (DSM) systems. However, as that work utilized unimodal GPT (GPT 3.5, text-only version), it required significant human interaction/intervention and feedback to obtain adequate results for STPA. Table 15 illustrates the major differences between that work and ours.

Table 15. Major differences between Qi et al. [40] and our work.

Feature	**Yi Qi et al***	**Our work**
Modality	Unimodal	Multimodal
LLM	GPT 3.5 (text only)	GPT 4, GPT 4o
RAG	No	Yes

Unlike unimodal GPT (i.e., GPT 3.5), the multimodal GPTs (GPT 4 and GPT 4o) work with both image and text data. We used this capability of GPT 4 and GPT4o to complete the most crucial step in STPA (i.e., identifying the model of the control structure). STPA using GPT is conducted in two steps. In the first step, the control structure is identified using system flow diagrams. In the second step, the GPT is instructed to perform STPA using the information extracted from the flow diagram. For less complex systems, GPT was able to successfully identify the model of the control structure. We tested the multimodal GPT for STPA of AEB systems. For example, in the case of the AEB system shown in Figure 12, GPT correctly identified the information flow (input and output) for all the blocks and summarized the control structure for STPA, without human feedback (see Table 16 for the GPT response).

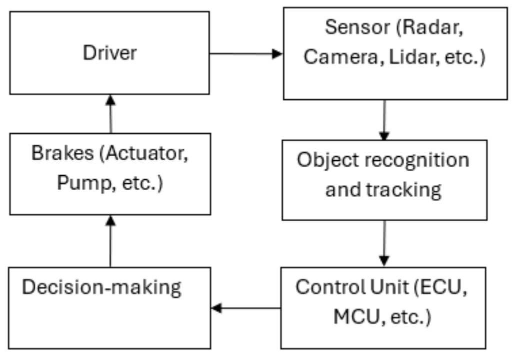


Figure 12. Flow chart for AEB system.

Table 16. Model control structure for AEB generated by GPT 4o.

Block Name	Input	Output
Driver	N/A	Signal to Sensor
Sensor (Radar, Camera, Lidar, etc.)	Signal from Driver	Data to Object Recognition and Tracking
Object Recognition and Tracking	Data from Sensor	Information to Control Unit
Control Unit (ECU, MCU, etc.)	Information from Object Recognition and Tracking	Commands to Decision Making
Decision Making	Commands from Control Unit	Commands to Brakes
Brakes (Actuator, Pump, etc.)	Commands from Decision Making	N/A

The response of GPT is governed by the contextual information present in the user prompt. Thus, the instructions for STPA must be carefully crafted based on initial testing and analysis. For example, in our initial test, GPT missed linking the hazards with the loss scenario. Therefore, we explicitly include this instruction in the user prompt. Based on the control structure for AEB, GPT was instructed to perform the STPA. Table 17–Table 19 list the key results obtained from GPT 4. It is evident from these results that GPT exhibits the capabilities and reasoning ability to perform basic STPA.

Table 17. Losses and hazards for AEB (obtained from GPT 4).

Loss ID	Loss Description	Hazard ID	Hazard Description
L-1	Loss of human life or injury to occupants	**H-1**	Failure to detect obstacles
		H-2	False detection of obstacles leading to unnecessary braking

		H-3	Delayed braking response
		H-4	Braking system failure
		H-5	Incorrect decision-making based on sensor data
L-2	Damage to the vehicle	**H-1**	Failure to detect obstacles
		H-2	False detection of obstacles leading to unnecessary braking
		H-3	Delayed braking response
		H-4	Braking system failure
		H-5	Incorrect decision-making based on sensor data
L-3	Damage to objects or other vehicles	**H-1**	Failure to detect obstacles
		H-2	False detection of obstacles leading to unnecessary braking
		H-3	Delayed braking response
		H-4	Braking system failure
		H-5	Incorrect decision-making based on sensor data
L-4	Loss of control of the vehicle leading to secondary accidents	**H-2**	False detection of obstacles leading to unnecessary braking
		H-4	Braking system failure
		H-5	Incorrect decision-making based on sensor data
L-5	Loss of mission (e.g., transportation mission)	**H-4**	Braking system failure
		H-5	Incorrect decision-making based on sensor data

Table 18. UCAs obtained from GPT 4.

Control Action	Not Providing	Providing Incorrectly	Wrong Timing/Order	Stopped Too Soon/Applied Too Long
Driver signals to sensor	H-1, H-3	H-2, H-5	H-3, H-5	N/A
Sensor detects objects	H-1	H-2, H-5	H-3, H-5	N/A
Object Recognition and Tracking processes data	H-1	H-2, H-5	H-3, H-5	N/A
Control Unit makes decisions	H-1, H-4, H-5	H-2, H-5	H-3, H-5	N/A
Decision-making sends commands to brakes	H-1, H-4, H-5	H-2, H-5	H-3, H-5	H-4
Brakes execute commands	H-1, H-4, H-5	H-2, H-5	H-3, H-5	H-4

Table 19. Loss scenarios and control strategies (obtained from GPT 4).

Scenario ID	Scenario Description	Hazard ID	Associated Losses	Control Strategies

S-1	Sensor fails to detect an obstacle	**H-1**	L-1, L-2, L-3	Redundant sensors, regular calibration, fail-safes
S-2	Sensor falsely detects an obstacle	**H-2**	L-1, L-2, L-4	Cross-verification, filtering algorithms, driver override
S-3	Decision-making unit delays sending the braking command	**H-3**	L-1, L-2, L-3	Real-time processing, prioritizing braking commands, performance monitoring
S-4	Brakes do not execute the command properly	**H-4**	L-1, L-2, L-3, L-4, L-5	Regular maintenance, redundant braking mechanisms, emergency protocols
S-5	Control unit makes incorrect decisions based on sensor data	**H-5**	L-1, L-2, L-3, L-4, L-5	Improved algorithms, real-time diagnostics, adaptive ML

6.2.2 Case B: STPA-DI&C

For the DI&C application, we focused on the Plant Protection System (PPS) of the APR 1400 as our target application. Open-source information related to APR1400 [41] was used in this analysis. For complex systems (such as PPS), we tested GPT with different volumes of information to examine its ability to extract relevant information from the knowledge source. In our initial attempt, we applied RAG with the entire document of the DI&C system of APR 1400. This document contains detailed information regarding PPS and other systems and their components. In this case, GPT performed the STPA with all the major steps. However, it was unable to accurately capture the model of the control structure. We also prompted GPT with the detailed figure of the PPS in the APR 1400 design control document (DCD) [41] to extract information on the control structure (inputs and outputs of all the blocks). This PPS figure contains a variety of information: (1) major blocks/components, (2) information flow between various blocks (indicated by the directed arrows), (3) type of connections or links between blocks/components (e.g., hardware cable, fiber optic cable, non-fiber-optic serial data link, intra-division safety system data network). In a blank attempt, we provided a picture of the PPS (with no separate information about it) and prompted it to identify the control structure (information flow based on inputs and outputs). In this case, GPT was able to extract the correct information flow for some of the blocks, but in the case of complex interactions involving multiple inputs and outputs, different links, and inter-channel communication, GPT made multiple errors. The response of GPT was also affected by the proximity of arrows and blocks.

GPT's response depends on the question and the contextual information included in the user prompt. Formulating the right instruction requires a few iterations and systematic testing and evaluation. GPT is trained using vast amounts of general domain datasets, but it lacks domain-specific information on DI&C systems in nuclear reactors. However, by following systematic procedure, RAG, and prompt engineering, the response of GPT can be significantly improved. Based on our testing and evaluation, we created a systematic procedure to conduct STPA using GPT. We also created a flowchart to depict the procedure for conducting STPA using GPT (see Figure 13 for the major steps in this procedure).

For the first step, we created a set of instructions for STPA. These instructions were generated based on the inherent knowledge of GPT and the additional information from the document on STPA. GPT is not familiar with the specific details (basic element descriptions, notations in block diagrams, terminologies, etc.) related to the DI&C system. Thus, in the next step, we created a custom dictionary of different elements and basic blocks in the DI&C system (i.e., PPS, in the current case). This dictionary enhances the contextual information and helps GPT to focus on important information needed to decipher the control structure. The PPS dictionary consists of:

- Descriptions of all the relevant PPS components (see Table 20)
- Types of connection links (see Figure 14)
- Supplementary information to decipher the ambiguous elements in the control flow diagram (see Table 21).

The descriptions of relevant components for the dictionary were extracted using GPT with RAG. GPT was prompted with the names or acronyms of different components related to PPS, enabling this information to be obtained from the APR1400 DCD [41]. The supplementary information was used in later stages for situations involving ambiguous block representation and information flow. For complex figures and block diagrams, GPT responds better when we gradually increase the complexity. Thus, we used a prompt chaining approach to instruct GPT to create the control structure model over multiple steps, and we gradually increased the complexity by adding more blocks. Finally, we asked GPT to identify the control structure for the entire channel. For situations in which information flow is not evident (ambiguous situations) due to the lack of directed arrows, supplementary information from the dictionary is provided to the GPT. Once the model control structure is identified, GPT is instructed to perform STPA by following the procedure it extracted in step 1.

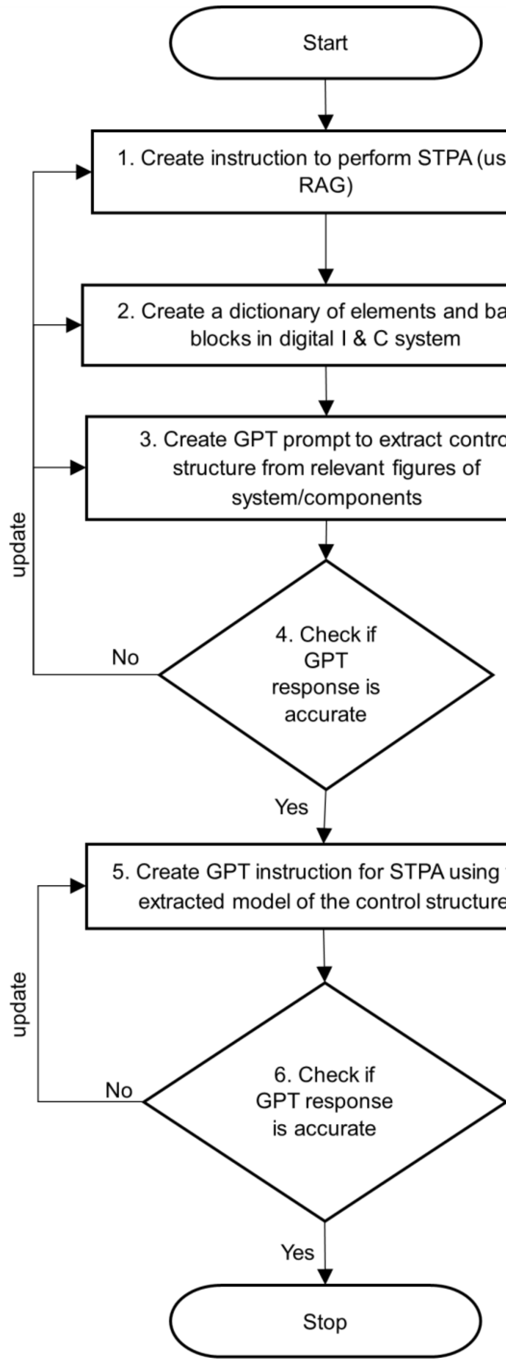


Figure 13. Flow chart describing the process of conducting STPA by using GPT.

The response of GPT is sensitive to the prompt (question and contextual information). Furthermore, a certain degree of stochasticity is associated with the LLM-generated content. The degree of stochasticity of GPT's response can be modified by adjusting its model parameters, such as temperature and top-p. In this work, we used the default values for the temperature and top-p parameter, as recommended by OpenAI for obtaining a balanced response. Given the stochastic nature of GPT and its sensitivity to the content of the prompt, it is important to review and check the information it generates. Thus, in our flowchart for procedures, we include two checkboxes to verify the accuracy of GPT's response. As an example, Figure 15 shows one of the PPS channels (channel A). For this case, we followed the procedure outlined in Figure 13. As shown in Table 22, GPT identified the correct control structure along with the type of connection link from the block diagram of the PPS. Once the control structure is identified, GPT is instructed to perform STPA per the instructions outlined in the initial step.

Table 20. Descriptions of key components (obtained from GPT using RAG).

Component/Block	Description
Plant Protection System (PPS)	The PPS performs the reactor protection system function and the engineered safety features (ESF) initiation function. It consists of four divisions located in separate I&C equipment rooms. Each division contains input/output modules, bistable processors, local coincidence logic (LCL) processors, and other hardware for interfacing with other PPS divisions. The PPS functions include bistable trip logic, LCL, reactor trip initiation, and testing.
TR	"TR" in the context of PPS often refers to the trip signals generated by the bistable processors. These trip signals are sent to the LCL processors within the PPS for further processing and determination of the final reactor trip signal.
Auxiliary Process Cabinet-Safety (APC-S)	The APC-S functions include signal conditioning/splitting for safety field sensor signals shared by the PPS, core protection calculator system (CPCS), and ESF-CCS loop controller (LC). It also provides isolation when signals are split to non-safety systems.
Ex-core Neutron Flux Monitoring System (ENFMS)	The ENFMS consists of four redundant safety channels and two redundant startup/control signal processing drawers. It includes neutron detectors around the reactor core and signal processing equipment in the auxiliary building. It provides neutron flux information across a wide range of power levels.
Core Protection Calculator System	The CPCS calculates the departure from nucleate boiling ratio (DNBR) and low power density (LPD) sing input signals and compares these with trip setpoints. It provides trip and pre-trip signals to the PPS. It includes CEACs to monitor CEA positions and apply penalty factors for deviations (DCD reference).
Bistable Processor	The bistable processor compares input signals from process measurement instrumentation to setpoints and generates trip signals when a parameter exceeds its setpoint. These signals are sent to the LCL processors within the PPS.
Operator Module (OM)	The OM allows the operator to monitor inputs and outputs from the PPS, as well as change addressable constants within the CPCS according to administrative procedures. It is a key interface for operators within the PPS.
LCL Processor	The LCL processor determines the output state based on the bistable trip signals and channel bypasses. For RPS, the LCL outputs are transmitted to the reactor trip switchgear system through selective 2-out-of-4 initiation logic.
Initiating Circuit	The initiating circuit in PPS involves the 2-out-of-4 LCL, which generates reactor trip signals. These signals de-energize the undervoltage trip devices in the reactor trip circuit breaker, leading to a reactor trip by allowing CEAs to drop into the core.
Reactor Trip	Reactor trip is initiated by the PPS when it detects a parameter exceeding set limits. The trip signals are processed by the LCL and result in the de-energizing of the undervoltage trip devices, causing the reactor to shut down safely.
Maintenance and Test Panel (MTP)	The MTP is used for testing and maintenance of PPS components, allowing changes to setpoints and monitoring of actual values during operation. It plays a crucial role in ensuring PPS reliability during both operation and maintenance.
Interface and Test Processor (ITP)	The ITP provides testing capabilities for the bistable logic within the PPS. It verifies the functionality of analog inputs, trip setpoints, and status information, ensuring the integrity of the trip logic.
Engineered Safety Features-Component Control System (ESF-CCS)	The ESF-CCS receives ESFAS initiation signals from the PPS and RMS, electrical panels, or operators, and generates actuation signals to the ESF system equipment, including the emergency diesel generator loading sequencer following loss of offsite power.
Division A, B, C, D	These refer to the four redundant divisions of the PPS, each located in separate I&C equipment rooms. Each division processes signals independently to ensure system reliability and safety, with selective 2-out-of-4 logic used for reactor trip initiation.

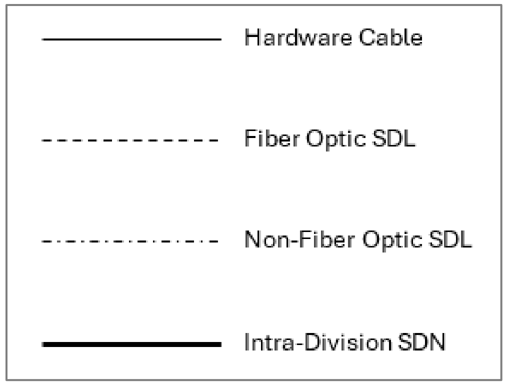
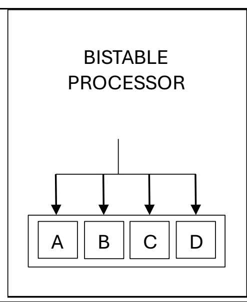
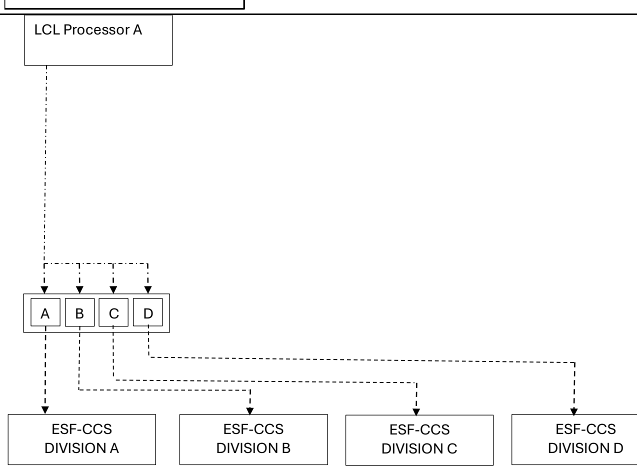


Figure 14. Types of connection links.

Table 21. Supplementary information required to decipher the control structure (part of the dictionary).

S. No.	Content	Related description
1	Direction of information flow for the MTP, OM, and ITP	Bistable and LCL processors send status data to the MTP, OM, and ITP through the Safety-system Data Network (SDN). Bistable processors and LCL processors receive testing data from the MTP through the SDN.
2		Different block representation of bistable processor
3		In the case of more complex block diagrams that involve connection between channels, sometimes ports with channel names are used in the diagram. For example, in the simplified block diagram that depicts a part of the complex block diagram, ports for channels A, B, C, and D are indicated by their names.

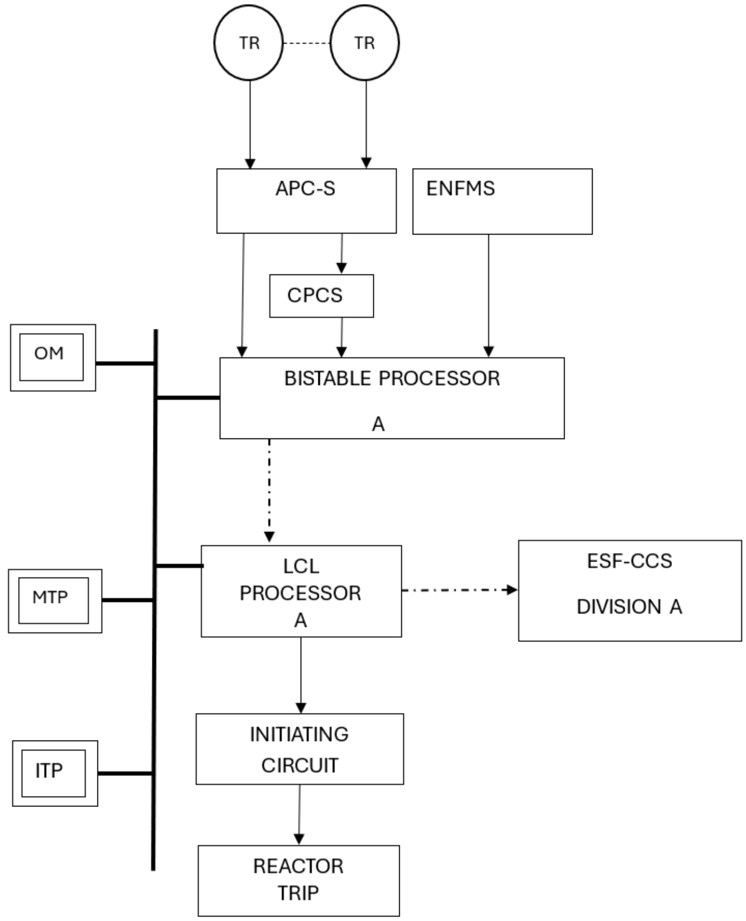


Figure 15. Abbreviated view of channel A of the PPS in APR 1400 based on [41].

Table 22. Control structure for channel A of the PPS.

Block	Inputs	Outputs	Type of Connection
TR	-	APC-S, TR	Hardware Cable, Fiber Optic SDL
TR	-	APC-S, TR	Hardware Cable, Fiber Optic SDL
APC-S	TR, TR	CPCS, Bistable Processor A	Hardware Cable
CPCS	APC-S	Bistable Processor A	Hardware Cable
ENFMS	-	Bistable Processor A	Hardware Cable
BISTABLE PROCESSOR A	APC-S, CPCS, ENFMS, MTP (testing data)	LCL Processor A, OM, MTP, ITP	Hardware Cable, Intra-Division SDN
LCL PROCESSOR A	Bistable Processor A, MTP (testing data)	Initiating Circuit, ESF-CCS Division A, OM, MTP, ITP	Hardware Cable, Intra-Division SDN, Non-Fiber Optic SDL
INITIATING CIRCUIT	LCL Processor A	Reactor Trip	Hardware Cable
REACTOR TRIP	Initiating Circuit	-	Hardware Cable
OM	Bistable Processor A, LCL Processor A	-	Intra-Division SDN
MTP	Bistable Processor A, LCL Processor A	Bistable Processor A, LCL Processor A	Intra-Division SDN
ITP	Bistable Processor A, LCL Processor A	-	Intra-Division SDN
ESF-CCS DIVISION A	LCL Processor A	-	Non-Fiber Optic SDL

A figure showing the PPS and all four channels—similar to the abbreviated view of channel A from Figure 15—can be found in [41]. Table 23 provides the PPS control structure obtained by GPT.

Table 23. Summary of the PPS control structure.

Channel	Block	Input(s)	Output(s)	Type of Link/Line	Comments
A	**TR (Sensor and Transmitter)**	None (in this context)	APC-S	Hardware Cable	Sends data to APC-S
A	**APC-S**	TR (Sensor and Transmitter)	CPCS, Bistable Processor	Hardware Cable	Receives data from TR, sends to CPCS and Bistable Processor
A	**ENFMS**	None (in this context)	Bistable Processor	Hardware Cable	Sends data to Bistable Processor
A	**CPCS**	APC-S	Bistable Processor	Hardware Cable	Receives data from APC-S, sends to Bistable Processor
A	**Bistable Processor**	APC-S, CPCS, ENFMS	LCL Processor A (Non-Fiber Optic Serial Data Link), LCL Processors B, C, D (Fiber Optic Serial Data Link), OM, MTP, ITP	Non-Fiber Optic Serial Data Link (to LCL Processor A), Fiber Optic Serial Data Link (to LCL Processors B, C, D), Safety-system Data Network (to OM, MTP, ITP)	Connects across channels to LCL Processors B, C, D
A	**LCL Processor**	Bistable Processor, MTP	Initiating Circuit, ESF-CCS Divisions A, B, C, D	Hardware Cable (to Initiating Circuit), Non-Fiber Optic Serial Data Link (to ESF-CCS Division A), Fiber Optic Serial Data Link (to ESF-CCS Divisions B, C, D), Safety-system Data Network (to OM, MTP, ITP)	Receives cross-channel data from Bistable Processors B, C, D
A	**Initiation Circuit**	LCL Processor	Reactor Trip	Hardware Cable	Receives from LCL Processor, sends to Reactor Trip
A	**Reactor Trip**	Initiation Circuit	None (in this context)	Hardware Cable	Receives from Initiation Circuit
A	**ESF-CCS Division A**	LCL Processor A	None (in this context)	Non-Fiber Optic Serial Data Link	Receives data from LCL Processor A
A	**OM**	Bistable Processor, LCL Processor	None (in this context)	Safety-system Data Network	Receives status data from Bistable Processor and LCL Processor
A	**MTP**	Bistable Processor, LCL Processor	Bistable Processor, LCL Processor	Safety-system Data Network	Receives status data from Bistable Processor and LCL Processor, sends testing data to them
A	**ITP**	Bistable Processor, LCL Processor	None (in this context)	Safety-system Data Network	Receives status data from Bistable Processor and LCL Processor

Along with its response for the control structure in Table 23, GPT also provides information on communications between the channels:

"This pattern repeats similarly across Channels B, C, and D, with corresponding connections to their respective LCL Processors and ESF-CCS Divisions.

- *Cross-Channel Connections: Bistable Processors in each channel (A, B, C, D) are interconnected with LCL Processors in the other channels via Fiber Optic Serial Data Links. This cross-channel connectivity ensures redundancy and safety.*
- *Safety-system Data Network: OM, MTP, and ITP receive status data and communicate testing data across channels through the Safety-system Data Network."*

Based on the control structure identified by the GPT, Table 24 shows the corresponding results of the STPA performed by GPT. For brevity and clarity, GPT was instructed to summarize the key steps in a tabular format. These results demonstrate that GPT can support STPA for complex systems, but it requires systematic guidance and relevant contextual information to successfully complete the analysis. Furthermore, the response generated by GPT must be treated carefully, including verifying whether it conforms to specific requirements (e.g., format, level of detail, specific element of STPA, and regulatory compliance).

Table 24. Key steps of STPA for PPS.

Step	**Details**
Purpose of the Analysis	Ensure the PPS operates safely by preventing hazardous conditions that could lead to losses such as reactor trip failure or failure to activate ESF
Hazards (H)	H1: Reactor fails to trip when required H2: Reactor trips unnecessarily H3: ESF fails to activate when required H4: ESF activates unnecessarily
Control Structure	- **TR (Sensor and Transmitter):** Provides data to PPS - **APC-S:** Processes sensor data, sends to CPCS and Bistable Processors - **CPCS:** Calculates parameters for reactor trip - **Bistable Processors:** Evaluate input, send signals to LCL Processors - **LCL Processors:** Determine reactor trip and ESF signals - **Initiating Circuit:** Activates reactor trip - **ESF-CCS Divisions:** Activate safety features - **OM, MTP, ITP:** Testing, maintenance, monitoring
UCAs	UCA1: Bistable Processor fails to send a trip signal (H1) UCA2: Bistable Processor sends an unnecessary trip signal (H2) UCA3: LCL Processor fails to activate ESF (H3) UCA4: LCL Processor activates ESF unnecessarily (H4) UCA5: APC-S provides incorrect data (H1, H2) UCA6: OM, MTP, ITP override trip signals incorrectly (H1, H2)
Loss Scenarios (LS)	LS1: Sensor failure in TR leads to missed trip condition (H1) LS2: APC-S malfunction causes erroneous data and unnecessary trip (H2) LS3: Fiber Optic Serial Data Link failure causes ESF activation failure (H3) LS4: MTP sends incorrect testing data, preventing a needed trip (H1)
Causal Factors	- **Causal Factor 1:** Software bugs in CPCS or Bistable Processors - **Causal Factor 2:** Hardware failures in TR sensors, Fiber Optic Serial Data Links, APC-S - **Causal Factor 3:** Human errors in OM, MTP, or ITP - **Causal Factor 4:** Cross-channel communication issues

Safety Requirements (SR)	SR1: Bistable Processors must provide accurate trip signals SR2: LCL Processors must reliably determine trip and ESF signals SR3: APC-S must validate sensor data SR4: OM, MTP, and ITP must prevent incorrect overrides
Safety Measures	- **Safety Mechanism 1:** Implement redundancy checks in Bistable and LCL Processors - **Safety Mechanism 2:** Use error-checking in APC-S - **Safety Mechanism 3:** Regular testing of Fiber Optic Links - **Safety Mechanism 4:** Training and safeguards for OM, MTP, ITP operators

6.3 Case C: D3-DI&C

D3 analysis of the nuclear power plant DI&C systems is a comprehensive process that requires identification and mitigation of vulnerabilities due to CCFs. DI&C systems are susceptible to CCFs due to shared software, hardware components, environmental factors, and human errors. D3 analysis of DI&C systems is based on different regulatory documents. These documents provide systematic guidance on requirements and criteria for D3 evaluation of DI&C systems and corresponding components. In this case study, we focused on information retrieval, knowledge abstraction, and the reasoning ability of GPT in order to support reviews and decision analyses of DI&C systems from a D3 perspective.

The knowledge base for this case study consists of two sets of references: (1) standard procedures and guidelines that provide requirements and decision criteria for evaluating DI&C systems from a D3 perspective [42] [43] [44], and (2) information related to the target system and its components [41] [45]. For this case study, we again chose the PPS of APR1400 as our target DI&C system. Open-source information on APR1400, along with regulatory documents, were used in this analysis. We began our analysis with the relevant regulatory documents from the knowledge base that provides the requirements and criteria for D3 assessment. For this application, GPT was first used to extract requirements and criteria from the regulatory documents for decision-making purposes. Based on the extracted requirements and criteria, GPT was then used to search relevant information (evidence) from documentation of the target application (PPS for APR1400) in order to support the decision analysis.

It is instructive to note that GAI, like GPT, is stochastic in nature and searches for specific queries based on the associated contextual information. During RAG, the volume of information in the external knowledge base (relevant documents) also affects the response of GPT. GPT searches for relevant information through different modalities (e.g., tables, figures, text, etc.) included in the documents. However, our initial testing indicates that text is the primary modality, meaning that greater emphasis is placed on textual data. When the information is spread across different data modalities, GPT can miss specific details and make errors during information retrieval and extraction. For example, when we ask GPT to retrieve or extract content from a specific table in the regulatory document, it makes some mistakes or mixes the information from different sections of the document. However, if we separate the figures and tables from the document and then instruct GPT to extract relevant information from them, GPT is less susceptible to mistakes. Furthermore, the figures (flowcharts) and tables in regulatory documents often provide compact, consolidated, and structured information related to the decision logic or procedure (e.g., the flowchart in BTP-7-19 [Figure 7.19.1] provides the decision logic for detailed D3 assessment [42]). Thus, they are considered important for extracting crucial information from regulatory documents.

Based on our initial testing and analysis, we created a flowchart to describe the process of conducting D3 analysis using GPT (see Figure 16). In the initial step (step 1), we supplemented GPT with the relevant document (e.g., BTP-7-19 [42]) from the knowledge base and instructed it to extract high-level regulatory requirements (see Table 25, for example). These high-level requirements provide the basic context to begin the D3 analysis. Next, we identified key figures and tables (e.g., decision flow charts and requirement tables) from relevant regulatory documents (step 2) and instructed GPT to extract the decision logic from individual tables and figures (step 3). Using the extracted decision logic, GPT was instructed to look for supporting information (evidence) from the DI&C documentation (step 4). As GPT is highly sensitive to the contextual information associated with the query, multiple checkpoints were added in the flow chart so that the instruction (prompt) for each step could be adjusted to produce the desired outcome (relevant response with appropriate level of detail).

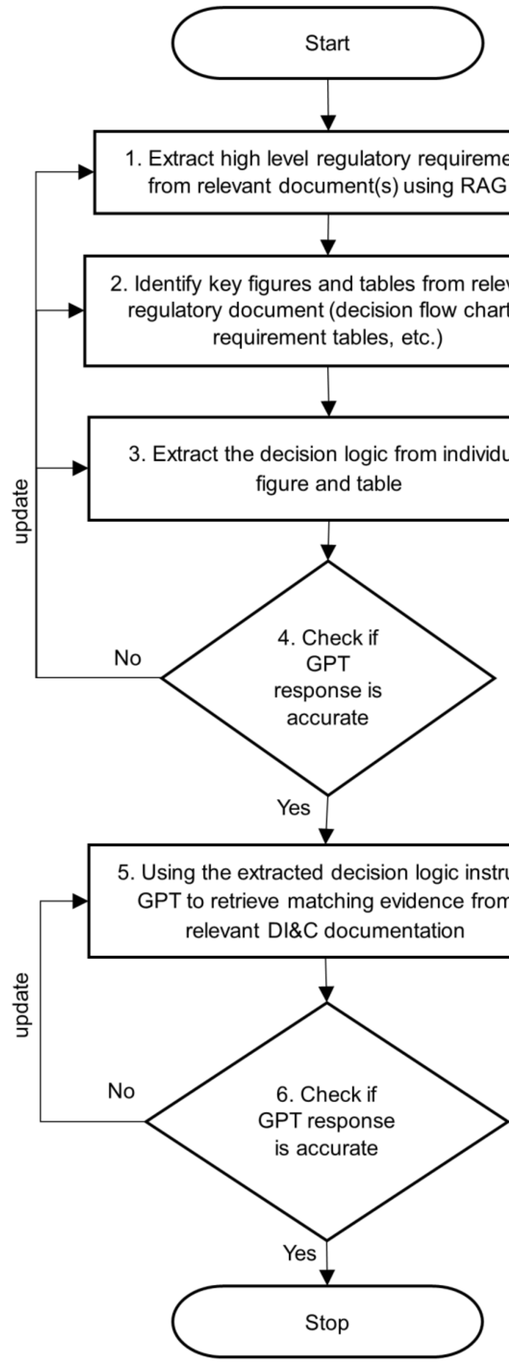


Figure 16. Flow chart describing the process of conducting D3 analysis using GPT.

Table 25. High-level description of various regulatory documents (obtained from GPT-4 by using RAG with relevant documents).

Document	Purpose	Key Components	Key Strategies/Guidance
NUREG/CR-6303	Provides a methodology for analyzing RPS to identify and mitigate vulnerabilities to common-mode failures (CMFs).	- **D3 Analysis**: Ensures design vulnerabilities are compensated by diversity or defense-in-depth. - **Block Diagrams**: System is broken into "blocks" for independent failure analysis. - **Echelons of Defense**: Four levels of defense are emphasized.	- **Types of Diversity**: Design, equipment, functional, human, signal, and software diversity are all critical to reducing the likelihood of CMFs.
NUREG/CR-7007	Offers strategies to implement diversity in nuclear power plant I&C systems to improve safety and prevent CMFs.	- **Diversity Strategies**: Functional, equipment, and human diversity are emphasized to prevent simultaneous failures in systems.	- **Functional Diversity**: Different methods for the same safety function. - **Equipment Diversity**: Different manufacturers or designs. - **Human Diversity**: Different teams involved in design and verification.
BTP 7-19 (Draft Revision 9)	Provides guidance for evaluating diversity and defense-in-depth in DI&C systems at nuclear power plants.	- **Evaluation Criteria**: Specific criteria for diversity and defense-in-depth. - **Common-Mode Failure Analysis**: Methods to identify and mitigate CMFs. - **Redundancy vs. Diversity**: Importance of differentiating and integrating both.	- **Manual Controls**: Independent and diverse manual controls are recommended as backups to digital systems.

Table 26 shows the decision logic extracted from the flowchart in BTP-7-19 and the corresponding qualitative assessment of PPS based on the information extracted from the APR1400 DCD [41]. Along with the information mentioned in Table 26, GPT also provided additional details based on specific criteria from BTP-7-19 [42]:

- **"*Section B.2.1 (Safety Significance)***: *The PPS qualifies as high safety significance due to its critical role in reactor protection and ESF initiation, directly affecting the safety margins and defense-in-depth of the nuclear plant.*
- ***Section B.2.2 (Exemption Criteria)***: *The PPS does not meet the exemption criteria because it is not a simple analog system, has complex digital interactions, and is highly integrated with other safety systems, increasing the potential for CCFs.*
- ***Section B.3.1 (Eliminating CCF from Consideration)***: *Although the PPS includes diverse systems and redundancy, the inherent complexity of digital systems and shared software platforms means that CCFs cannot be fully eliminated without further detailed analysis.*
- ***Section B.3.2 (Detailed D3 Assessment)***: *A detailed D3 assessment is required to thoroughly evaluate the PPS's design, focusing on how it manages potential CCFs, the effectiveness of diversity measures, and the robustness of its safety functions under different failure scenarios."*

Table 27 shows another example in which information extracted from the diversity usage table (Table 2.1 in NUREG/CR-7007 [44]) is used to guide the diversity evaluation of the PPS. As the DCD does not contain detailed information related to the D3 analysis of APR1400, another document [45] that specifically focuses on D3 of APR1400 served as the source of external knowledge (evidence) for RAG.

The results presented in this section indicate that GPT can be used for information retrieval, knowledge abstraction, and reasoning to support D3 analysis of reactor systems. However, it systematic guidance and instruction are needed to provide the expected outcome with the required level of detail and accuracy. For comprehensive evaluation, it is important to consider all the requirements and criteria mentioned in the regulatory documents. GPT can extract decision logic from individual tables and figures, but combining decision logic from multiple tables and figures in text format can be challenging. One way to address this challenge is to create knowledge graphs to capture the decision logic from multiple modalities (figures, tables, and textual data). Furthermore, RAG with a knowledge graph carries the potential to significantly enhance the reliability of LLM-generated content [46] [47].

Table 26. Decision logic for D3 assessment and evaluation of PPS (extracted by GPT using RAG).

Step	Decision Point	Action	Next Step	Relevant Details from DCD and BTP 7-19	Decision
A.1	Safety-significance determination (Section B.2.1)	Determine the safety significance of the system based on its role in safety-related functions.	Proceed to A.2	**DCD Reference**: The PPS is a safety system that includes the RPS and ESFAS, both of which are critical for reactor trip initiation and ESF activation during abnormal conditions (DCD Sections 7.1.1.1, 7.2.1.2). **BTP 7-19 Criteria**: Systems whose failure could directly result in a significant reduction in defense in depth or public safety are considered high safety significance.	**High Safety Significance**
A.2	Are the acceptance criteria from section B.2.2 met?	Evaluate whether the PPS meets the criteria for exemption from detailed D3 assessment.	A.2.1 or A.2.2	**BTP 7-19 Criteria**: Systems can be exempt from detailed D3 assessment if they are of lower safety significance, use simple hardware (analog components), or have minimal interaction with other systems. **DCD Reference**: The PPS uses complex digital I&C systems, with extensive interconnections and integration with other safety systems, increasing the likelihood of CCFs (DCD Sections 7.1.1.1, 7.2.1). These characteristics mean that the PPS does not meet the simple, low-risk criteria for exemption.	**No, Criteria Not Met**
A.2.2	Can the CCF be eliminated from further consideration? (Section B.3.1)	Assess if internal diversity, testing, or alternative methods can fully mitigate CCF concerns.	A.2.1 or A.3	**BTP 7-19 Criteria**: To eliminate CCF from further consideration, the design must provide robust internal diversity, deterministic processing, or alternative methods that can ensure independence of safety functions. **DCD Reference**: While the PPS includes measures for diversity and redundancy, the complexity of its digital systems, shared software platforms, and integrated architecture make it challenging to fully eliminate CCF concerns (DCD Sections 7.2.1.9, 7.3.1). The presence of diverse actuation systems and manual backup controls, though helpful, may not fully address all potential CCF scenarios.	**No, CCF Cannot Be Fully Eliminated**
A.3	Detailed D3 assessment documentation required (Section B.3.2)	If CCF cannot be eliminated, proceed with a detailed D3 assessment to evaluate all potential failure modes.	Evaluate Points 2, 3, and 4	**BTP 7-19 Criteria**: A detailed D3 assessment is required for systems where CCFs cannot be fully eliminated and where the system has high safety significance. This assessment involves evaluating the adequacy of diverse means, analyzing failure modes, and ensuring that the design can manage CCFs effectively. **DCD Reference**: Given the PPS's role in critical safety functions, the complexity of its DI&C systems, and the	**Yes, Detailed D3 Assessment Required**

				challenges in fully mitigating CCFs, a detailed D3 assessment is necessary to ensure compliance with Nuclear Regulatory Commission expectations and safety standards (DCD Section 7.3.2).	

Table 27. Diversity table for PPS and diverse protection system (DPS) (obtained using GPT).

Diversity Attribute	Usage	Details
Design		
Different technologies	Yes	The PPS uses PLC technology, whereas the DPS uses FPGA logic controllers, providing technology diversity.
Different approach—same technology	No	The approach within the PPS technology remains consistent.
Different architectures	Yes	PPS and DPS have different system architectures, ensuring diversity in design.
Equipment Manufacturer		
Different manufacturer—different design	Yes	PPS and DPS are provided by different manufacturers with fundamentally different designs.
Same manufacturer—different design	No	Not applicable as the designs are from different manufacturers.
Different manufacturer—same design	No	The equipment designs differ across manufacturers.
Same manufacturer—different version	No	The designs are from different manufacturers, making this attribute not applicable.
Logic Processing Equipment		
Different logic processing architecture	Yes	PPS uses PLCs while DPS uses FPGA, offering diverse logic processing architectures.
Different logic processing versions in the same architecture	No	The architectures themselves are different, so versions within the same architecture are not applicable.
Different component integration architecture	Yes	Different methods of integrating components in PPS and DPS contribute to diversity.
Different data-flow architecture	Yes	The data-flow architectures differ between PPS and DPS, providing diversity.
Functional		
Different underlying mechanisms	Yes	The reactor trip mechanisms are diverse (PPS uses undervoltage, DPS uses shunt trip).
Different purpose, function, control logic, or actuation means	Yes	DPS and PPS serve similar purposes but use different control logic and actuation methods.
Different response-time scale	No	The response times are not designed to differ significantly.
Life-cycle		
Different design organizations/companies	Yes	PPS and DPS are designed by different companies.
Different management teams within the same company	Yes	The design and testing teams for PPS and DPS are distinct even within the same organization.
Different design/development teams (designers, engineers, programmers)	Yes	Separate teams are responsible for the design and development of PPS and DPS.
Different implementation/validation teams (testers, installers, or certification personnel)	Yes	Independent teams handle implementation and validation for PPS and DPS.
Logic		
Different algorithms, logic, and program architecture	Yes	PPS uses software in PLCs, while DPS uses HDL in FPGAs, ensuring diverse logic and program architecture.
Different timing or order of execution	Yes	The timing/order of logic execution differs between the PLC-based PPS and FPGA-based DPS.
Different runtime environment	Yes	PPS runs in a PLC environment, while DPS operates on FPGA, providing diverse runtime environments.
Different functional representation	Yes	PPS and DPS represent functions differently due to their distinct logic and architectures.
Signal		

Different parameters sensed by different physical effects	No	Both PPS and DPS share the same safety class sensors.
Different parameters sensed by the same physical effects	No	The same parameters and physical effects are used in PPS and DPS.
Same parameter sensed by a different redundant set of similar sensors	No	The sensors are shared between PPS and DPS.

6.4 Conclusion and Discussion on Future work

Frontier LLMs such as GPT 4 and GPT4o exhibit advanced reasoning capabilities and the ability to work with different data modalities (text, figures, tables, etc.). These models perform very well in general domain applications. However, their performance for specific domain applications such as nuclear engineering is governed by various factors such as:

- Required data sufficiency vs. existing data deficiency and gaps
- Required model capability vs. existing model deficiency
- Possible failure modes, related risks, and consequences.

By focusing on these aspects of LLM application in nuclear engineering, this work presents a preliminary study exploring the use of multimodal LLM (GPT 4 and GPT 4o) for conducting STPA and D3 analysis of DI&C systems. The PPS of APR 1400 was selected as the target system in this work. We used RAG and prompt engineering approaches (e.g., prompt chaining and chain of thought) to adapt GPT to specific applications. Our analysis and testing revealed that GPT performance is governed by various factors such as the volume and complexity of information used for RAG, the content of prompts (questions, contextual information, formatting instructions, etc.), and their sequence. Based on the results of our testing and evaluation, we formulated a set of procedures and guidance to conduct STPA and D3 analysis of the DI&C systems by using multimodal LLMs.

LLMs are powerful reasoning engines. However, the higher risk and consequences associated with nuclear reactor applications makes it is important to explore the capabilities and limitations of these models, then learn efficient ways to adapt them for nuclear reactor design and safety applications.

For FY 25 and subsequent years, our objective is to build upon the findings of the FY 24 study. FY 24 primarily focused on frontier models (GPT 4, GPT 4o, etc.), whereas our future work will extend testing and exploration to open-source unimodal and multimodal LLMs and adapt these models (via fine-tuning, RAG, and prompt engineering) for specific DI&C applications. Open-source models can provide more control in GAI-based application development and assessment. Furthermore, it can help us formulate a systematic workflow for the integration of GAI with other software modules in the LWRS-developed DI&C risk assessment framework (risk assessment module, significance evaluation module). Future work will also focus on the reliability and trustworthiness analysis of GAI models and incorporate intrusive and non-intrusive methods and metrics for evaluating the response of LLM.

7. SOFTWARE DEVELOPMENT ROADMAP

This section focuses on future development of software tools to perform hazard identification and evaluation of digital systems. This includes the development of RESHA, BAHAMAS, ORCAS, and hybrid CCF modeling, along with the integration of these tools. Together these tools will be referred to as the Software for Hazard Identification and Evaluation of Digital Systems (SHIELDS). We are planning to deploy our developed tools for specific use cases that are of interest to our industry partners. We will follow a cost-effective and time-efficient process by following a structured SDLC methodology and employing version control to support software management and collaboration.

Software development process

Structured SDLCs represent a cost-effective and time-efficient process for designing and developing high-quality software by providing a systematic way to divide the software development process into manageable, measurable tasks. There are multiple SDLC models (e.g., waterfall, spiral, and agile), but each has similar activities or phases [48]. Some of the main phases found in an SDLC include [49]:

- Requirement Phase: develop a detailed software requirement document based on the project goals and user needs.
- Design Phase: create a software design document outlining the overall software architecture and specifics based on the requirement document.
- Implementation Phase: develop codes according to the software design document.
- Testing Phase: once the software is developed, it is tested against the requirements specified in the software requirement document. This includes various types of testing, such as unit testing, integration testing, and system testing.
- Deployment Phase: after successful testing, the software is delivered to the customer for use. This entails installation, customization, and initial support.
- Maintenance Phase: after deployment, the software may require updates and maintenance to fix deficiencies, enhance its performance, or upgrade it with additional features.

Version control

Version control is a system that records changes to a software over time so that developers can later recall any specific version. The key aspects include (1) tracking histories, (2) branching and merging, (3) backup and restore, and (4) collaboration. It is an essential tool in SDLC, as it allows multiple developers to work on the same software without conflicting with each other's contributions. This project will use Git, a distributed version control system, in conjunction with various hosting services such as INL GitHub and INL GitLab to support the software management and collaboration. Git is currently the most widely used version control system for software development.

Software quality assurance

Another import aspect of software development is software quality assurance (SQA). SQA is a set of activities necessary to provide adequate confidence that a software product conforms to the set of functional and technical requirements specified for it. The SQA plan presents the required activities to enable consistent SQA implementation within the software development framework. It provides a standardized method of capturing software requirements, how those requirements will be implemented, how the software will be tested, how changes to the software will be controlled, and how software deficiencies will be handled. The SQA plan covers the periods of software development, maintenance and operations, and retirement. SQA is import because it (a) maintains compliance with DOE O 414.1C, "Quality Assurance," and 10 CFR 830, "Nuclear Safety Management," Subpart A, "Quality Assurance Requirements," and (b) provides a foundation for ensuring the quality of software and that the software development will consistently meet customer requirements. In DI&C software development, we will implement applicable requirements in conformance with INL internal document PDD-13610, "Software Quality Assurance Program." This document is interpreted from DOE Order 414.1C, "Quality Assurance," 10 CFR 830, Subpart A, "Quality Assurance Requirements," and ASME NQA-1-2000, "Quality Assurance Requirements for Nuclear Facility Applications."

Risk assessment modules and SHIELDS development

The SHIELDS platform will include RESHA, BAHAMAS, ORCAS, and hybrid CCF modeling, to accelerate hazard identification and evaluation for DI&C applications. The architecture of SHIELDS is illustrated in Figure 17. There are two main tasks for our developed approaches: (1) extract information from the system design document, requirement specification document, or system control structure document; (2) leverage standard methods such as STPA, BBN, ODC, FT analysis, and modified beta factor to identify hazards and evaluate system reliabilities. We will divide our development of risk assessment modules into two stages. Stage I is the implementation of standard methods, and Stage II is the processing of different text documents to extract the required information for Stage I application.

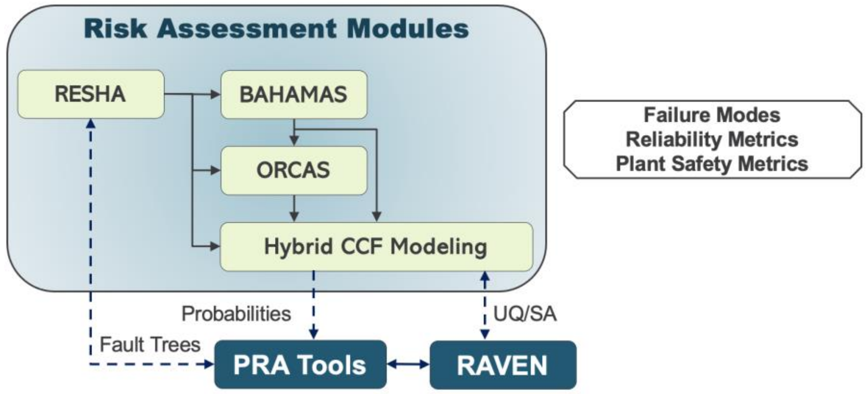


Figure 17. Architecture of the risk assessment platform (SHIELDS).

For example, there are seven important steps for analysts to follow when performing RESHA [5] analysis:

- Step 1: create a detailed representation of the system of interest.
- Step 2: develop an FT consisting of the hardware failures for a chosen function of the system of interest.
- Step 3: determine UCAs or unsafe information flows (UIFs), based on a redundancy-guided application of STPA.
- Step 4: construct an integrated FT by adding applicable UCAs/UIFs as basic events.
- Step 5: identify potential CCFs to add to the FT.
- Step 6: solve the FT for the minimal cut sets in order to determine critical failures in the design.
- Step 7: identify and provide guidance to eliminate critical failures or their causes

These steps can be categorized into two groups. The first group includes step 1, step 3, and step 5—targeting system design, specification, and requirement analysis, which will be used to generate system diagrams, UCAs/UIFs, and CCFs. The second group includes step 2, step 4, step 6, and step 7—targeting FT creation, modification, integration, and risk significance analysis. As illustrated in Figure 18, based on the provided system information, we are planning to utilize a model-based systems engineering (MBSE) methodology with system modeling language (SysML) to support RESHA development. MBSE is a formalized method to create a digital modeling environment usable to support the requirements, design, analysis, verification, and validation associated with the development of complex systems. SysML is a general-purpose modeling language for MBSE applications that provides a standard way of analyzing and visualizing the design of a system. With MBSE, the system diagrams created within a common modeling environment for RESHA analysis can be programmatically validated to remove inconsistencies by utilizing a common standard for all participants/stakeholders. This could significantly improve the RESHA analysis and reduce the number of defects commonly injected in a traditional document-based approach. The system diagrams will be used to identify UCAs/UIFs and potential CCFs, which will be inputted into the FT module (the right block in Figure 18) to identify the potential hazards and provide guidance on eliminating critical failures or their causes.

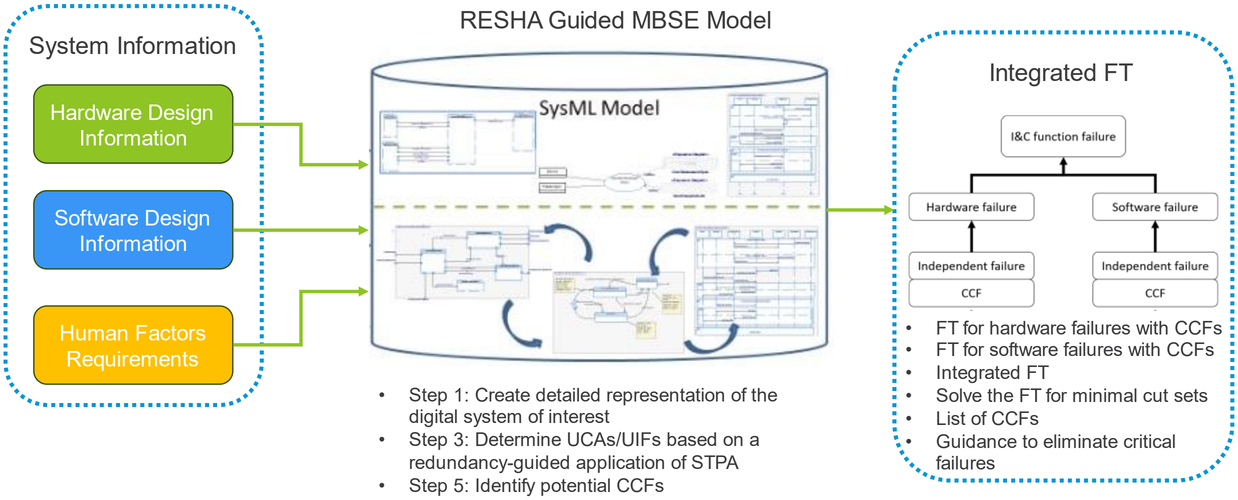


Figure 18. MBSE-aided RESHA analysis.

As discussed, development of each module in SHIELDS will consist of two stages. For the development of RESHA, the stages consist of the details listed below. In the coming FY, we will focus on the Stage I development, using the general structure of the FT module illustrated in Figure 19.

- Stage I for RESHA: develop an FT module to process hardware, software, and CCFs, and provide guidance to eliminate critical failures or their causes.
- Stage II for RESHA: use SysML to create MBSE system diagrams based on existing system information, including hardware design, software design, and human factors requirement documents. Utilize the MBSE system diagrams to identify UCAs/UIFs and potential CCFs.

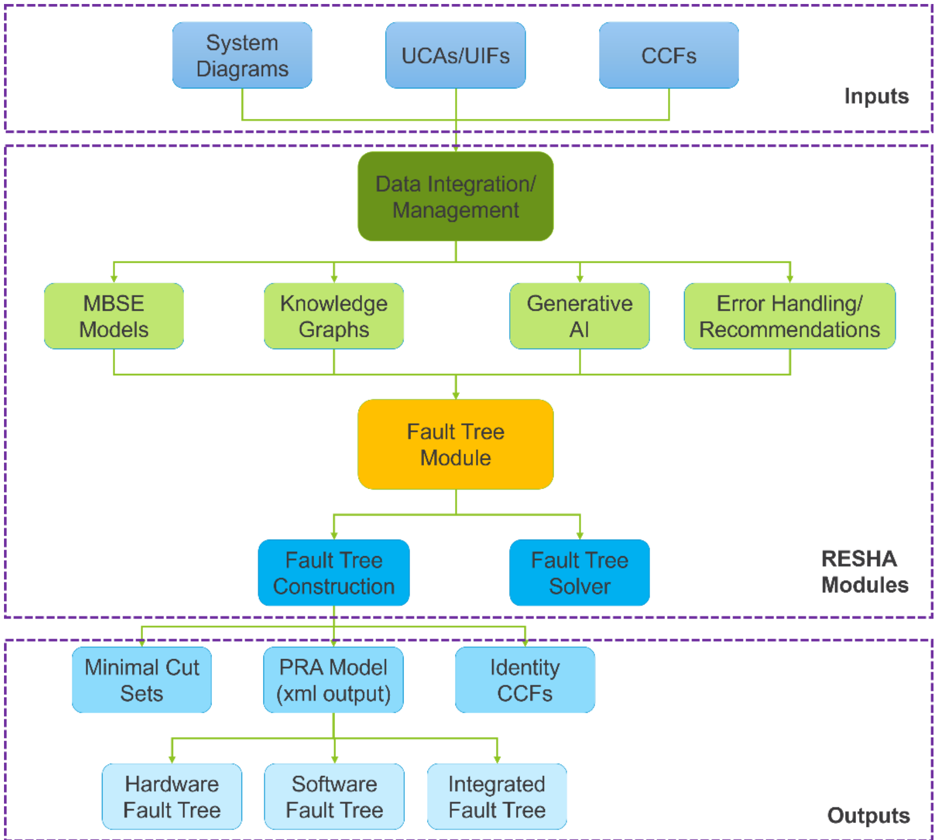


Figure 19. Structure of the FT module for RESHA analysis.

8. CONCLUSIONS AND FUTURE WORKS

8.1 Conclusions

In this FY, we focused on methodology refinement and capability exploration, and the LWRS Program focused on DI&C risk assessment capabilities. The result was improved applicability, usability, and capabilities of the tools developed by INL to support risk assessments by industry. The work encompassed several topics, including incorporating NLP and GAI tools. The risk assessment framework developed at INL consists of four main tools: RESHA (for identifying hazards, with particular emphasis on the identification of potential software CCFs within safety-related DI&C systems), the hybrid CCF model (a parametric CCF model specifically intended for modeling software CCFs), BAHAMAS (a software failure quantification method for use under data-limited conditions such as during early design stages), and ORCAS (a software failure quantification method usable during the testing stages of software development).

This FY saw an expansion made for ORCAS; this work proposed a method to automate the root cause analysis of software defects by incorporating NLP techniques. Lbl2Vec assists in formulating a correlation between software defects and UCAs, thereby relieving the elicitation burden felt by ORCAS developers and users. Given shared concepts between ORCAS and BAHAMAS, NLP offers benefits to both. The results of this effort suggest continued beneficial capabilities and application of NLP to aid in the risk assessment field.

RESHA was selected as a tool that could be enhanced by cutting-edge GAI modeling. It relies on concepts based on STPA. One of the most crucial steps in both analyses is identifying the control structure diagram, as it requires detailed knowledge of the control architecture and the interaction between different components. Such information is documented in various reports, manuals, and literature, and appears in different modalities (i.e., textual descriptions, as well as figures and block diagrams of the control system and its various components). GAI was investigated to support the time-consuming process of identifying the control structure diagram.

A similar investigation was performed regarding the ability of GAI to support D3 analysis, as this too involves various reports that must be reviewed. D3 analysis of DI&C systems is based on different regulatory documents, each providing systematic guidance for D3 evaluation. A case study investigated information retrieval, knowledge abstraction, and the reasoning ability of GPT to support review and decision analyses of DI&C systems from a D3 perspective. The results indicate that GPT can support such efforts; however, it requires systematic guidance and instruction in order to provide the expected outcomes at the required level of detail and accuracy.

This work helped reveal that GPT performance is governed by such factors as the volume and complexity of information used for RAG, the content of prompts (question, contextual information, formatting instructions, etc.), and their sequence. Based on the results of our testing and evaluation, we formulated a set of procedures and guidance to conduct STPA and D3 analysis of the DI&C systems, using multimodal LLMs.

This FY continued efforts to support collaboration with industry in terms of CCF analyses of digital systems. Two industry projects were performed this year. In one, our tools were demonstrated in terms of supporting the hazard identification and analysis specific to potential CCFs. The other entailed continued demonstration and refinement of results obtained previously. These demonstrations have allowed the team to assess the capabilities of the LWRS-developed tools to support industry during different stages of system development. This work is an example of a successful partnership and the research benefits achievable through industry collaboration. It is a goal of the LWRS team to continue such collaborations and provide science- and technology-based solutions and insights that support the long-term operations of the existing nuclear fleet. It is anticipated that the methods presented in this work will continually evolve and become increasingly useful due to these collaborations.

Thanks to the FY 24 efforts and industry collaboration feedback, it was decided that BAHAMAS should be refined to be more user friendly. Thus, it was refined during FY 24, resulting in a more clearly defined theory and structure, in addition to a usable interface that enables more convenient application. A software code was created for BAHAMAS, and details on the inputs and usage of the software are included in this report.

Over the course of this FY, the hybrid CCF model was improved in two ways. First, the model equations where clarified. Second, given that numerous coupling mechanisms can be identified for similar and diverse software within DI&C systems, it is recognized that there is a need for a structured identification of potential CCCGs. This work developed a means of categorizing coupling factors so as to provide a systematic way of identifying potential CCFs. These improvements provide a basis for qualitative and quantitative analyses. This work also serves to further our efforts in preparing for continued industry collaboration.

The tools developed for the risk assessment framework have reached the point where their capabilities should be expanded. As part of the preparation for future work, a plan was made for software tool development. This work laid out a plan for developing the software modules RESHA, BAHAMAS, ORCAS, hybrid CCF modeling, and integrating them all under the SHIELDS umbrella. The future work will emphasize the development of modules for SHIELDS, corresponding to the primary goals of identification, quantification, and evaluation of risks for DI&C systems.

8.2 Future Work

As part of future R&D, project will further improve its tools to better meet the needs of the industry in terms of DI&C reliability and risk analysis, especially for (1) function-based risk assessment of multi-function DI&C systems to continue supporting ongoing collaborations with industry, (2) development of evidence generation and evaluation capabilities to support DI&C safety assurance and design optimization, and (3) development of the software modules for SHIELDS. Ultimately, the team will continue to develop novel approaches to inform risk management and design optimization of DI&C systems intended for the existing light-water reactor fleet.

9. REFERENCES

[1] 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, August 2019.

[2] H. Bao, H. Zhang and T. Shorthhill, "Redundancy-guided System-theoretic Hazard and Reliability Analysis of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants," Idaho National Laboratory, Idaho Falls, ID, August 2020.

[3] H. Bao, T. Shorthill, E. Chen and H. Zhang, "Quantitative Risk Analysis of High Safety-significant Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants using IRADIC Technology," Idaho National Laboratory, Idaho Falls, ID, August 2021.

[4] H. Bao, S. Zhang, R. Youngblood, T. Shorthill, P. Pandit, E. Chen, J. Park, H. Ban, M. Diaconeasa, N. Dinh and S. Lawrence, "Risk Analysis of Various Design Architectures for High Safety-Significant Safety-Related Digital Instrumentation and Control Systems of Nuclear Power Plants During Accident Scenarios," INL/RPT-22-70056, Idaho National laboratory, Idaho Falls, ID, November 2022.

[5] H. Bao, T. Shorthill, E. Chen, J. Park, J. Kim, G. S. Turkmen, H. Ban, N. Dinh, T. Aldemir, S. Zhang and S. Lawrence, "An Integrated Framework for Risk Assessment of Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants: Methodology Refinement and Exploration," INL/RPT-23-74412, Idaho National Laboratory, Idaho Falls, ID, September 2023.

[6] H. Bao, T. Shorthill, E. Chen, J. Park, S. Zhang, A. V. Jayakumar, C. Elks, N. Dinh, H. Ban, H. Zhang, E. Quinn and S. Lawrence, "An Integrated Framework for Risk Assessment of High Safety-significant Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants: Methodology and Demonstration," INL/RPT-22-68656, Idaho National Laboratory, Idaho Falls, ID, August 2022.

[7] 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," *Nuclear Technology*, vol. 208, no. 5, pp. 892-911, 2022.

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

[9] T. Shorthill, H. Bao, H. Zhang and H. Ban, "A Novel Approach for Software Reliability Analysis of Digital Instrumentation and Control Systems in Nuclear Power Plants," *Annals of Nuclear Energy*, vol. 158, 2021.

[10] E. Chen, H. Bao, T. Shorthill, C. Elks and N. Dinh, "Failure Mechanism Traceability and Application in Human System Interface of Nuclear Power Plants using RESHA," Arxiv, 24 May 2022. [Online]. Available: https://arxiv.org/abs/2205.12075. [Accessed 11 August 2022].

[11] S. Zhang, H. Bao, T. Shorthill, J. Park and E. Chen, "Sensitivity and Importance Analyses for Various Design Architectures for High Safety-Significant Safety-Related Digital Instrumentation and Control Systems of Nuclear Power Plants," in *NPIC&HMIT 2023 and PSA 2023 Co-Located Meetings*, Knoxville, TN, July 15 -21, 2023.

[12] H. Bao, T. Shorthill, E. Chen, S. Zhang and S. Lawrence, "Summary of Technical Peer Review on the Risk Assessment Framework proposed in Report INL/RPT-22-68656 for Digital Instrumentation and Control Systems," Idaho National Laboratory, Idaho Falls, ID, INL/RPT-23-71699, March 2023.

[13] E. Chen and T. Shorthill, "Preliminary Demonstration of Risk Assessment Methodologies for Safety Significant Digital Instrumentation and Control Systems," INL/RPT 23-71244, Idaho National Laboratory, Idaho Falls, ID, 2023.

[14] E. Chen, H. Bao and T. Shorthill, "Root Cause Correlation Analysis of Software Failures via Orthogoanl Defect Classification and Natural Language Processing," in *ICAPP*, 2024.

[15] F. Lopes, J. Agnelo, C. A. Teixeira, N. Laranjeiro and J. Bernardino, "Automating orthogonal defect classification using machine learning algorithms," *Future Generation Computer Systems,* vol. 102, pp. 932-947, 2020.

[16] International Business Machines (IBM), "Orthogonal Defect Classification v 5.2 for Software Design and Code," 12 September 2013. [Online]. Available: https://s3.us.cloud-object-storage.appdomain.cloud/res-files/70-ODC-5-2.pdf.

[17] E. Chen, N. Dinh, T. Shorthill, C. Elks, A. V. Jayakumar and H. Bao, "Application of Orthogonal Defect Classification for Software Reliability Analysis," in *Probabilistic Safety Assessment and Management Conference*, Honolulu, 2022.

[18] M. F. Porter, "An algorithm for suffix stripping," *Program,* vol. 14, no. 3, pp. 130-137, 1980.

[19] T. Schopf, D. Braun and F. Matthes, "Web Information Systems and Technologies," in *Semantic Label Representations with Lbl2Vec: A Similarity-Based Approach for Unsupervised Text Classification*, Cham, Springer International Publishing, 2023, pp. 59-73.

[20] N. G. Leveson and J. P. Thomas, STPA Handbook, Massachusetts Institute of Technology, 2018.

[21] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educational and Psychological Measurement*, vol. 20, pp. 37-46, 1960.

[22] International Electrotechnical Commission, "IEC 61508," International Electrotechnical Commission, Geneva, 2010.

[23] R. W. Rolland and R. Schneider, "Lessons Learned in PRA Modeling of Digital Instrumentation and Control Systems," in *Probabilistic Safety Assessment and Management*, Honolulu, 2022.

[24] W. Claes, M. Höst, P. Runeson and A. Wesslén, "Software Reliability," in *Encyclopedia of Physical Science and Technology (Third Edition)*, vol. 15, Academic Press, 2001, pp. 25-39.

[25] S. Basu, Plant Hazard Analysis and Safety Instrumentation Systems, Academic Press, 2016.

[26] T. Tyrväinen, "Probabilistic modelling of common cause failures in digital I&C systems - Literature review, VTT Research Report No. VTT-R-00728-21," VTT Technical Research Centre of Finland, 2021.

[27] T. Chu, G. Martinez-Guridi, M. Yue, J. Lehner and P. Samanta, "Traditional Probabilistic Risk Assessment Methods for Digital Systems," NUREG/CR-6962, U.S. Nuclear Regulatory Commission, 2008.

[28] H. Bao, T. Shorthill, E. Chen, J. Park, S. Zhang, A. V. Jayakumar, C. Elks, N. Dinh, H. Ban, H. Zhang, E. Quinn and S. Lawrence, "An Integrated Framework for Risk Assessment of High Safety-significant Safety-related Digital Instrumentation and Control Systems in Nuclear Power Plants:

Methodology and Demonstration," INL/RPT-22-68656, Idaho National Laboratory, Idaho Falls, ID, August 2022.

[29] 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," *Annals of Nuclear Energy,* vol. 148, p. 107686, 2020.

[30] O. Backstrom, J.-E. Holmberg, M. Jockenhovel-Barttfeld, M. Porthin and A. Taurines, "Software reliability analysis for PSA, NKS-304," Nordic Nuclear Safety Research, Roskilde, Denmark, 2014.

[31] R. Bommasani and et al., "On the Opportunities and Risks of Foundation Models," ArXiv, arXiv:2108.07258, 2021.

[32] OpenAI, "GPT-4 Technical Report," arXiv, arXiv preprint arXiv:2303.08774, 2023.

[33] Meta, "Llama," Meta, [Online]. Available: https://llama.meta.com/. [Accessed 2024].

[34] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel and D. Schuurmans, "Foundation Models for Decision Making: Problems, Methods, and Opportunities," arXiv, arXiv preprint arXiv:2303.04129, 2023.

[35] Symphony AI, "Industrial Copilots," Symphony AI, [Online]. Available: https://www.symphonyai.com/industrial/generative-ai-copilots/. [Accessed 2024].

[36] T. Nguyen, J. Brandstetter, A. Kapoor, J. K. Gupta and A. Grover, "ClimaX: A foundation model for weather and climate," arXiv, arXiv:2301.10343, 2023.

[37] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv, arXiv preprint at arXiv:2312.10997, 2023.

[38] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith and D. C. Schmidt, "A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT," arXiv, arXiv preprit arXiv:2302.11382, 2023.

[39] P. Athe and N. Dinh, "Testing Generative Pre-trained Transformer for Knowledge Abstraction and Reasoning for Nuclear Reactor Design and Safety Applications," in *Probabilistic Safety Assessment and Management (PSAM) Topical, October 23-25, 2023, Virtual*, 2023.

[40] Y. Qi, X. Zhao, S. Khastgir and X. Huang, "Safety Analysis in the Era of Large Language Models: A Case Study of STPA using ChatGPT," arXiv, arXiv preprint arXiv:2304.01246, 2023.

[41] Korea Electric Power Corporation, Korea Hydro & Nuclear Power Co., LTD, "Chapter 7: Instrumentation and Controls. Rev 3," in *APR1400 Design Control Document Tier 2*, Washington, Nuclear Regulatory Commission, 2018.

[42] U.S. NRC, "Guidance for Evaluation of Diversity and Defense in Depth in Digital Computer Based Instrumentation and Control Systems: BTP 7-19, Rev 9," in *Standard Review Plan for the Review of Safety Analysis Reports for Nuclear Power Plants: LWR Edition, NUREG-0800*, Washington, D.C., USA, U.S. Nuclear Regulatory Commission, 2024.

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

[44] R. T. Wood, R. Belles, M. S. Cetiner, D. E. Holcomb, K. Korsah, A. S. Loebl, A. S. Mays, M. D. Muhlheim, J. A. Mullens, W. P. Poore III, A. L. Qualls, T. L. Wilson, Jr. and M. E. Waterman, "Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems," U.S. NRC, Washington, D.C., 2008.

[45] Korea Electric Power Corporation; Korea Hydro & Nuclear Power Co., Ltd, "Diversity and Defense-in-Depth, Rev 0," Korea Electric Power Corporation & Korea Hydro & Nuclear Power Co., Ltd, 2014.

[46] N. Matsumoto, J. Moran, H. Choi, M. E. Hernandez, M. Venkatesan, P. Wang and J. H. Moore, "KRAGEN: a knowledge graph-enhanced RAG framework for biomedical problem solving using large language models," *Bioinformatics,* vol. 40, no. 6, 2024.

[47] C. Edwards, "Hybrid Context Retrieval Augmented Generation Pipeline: LLM-Augmented Knowledge Graphs and Vector Database for Accreditation Reporting Assistance," arXiv, arXiv preprint arXiv:2405.15436, 2024.

[48] "Top 8 software Development Life Cycle (SDLC) Models use in Industry," Geeks for Geeks, 19 June 2024. [Online]. Available: https://www.geeksforgeeks.org/top-8-software-development-models-used-in-industry/?ref=.

[49] "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.

[50] M. Muhlheim and R. Wood, "Technical Basis for Evaluating Software-Related Common-Cause Failures," Oak Ridge National Laboratory, Oak Ridge, 2016.

[51] K. Qin, B. Yang, Y. Liu, H. Liu, Y. Qian and K. Li, "Uncertainty quantification and target accuracy assessment of nuclear data to effective neutron multiplication factor of heat pipe cooled reactor," *Annals of Nuclear Energy,* vol. 190, 2023.

[52] International Council on Systems Engineering (INCOSE), Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin and T. M. Shortell, Eds., Hoboken, New Jersey, USA: John Wiley & Sons, 2015.

[53] ISO/IEC/IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle management -- Part 1: Guidelines for life cycel management," in *ISO/IEC/IEEE 24748-1:2018(E)*, 2018.

[54] 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.

[55] R. Chillarege and K. R. Prasad, "Test and Development Process Retrospective - a Case Study using ODC Triggers," in *International Conference on Dependable Systems and Networks*, 2002.

[56] M. Butcher, H. Munro and T. Kratschmer, "Improving software testing via ODC: Three Case Studies," *IBM Systems Journal,* vol. 41, no. 1, pp. 31-44, 2002.

[57] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray and M.-Y. Wong, "Orthogonal Defect Classification - A Concept for In-Process Measurements," *International Institute of Electrical Engineers Transactions on Software Engineering,* vol. 18, no. 11, pp. 943-956, 1992.

[58] J. Agnelo, N. Naranjeiro and J. Bernardino, "NoSQL odc dataset, results, and support code," March 2019. [Online]. Available: https://eden.dei.uc.pt/~cnl/papers/2019-jss.zip. [Accessed August 2022].

[59] J. Agnelo, N. Laranjeiro and J. Bernardino, "Using Orthogonal Defect Classification to Characterize NoSQL Database Defects," *The Journal of Systems and Software,* vol. 159, 2020.

[60] H. Dezfuli, D. Kelly, C. Smith, K. Vedros and W. Galyean, "Bayesian Inference for NASA Probabilistic Risk and Reliability Analysis," National Aeronautics and Space Administration, NASA/SP-2009-569, 2009.

[61] 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.

[62] R. A. Humphreys, "Assigning a Numerical Value to the Beta Factor Common Cause Evaluation," in *Reliability '87*, 1987.

[63] U. B. Kjaerulff and A. L. Madsen, Bayesian Networks and Influence Diagrams: A Guide to Construction and Analysis, 2nd ed., New York: Springer, 2013.

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

[65] Mathworks, "Matlab," 2019. [Online]. Available: https://www.mathworks.com/products/matlab.html. [Accessed 2024].

[66] E. Chen, L. Lin and N. Dinh, "Advanced Transient Diagnostic wwith Ensemble Digital Twin Modeling," in *American Nuclear Society Mathematics and Computation Conference*, Virtual, 2021.

[67] Z. Xu and J. H. Saleh, "Machine Learning for Reliability Engineering and Safety Applications: Review of Current Status and Future Opportunities," *Reliability Engineering & System Safety,* vol. 211, 2021.

[68] Y. Zhang, C. Zhang, S. Wang, H. Dui and R. Chen, "Health indicators for remaining useful life prediction of complex systems based on long short-term memory network and improved particle filter," *Reliability Engineering and System Safety,* vol. 241, 2024.

[69] V. Agarwal, C. Walker, K. Manjunatha, T. Mortenson, N. Lybeck and A. Gribok, "Technical Basis for Advanced Artificial Intelligence and Machine Learning Adoption in Nuclear Power Plants," Idaho National Laboratory, Idaho Falls, 2022.

[70] A. Theissler, J. Perez-Velazquez, M. Kettelgerdes and G. Elger, "Predictive Maintenance enabled by machine learning: Use cases and challenges in the automotive industry," *Reliability Engineering & System Safety,* vol. 215, 2021.

[71] M. Mikhail, M.-S. Ouali and S. Yacout, "A data-driven methodology with a nonparametric reliability method for optimal condition-based maintenance strategies," *Reliability Engineering & System Safety,* vol. 241, 2024.

[72] L. Lin, P. Athe, P. Rouxelin, M. Avramova, A. Gupta, R. Youngblood, J. Lane and N. Dinh, "Development and assessment of a nearly autonomous management and control system for advanced reactors," *Annals of Nuclear Energy,* vol. 150, no. 1, p. 107861, 2021.

[73] J. Xia, R. Huang, Z. Chen, G. He and W. Li, "A novel digital twin-driven approach based on physical-virtual data fusion for gearbox fault diagnosis," *Reliability Engineering & System Safety,* vol. 240, 2023.

[74] H.-H. Gu, R.-Z. Wang, M.-J. Tang, X.-C. Zhang and S.-T. Tu, "Data-physics-model based fatigue reliability assessment methodology for high-temperature components and its application in steam turbine rotor," *Reliability Engineering & System Safety,* vol. 241, 2024.

[75] T. Hastie, R. Tibshirani and J. Friedman, The Elements of Statiscal Learning 2nd Edition: Data Mining, Inference, and Prediction, Springer Series in Statistics, 2017.

[76] K. Z. Y. L. Z. L. Jingkang Yang, "Generalized Out-of-Distribution Detection: A Survey," 2021. [Online]. Available: https://arxiv.org/abs/2110.11334. [Accessed 29 April 2022].

[77] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum and B. Katz, "ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," in *Proceedings of the Conference on Neural Information Processing Systems*, Vancouver, 2019.

[78] S. Wang and Z. Li, "Exploring the mechanism of crashes with automated vehicles using statistical modeling apporaches," *Public Library of Science,* 2019.

[79] R. Roelofs, S. Fridovich-Keil, J. Miller, V. Shankar, M. Hardt, B. Recht and L. Schmidt, "A Meta-Analysis of Overfitting in Machine Learning," in *Proceedings of the Conference on Neural Information Processing Systems*, Vancouver, 2019.

[80] P. Ortega and V. Maini, "Building safe artificial intelligence: specification, robustness, and assurance," DeepMind Safety Research Blog, 2018.

[81] D. Manheim and S. Garrabrant, "Categorizing Variants of Goodhart's Law," Cornell University, arXiv, 2019. [Online]. Available: https://arxiv.org/abs/1803.04585.

[82] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt and D. Song, "Natural Adversarial Examples," 2021. [Online]. Available: https://arxiv.org/abs/1907.07174.

[83] National Institute of Standards and Technology, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," Department of Commerce, Washington, 2023.

[84] M. Dennis, T. Lalain, L. Betancourt, A. Hathaway and R. Anzalone, "Artificial Intelligence Strategic Plan," U.S. Nuclear Regulatory Commission, Washington, 2022.

[85] C. Lessmeier, J. K. Kimotho, D. Zimmer and W. Sextro, "Condition Monitoring of Bearing Damage in Electromechanical Drive Systems by Using Motor Current Signals of Electric Motors: A Benchmark Data Set for Data-Driven Classification," *PHM Society European Conference,* vol. 3, no. 1, 2016.

[86] J. W. Lane, J. M. Link, J. M. King, T. L. George and S. W. Claybrook, "Benchmark of GOTHIC to EBR-II SHRT-17 and SHRT-45R Tests," *Nuclear Technology,* vol. 206, no. 7, pp. 1019-1035, 2020.

[87] M. Abramovitz and A. I. Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, Washington: United States Department of Commerce, 1964.

[88] S. J. Sheather and M. C. Jones, "A reliable data-based bandwidth selection method for kernel density estimation," *Journal of the Royal Statistical Society, Series B,* vol. 53, pp. 683-690, 1991.

[89] B. U. Park and J. S. Marron, "Comparison of data-driven bandwidth selectors," *Journal of the American Statistical Association,* vol. 85, no. 409, pp. 66-72, 1990.

[90] R. P. Duin, "On the Choice of Smoothing Parameters for Parzen Estimators of Probability Density Functions," *IEEE Transactions on Computers,* vol. 25, no. 11, pp. 1175-1179, 1976.

[91] L. Lin, H. Bao and N. T. Dinh, "Uncertainty Quantification and Software Risk Analysis for Digital Twins in the Nearly Autonomous Management and Control Systems: A Review," *Annals of Nuclear Energy,* vol. 160, 2021.

[92] V. Yadav, V. Agarwal, P. Jain, P. Ramuhalli, X. Zhao, C. Ulmer, J. Carlson, D. Eskins, C. Nelli, J. Matrachisia, J. Bass, B. Cohen and R. Iyengar, "State-of-Technology and Technical Challenges in Advanced Sensors, Instrumentation, and Communication to Support Digital Twin for Nuclear Energy Application, INL/RPT-23-70853," Nuclear Regulatory Commission, Washington, 2023.

[93] Y. Xu, S. Kohtz, J. Boakye, P. Gardoni and P. Wang, "Physics-informed machine learning for reliability and systems safety applications: state of the art and challenges," *Reliability Engineering & System Safety,* vol. 230, 2023.

[94] J. Fernandez, J. Chiachio, J. Barros, M. Chiachio and C. Kulkarni, "Physics-guided recurrent neural network trained with approximate Bayesian computation: A case study on structual response prognostics," *Reliability Engineering & System Safety,* vol. 243, 2024.

[95] A. S. Iskhakov, N. Dinh and E. Chen, "Integration of neural networks with numerical solution of PDEs for closure models development," *Physics Letters A,* vol. 406, 2021.

[96] International Standards Organization, "ISO/IEC TS 5723:2022(en) Trustworthiness - Vocabulary," International Standards Organization, 2022. [Online]. Available: https://www.iso.org/obp/ui/en/#iso:std:iso-iec:ts:5723:ed-1:v1:en. [Accessed 2024].

[97] National Institute of Standards and Technology, "Failure (or hazard) rate," in *e-Handbook of Statistical Methods,* National Institute of Standards and Technology, 2012.

[98] U.S. Nuclear Regulatory Commission, "Glossary of Risk-Related Terms in Support of Risk-Informed Decisionmaking," U.S. Nuclear Regulatory Commission, Washington, 2013.

[99] K. S. Wang, F. S. Hsu and P. P. Liu, "Modeling the bathub shape hazard rate function in terms of reliability," *Reliability Engineering and System Safety,* vol. 75, pp. 397-406, 2002.

Appendix A
BAHAMAS Theory

Page intentionally left blank

APPENDIX A – BAHAMAS THEORY

BAHAMAS is a tool within the LWRS framework to quantify software failure probability. BAHAMAS was developed for cases when limited testing or operational data are available, such as for reliability estimations of software in early development stages[a] [5]. BAHAMAS was created based on the notion that software failure occurs when a latent defect is activated by use or operation of the software. Defects (e.g., coding errors, installation errors, maintenance errors, setpoint changes, and requirements errors) may be introduced into the software code by human errors at any stage of the software development life cycle. Activation of these defects results from certain operational conditions [50]. BAHAMAS combines these factors into a network that traces software failure to root human causes. Instead of relying on testing data, BAHAMAS employs a BBN to map the causes of software failures to specific defect types. Figure A-1 shows the general concepts employed within BAHAMAS, as well as the general structure of the BBN.

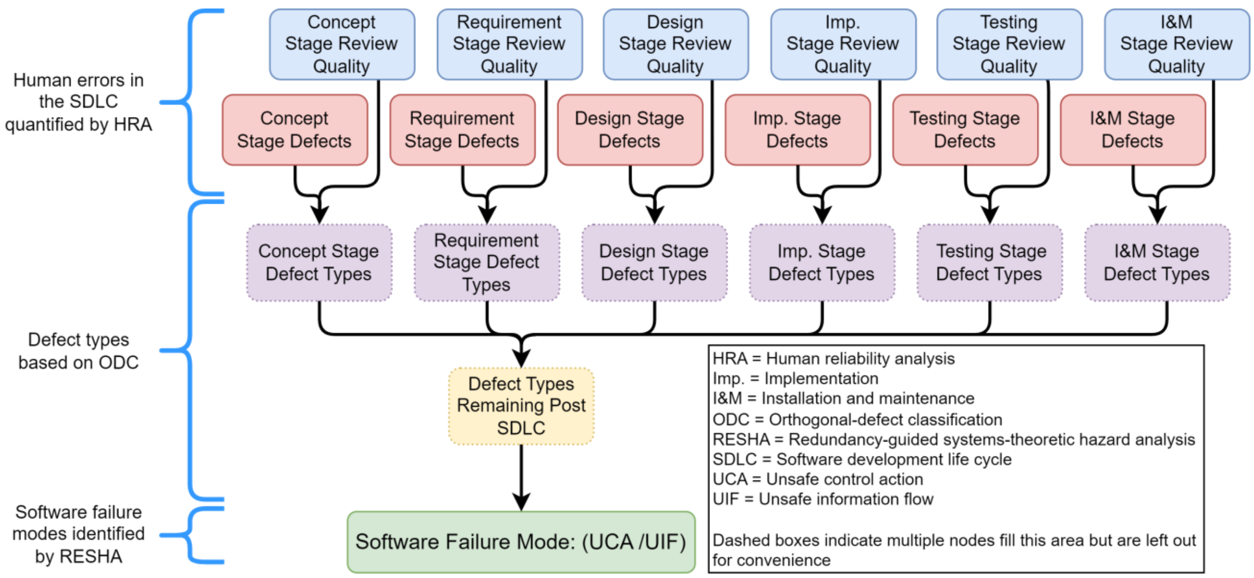


Figure A-1. General structure of BAHAMAS.

BAHAMAS evaluates software failures by tracking defect introduction and defect removal activities, their impact on the types of defects that may remain within software, and ultimately the probability of software failure. Figure A-1 shows the general form of BAHAMAS. The network consists of the following nodes:

- Red nodes: Account for the introduction of defects by considering human errors at each stage of the software development life cycle.
- Blue nodes: Account for the defect removal activities employed during the software development life cycle.
- Purple nodes: Account for the types of defects that remain in the software at a given stage of the SDLC.
- Yellow node: Accounts for the types of defects that remain in the software after all stages of the SDLC.

---

[a] Note that in this work stage and phase are used synonymously in reference to the software development life cycle.

- Green node: Accounts for the software failure probability, based on the defects that remain within the software after the SDLC.

In total, BAHAMAS employs the following concepts. When accounting for defect introduction and removal, specific types of defects will remain in the software. Depending on the types of defects that remain, the software will exhibit certain failure modes. The next sections detail each of these concepts and how they are accounted for by BAHAMAS.

A-1 Defect Introduction

Defects or faults arise within software as a result of human errors in the SDLC activities (e.g., design and requirements specification, verification, validation, etc.). These faults are the main reason for unwanted software behavior [51]. To account for defect introduction, BAHAMAS considers human tasks of SDLC, which are commonly divided into stages or phases. Six common stages are employed by BAHAMAS (see Figure A-1), as based on [52] [53] [49]. These stages represent root nodes of the network. Each stage has two states: defects or no defects. For quantification of the root nodes, these states are given through HRA. Numerous HRA methods exist, but none have been readily identified as being specifically developed for human error prediction for software development activities. Consequently, this remains a topic for future research.

Each SDLC stage has multiple activities consisting of multiple tasks. Each of these activities may introduce errors to the software. HRA is applied to determine the human error probability (HEP) of a subset of human activities (i.e. relevant tasks)—specifically those pertaining to development, creative, problem-solving, or other such activities. Review activities are credited separately. The union of human errors for the relevant tasks (*T*) serves to indicate the probability of defects for each stage (see Equation (A.1)below).

$$P(Stage\ Defects) = \sum_{i=1}^{T} (HEP)_i \tag{A.1}$$

Human Error Modes

This work assumes that THERP can be extended to address human errors in the SDLC by assuming that the SDLC errors can be classified as diagnostic, commission, or omission errors. THERP has guidance for scoring certain activities closely related to these three categories. Ultimately, it is these HEPs and the combinations thereof that will be employed by BAHAMAS. The various combinations of diagnosis, omission, and commission were evaluated through Monte Carlo sampling. The results are the distributions used for assessing each SDLC task. The distributions are given below in Table A-3. Each SDLC task is assessed for the dominant combination of human error modes. Sampling is performed to capture uncertainty.

Table A-1. Diagnosis- or understanding-type errors (Diagnosis 1).

Performer	Group consisting of stakeholders, engineers, and managers
Background	The process of defining a project is similar to assessing or diagnosing a new situation. There may be similarities from past experiences, but ultimately the scenario represents a new problem that must be solved. THERP was designed for assessing activities within the context of a nuclear power plant; therefore, its application must fit similar conditions. THERP prescribes HEPs for group activities associated with diagnoses of abnormal events. The assumption is made that the diagnosis of an abnormal event is akin to solving a new problem. Experience, training, and expertise play a role in the diagnosis. But ultimately, the effort is not strictly routine. Hence, because each project is unique, we shall assume this is equivalent to diagnosing an abnormal event.
Error source	Faulty diagnosis

Guidance	Diagnosis Error: Table 20-3 in [54] provides guidance diagnostic actions made by a team or group. Item 5 corresponds to 1 hour for time to diagnose. This amount of time is assumed sufficient for enabling a group of qualified individuals to define the purpose of the project. The HEP is not modified because the adjustment rules (Table 12-5 in [54]) do not apply well to the SDLC activities.
HEP	Lognormal distribution. Median 1×10-4, EF 30.
Note: Diagnosis-1 will be used for all general diagnosis- or understanding-type errors.	

Table A-2. Diagnosis- or understanding-type errors (Diagnosis 2).

Performer	Group consisting of stakeholders, engineers, and managers
Background	Same as previous, but extended to represent less stress, more time, or, equivalently, to represent a simpler diagnostic event. This has the effect of shifting the median of the distribution from 1×10-4 to 1×10-5.
Error source	Faulty diagnosis
Guidance	Diagnosis Error: Table 20-3 in [54] provides guidance diagnostic actions made by a team or group. Table 20-3, item 6 corresponds to a diagnostic action taking approximately 24 hours. This duration to perform a diagnosis is reasonable; however, reducing corresponding HEP to account for additional time may produce an unrealistic decrease in HEP. The HEP is not modified because the adjustment rules (Table 12-5 in [54]) do not apply well to the SDLC activities.
HEP	Lognormal distribution. Median 1×10-5, EF 30.
Note: Diagnois-2 is applied for less complex tasks (i.e., pre-defined, well-understood tasks)	

Table A-3. Errors of omission and errors of commission.

Performer	Group consisting of stakeholders, engineers, and managers
Background	THERP does not provide guidance for the creation of technical documents, code, etc. This process may involve omitting or committing errors.
Error source	Something or some things may be overlooked or left out when completing a task. At other times, wrong or incorrect details or actions may arise.
Guidance	THERP indicates when an appropriate HEP is unavailable, the nominal value of 3×10-3 is assigned for errors of omission or for errors of commission (see page 20-13 in [54]). It is an assumption of the case study that, for events not clearly omission- or commission-dominant, a union of the two will be used. This value can be approximated using the rare event approximation or by assuming that A and B are independent. Ultimately, the result is given as P(A or B) = P(A)+P(B)= 6×10-3. Table 20-20 in [54] provides the error factor (EF) (e.g., All EF values for HEP>.001 are 5 for activities that are not strictly performed step-by-step.)
HEPs	Omission only (O): Lognormal distribution. Median 3×10-3, EF 5. Commission only (C): Lognormal distribution. Median 3×10-3, EF 5. Omission or commission possible (O-C): Lognormal distribution. Median 6×10-3, EF 5.

Table A-4. Error mode distributions.

Key	Description	Parameter1 “mu”	Parameter “sigma”
D1	Diagnosis error (Diagnosis-1)	-9.21034	2.0676
D2	Simple diagnosis error (Diagnosis-2)	-11.5129	2.0676
C	Omission error2	-5.80914	0.978382
O	Commission error	-5.80914	0.978382
OC	Omission and Commission errors	-5.116	0.978382
D1C	Diagnosis-1 and Omission	-5.63215	0.94217

D1O	Diagnosis-1 and Commission	-5.63215	0.94217
D1OC	Diagnosis-1, Omission and Commission	-5.00712	0.942896
D2C	Diagnosis-2 and Omission	-5.77788	0.960271
D2O	Diagnosis-2 and Commission	-5.77788	0.960271
D2OC	Diagnosis-2, Omission and Commission	-5.09867	0.966776
Note 1: "mu" and "sigma" are lognormal distribution model parameters. Note 2: "C" and "O" have the same error mode assigned. This results in D1O and D1C having the same distribution (likewise for D2O and D2C). Future work may modify these distributions.			

A-2. Defect Removal

Verification and review activities are employed during software development to ensure that the software is reliable. The quality of these activities influences defect removal and overall software quality. ODC (see [16]) indicates that reviews performed during the software life cycle can help identify and remove defects. Depending on the review, specific concepts, or triggers, are employed and may bring out or allow defects to surface [55]. Table A-5 shows the typical mapping of the triggers and review activities [16]. Our work adopts this mapping for convenience. High-quality review activities should employ these triggers during each stage of the SDLC, thereby reducing the probability of defects remaining within the software (as demonstrated in [56]). On this basis, trigger coverage has been selected as a metric for defect removal. The stage-level trigger coverage (TC), given by Equation (A.2), is determined from the average of each task-level trigger coverage (tc), which is the percent of relevant triggers that have been covered for a task in a particular SDLC stage. Note that T is the total number of relevant tasks for a given stage.

$$TC = \frac{1}{T}\sum_{i=1}^{T}(tc)_i \tag{A.2}$$

In addition to trigger coverage, the average number of reviews for a given SDLC stage is also considered as a metric for defect removal. When tasks are reviewed or checked, they are less likely to fail. Checkers are considered in HRA analysis (e.g., THERP) and represent recovery paths for a given activity [54]. The average number of reviews for a particular stage is defined by Equation (A.3), where r is the average number of reviews performed for each relevant task and T is the total number of relevant tasks for a given stage. As an example, for a given task some triggers may be reviewed or checked multiple times, whereas others are not. The average of these reviews represents r.

$$R = \frac{1}{T}\sum_{i=1}^{T} r_i \tag{A.3}$$

Table A-5. ODC triggers.

Review Activities	Triggers
For use during review activities associated with concept, design, and implementation stages of the software development life cycle.	Design Conformance
	Logic/Flow
	Backward Compatibility
	Lateral Compatibility
	Concurrency
	Internal Document
	Language Dependency
	Side Effect
	Rare Situations
	Simple Path

For use during review activities associated with testing stages of the software development life cycle.	Complex Path
	Test Coverage
	Test Variation
	Test Sequencing
	Test Interaction
	Workload/Stress
	Recovery/Exception
	Startup/Restart
	Hardware Configuration
	Software Configuration
	Blocked Test (Previously Normal Mode)
Descriptions for each trigger can be found in 2013 IBM "Orthogonal Defect Classification v 5.2 for Software Design & Code" [16].	

A-3. Defect Types

During software development, defects or bugs arise, and their concentrations change throughout development as they are influenced by the conducted review activities [57]. The ODC methodology has developed a scheme to categorize software defects into eight orthogonal classes (i.e., types). This work relies on the classification scheme to help predict software failure probability. Historical ODC data (from [57], [58], and [59]) provides a collection of defects and the activities (and triggers) used to find them. We mapped these activities to SDLC stages for a convenient view of defect types for a given stage. ODC design review activities are mapped to the Concept and Requirements stages. ODC code inspection activities are mapped to the Design and Implementation stages. And ODC testing activities are mapped to the Testing and the Installation & Maintenance stages. Bayesian inference was applied to develop distributions for each defect type at each stage of the SDLC. This was done by taking a binomial model, Jeffreys noninformative prior, and obtaining a posterior beta distribution to represent the general expected defect types for each stage. The posterior distribution is given as a beta with parameters $\alpha + x$ and $\beta + y - x$, where x is the number of observed events, y is the number of trials, and α = 0.5 and β = 0.5 for Jeffreys noninformative prior [60] [61]. Distributions can be produced readily from the data in Table A-6.

Table A-6. ODC defect type data for each review activity.

	Defect types from historical data*							
Typical categories of review activities	Al.	As.	Ch.	Doc.	Fun.	Int.	Rel.	Tim.
Design review	56	26	26	107	90	57	0	13
Code inspection	1215	153	240	58	372	221	8	25
Testing	1277	181	269	12	245	361	91	28
*Datasets from [57], [58], and [59]. The datasets contained no relationship defects for design review.								

The classification of defects serves as a basis for an equation that can predict distributions of defects for each stage of software development. Depending on the defect removal efforts, there is an expectation that the probability of defects remaining will be influenced. As expressed by Humphreys [62], design improvements can be considered a function (likely nonlinear) of effort and time. A model was developed for this work to express the conditional probability of specific defect types remaining given the state defect introduction and removal activities. Equation (A.4) is the defect conditional probability (DCP) equation, where G is the general expected probability for a defect type for a given SDLC stage (based on the posterior distributions obtained Table A-6), TC is the stage-level trigger coverage and is determined from Equation (A.2), and R is the average number of reviews performed for each task of the SDLC stage, as defined by Equation (A.3).

$$DCP = \begin{cases} StageReview(all\ states)\ and\ StageDefect\ (yes), & DCP = Ge^{-4TCR} \\ StageReview(all\ states)\ and\ StageDefect(no), & DCP = 0 \end{cases} \tag{A.4}$$

A-4. Failure Modes

Software has different modes of failure. The conditional probability of software failure given the existence of certain defect types has been discussed and developed in previous work by INL (see [5]). Data from [58] and [59] were assessed in terms of how they impacted the usability, functionality, and performance from an end-user perspective, and were assigned a UCA/UIF category (equivalently, a mode of failure). The result of the assessment provided conditional relationships shown in Table A-7. Specifically, Table A-7, which provides a global perspective on the relationship between defect types and UCA/UIF failure modes [5]. As further classifications are performed, these distributions can be continually improved. These correlations are used by Equation (A.5), where I is the total number of defect types considered by BAHAMAS and UC_x is the UCA/UIF failure mode being evaluated. Table A-7 provides the conditional probabilities used for $P(UC_x|Type_i)$:

$$P(software\ failure\ of\ mode\ x) = \sum_{i=1}^{I} P(UC_x|Type_i)P(Type_i) \tag{A.5}$$

An inherent assumption in this work is that software failure results from active defects. An implication of Equation (A.5) is that the defect activation probability equals one, and that the activation of defects is mutually exclusive (i.e., multiple defects are not activated simultaneously). In other words, the probability of software failure is a summation of the contribution of each defect type. Future work may investigate refinements to this activation probability.

Table A-7. UCA/UIF defect type correlation table (early 2024).

Defect Class	UCA/UIF – A	UCA/UIF – B	UCA/UIF – C	UCA/UIF – D
Algorithm	0.217 ± 0.051	0.525 ± 0.068	0.124 ± 0.044	0.134 ± 0.019
Assignment	0.288 ± 0.152	0.667 ± 0.130	0.045 ± 0.072	N/A
Checking	0.219 ± 0.074	0.539 ± 0.115	0.102 ± 0.072	0.141 ± 0.129
Documentation	0.250 ± 0.250	0.250 ± 0.250	0.250 ± 0.250	0.250 ± 0.250
Function	0.250 ± 0.154	0.518 ± 0.064	0.157 ± 0.092	0.074 ± 0.216
Interface	0.262 ± 0.065	0.579 ± 0.086	0.093 ± 0.095	0.065 ± 0.039
Relationship	0.250 ± 0.250	0.250 ± 0.250	0.250 ± 0.250	0.250 ± 0.250
Timing	0.095 ± 0.334	0.190 ± 0.289	0.524 ± 0.423	0.190 ± 0.289
Note: This table is based on [5]. These distributions may change as more data is collected and methods for data collection improve. As demonstrated by ORCAS refinement in Section 2 of the current report.				

A-5. Main Calculation

Bayesian networks consist of variables, also called nodes, that represent events of interest for a given domain and are classified as problem variables, information variables, and mediating variables [63]. Figure A-2 provides a generalized structure of a Bayesian network, with the types of variables labeled. The details on these variables are as follows:

- **Problem variables** (sometimes called hypothesis variables) are variables that typically cannot be measured or observed directly.

- **Information variables** are variables about which observations, measurements, etc., can be made. Information variables include:
    - **Background variables** provide information or prior knowledge pertaining to the problem.
    - **Symptom variables** provide information that serves as a type of evidence of the problem variable.
- **Mediating variables** are variables that are not directly observed but are useful for building the BBN. They are often used through a process called "divorcing," which is a modeling technique that reduces the burden of assessing the conditional probability between parent and child nodes [63].

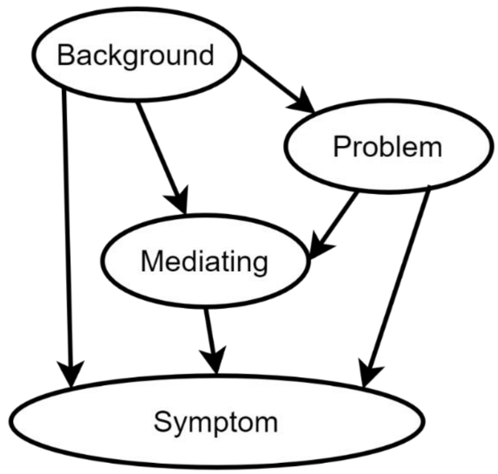


Figure A-2. Generalized Bayesian network with labeled node types, based on [63].

For BAHAMAS, software failure is a variable of interest—or "problem variable," according to the descriptions above. A primary motivation of this work is to estimate software failure from limited data. Data observed through testing fulfill the role of a symptom variable; without data, estimating or predicting software failure becomes the task of a reduced network. For BAHAMAS, the network is reduced to consist only of background and problem nodes. Intermediate or mediating nodes are employed to help evaluate software failure probabilities. Reducing the network for limited data also alters the type of calculations needed. If there are no symptoms, no observations will be made, and the network will remain at a level that will rely on prediction type calculations to estimate the probability of software failure.

Software failure probability (i.e., the green-colored node from Figure A-1) is determined by evaluating the BBN, which involves calculating the marginal probability of a child node given the state of its parents. The marginal probability represents the probability of a specific state within the joint probability of all other states [64]. This calculation proceeds in the direction of the arrows shown in Figure A-1 and is repeated until the final node's marginal probability is determined. Equation (A.6) shows the marginal probability of a specific state (i) of a child node (i.e., c) with two parents (i.e., a, b), within a BBN. The parents have (n) and (m) total states.

$$P(c_i) = \sum_{j=1}^{n}\sum_{k=1}^{m} P\left(c_i | a_j, b_k\right) P(a_j) P(b_k) \tag{A.6}$$

The conditional probability of a child state, given the state of its parents (i.e., $P\left(c_i | a_j, b_k\right)$ of Equation (A.6)changes depending on which of the nodes shown in Figure A-1 are being assessed.

The main calculation proceeds by applying Equation (A.6) to determine the specific software defect types for each stage (i.e. the purple nodes shown in Figure A-1). $Type_{ij}$ is defect type (i) remaining from stage (j). The probability of a specific defect type remaining for a particular stage is given by Equation (A.11), where (a) represents human errors, (b) represents review, and (c) represent a specific defect type for a given stage. Each has two states (i.e., 1 = yes; 2 =no). The expansion of Equation (A.11) is as follows:

$$= P(c_i|a_1, b_1)P(a_1)P(b_1) + P(c_i|a_2, b_1)P(a_2)P(b_1) + P(c_i|a_1, b_2)P(a_1)P(b_2) + P(c_i|a_2, b_2)P(a_2)P(b_2) \quad \text{(A.7)}$$

This expansion can be reduced. The conditional probability of a specific defect remaining is zero when there are no human errors for a given stage. Thus, $P(c_i|a_2, b_k) = 0$ when $P(a_2) \geq 0$.

$$= P(c_i|a_1, b_1)P(a_1)P(b_1) + P(c_i|a_1, b_2)P(a_1)P(b_2) \quad \text{(A.8)}$$

Further reduction is achieved by recognizing that review either occurs or it does not, with a probability of zero or one (i.e., if $P(b_1) = 1$, then $P(b_1) = 0$). If review is performed, then $P(b_1) = 1$ and $P(b_2) = 0$. Equation (A.8) reduces to:

$$= P(c_i|a_1, b_1)P(a_1) \quad \text{(A.9)}$$

If no review is performed, then $P(b_1) = 0$ and $P(b_2) = 1$. Equation (A.8) reduces to:

$$= P(c_i|a_1, b_2)P(a_1) \quad \text{(A.10)}$$

Thus, the probability of a specific defect type remaining for a particular stage of the SDLC is given by Equation (A.11), which is a reduced form of the marginal probability, where $P(D_j)$ is the probability of defects introduced (i.e., human error, given by Equation (A.1).

$$P(Type_{ij} = True) = P(Type_{ij}|D_j, R_j)P(D_j) \quad \text{(A.11)}$$

$P(Type_{ij}|D_j, R_j)$ is the conditional probability of a specific defect type given the human errors and review activities, and is determined via Equation (A.12):

$$DCP = \begin{cases} StageReview(all\ states)\ and\ StageDefect\ (yes), & DCP = Ge^{-4TCR} \\ StageReview(all\ states)\ and\ StageDefect(no), & DCP = 0 \end{cases} \quad \text{(A.12)}$$

Equation (A.12) is the defect conditional probability equation, where G is the general expected probability for a defect type for a given SDLC stage, TC is the stage-level trigger coverage from step 4, and R is the average number of reviews performed for each task of the SDLC stage.

Next, the marginal probability equation is likewise employed for the yellow-colored node in Figure A-1. This determines each defect $Type_i$ remaining after all six stages (e.g., a, b, c, d, e, f) of the SDLC (see Equation (A.13). The conditional probability $P(Type_i|a_k, b_l, c_m, d_n, e_p, f_q) = 1$ when the probability of stages defect types existing exceeds zero, $P(Type_{ij}) > 0$, otherwise, $P(Type_i|a_k, b_l, c_m, d_n, e_p, f_q) = 0$.

$$P(Type_j = True) = \sum_{k=1}^{2}\sum_{l=1}^{2}\sum_{m=1}^{2}\sum_{n=1}^{2}\sum_{p=1}^{2}\sum_{q=1}^{2} P(G_j|a_k, b_l, c_m, d_n, e_p, f_q)P(a_k)P(b_l)P(c_m)P(d_n)P(e_p)P(f_q) \quad \text{(A.13)}$$

Finally, the software failure probability is given by Equation (A.14). I is the total number of defect types considered by BAHAMAS, and UCA_x is the failure mode being evaluated. Table A-7 provides the conditional probabilities used for $P(UCA_x|Type_i)$.

$$P(UCA_x) = \sum_{i=1}^{I} P(UCA_x|Type_i)P(Type_i) \tag{A.14}$$

An inherent assumption in this work is that software failure results from active defects. An implication of Equation (A.14) is that defect activation probability equals one and that the activation of defects is mutually exclusive (i.e., multiple defects are not activated simultaneously). In other words, the probability of software failure is a summation of the contribution of each defect type. Future work may investigate refinements to this activation probability.

A-6. Uncertainty Considerations

This work incorporates uncertainty by employing distributions for the inputs and iterating on the calculation. The result is a distribution for software failure probability. The following is a discussion on uncertainty sources.

Uncertainty in defect introduction:

- Uncertainty is captured by using distributions rather than point estimates for HEP during software development tasks. Statistical sampling is used to evaluate the uncertainty for each task so as to develop a distribution for each stage of the SDLC.

Uncertainty in defect removal:

- The number of reviews can be directly measured to eliminate uncertainty from this source.
- Trigger coverage can be directly measured to eliminate uncertainty from this source.
- When it is impractical to directly measure the review number and trigger coverage, expert judgement can be employed. In such cases, distributions for the average number of reviews and average trigger coverage per stage of the SDLC can be developed to account for the uncertainty of defect removal.

Uncertainty in defect types remaining:

- Uncertainty arises from the selected exponential model used for (A.12). An alternative model could be selected given insights from future research. The influence of this uncertainty is not captured in the results.
- Equation (A.12) uses a general expected distribution for defect types at each stage of the SDLC. The uncertainty of this parameter is captured by the distribution created and discussed in Section 2.4. Sampling from these distributions enables uncertainty to be captured in the results.

Uncertainty in the failure modes:

- This uncertainty is captured by the uncertainty given in Table A-7. Future work may lead to an alternate distribution to reflect these uncertainties.

Unique conditions are enforced for the Monte Carlo sampling scheme employed by BAHAMAS. First, the sum of defect type probabilities must equal 1. Second, the sum of UCA probabilities must also equal 1. The sampling algorithm employed by BAHAMAS ensures that these conditions are met. It is recognized that these conditions will naturally modify the sample distributions, causing them to be similar but imperfect representations of the parent distributions from which they were sampled.

A-7. Consideration for CCF Modeling

In previous work, we investigated CCFs of redundant components that share the same software [5]. For a CCF, there must be (1) a failure involving multiple components and (2) a common cause that is made “shareable” by the existence of some coupling mechanism. The shared cause for software must include a common “active” defect or fault. A CCF can occur when multiple components share copies of the same software and can be influenced by the same activation scenario.

BAHAMAS works by employing details of the SDLC as inputs to evaluate the probability of defects remaining within a software, then provides an indication of that software’s failure probability. When software is used redundantly, BAHAMAS is effectively evaluating the probability of defects that exist between redundant software components. For the purpose of supporting CCF assessments, BAHAMAS employs the common or shared attributes found between the software of a DI&C system. The shared SDLC details act as input for BAHAMAS to predict the probability of shared defects and ultimately the probability of shared failures or CCFs.

Appendix B
BAHAMAS Methodology and User Guide

Page intentionally left blank

APPENDIX B – BAHAMAS METHODOLOGY AND USER GUIDE

This section details the BAHAMAS methodology which comprises of five steps. Together these steps, and the details provided, serve as the user manual for BAHAMAS. Figure B-1 shows the steps of BAHAMAS. These steps incorporate the main concepts and theory described in sections A.1-A.6. Throughout the following section, details will be provided for implementing and using BAHAMAS code. In its current form, the BAHAMAS has been implemented with MATLAB (version 2019b) [65].

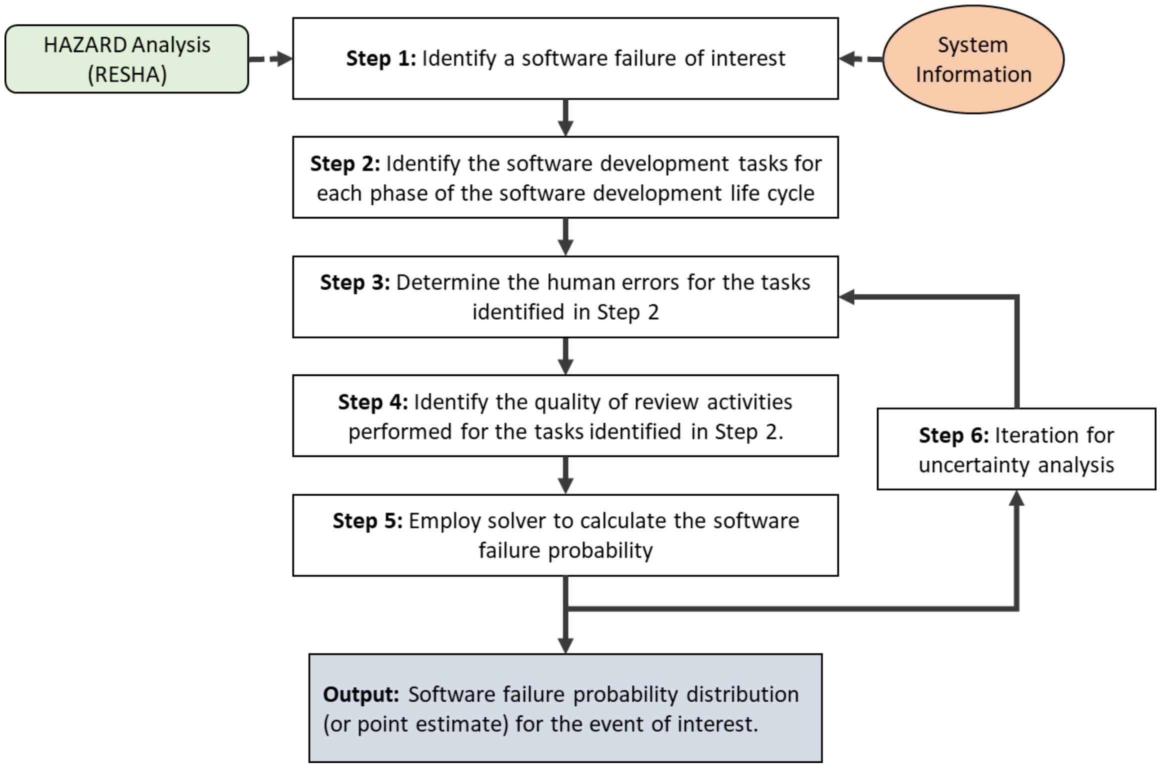


Figure B-1. Workflow of BAHAMAS.

B-1. Identify a Software Failure of Interest.

This step is coupled with hazard analysis activities. Hazard analysis will identify failures of interest. These failures belong to DI&C systems, structures, and components (SSCs). Within the LWRS-developed framework, RESHA postulates software failure events with different modes of failure (i.e., UCA/UIF-A, B, C & D). The details of the failure of interest should be used to establish the overall scope of the quantitative assessment. From there, it is necessary to identify the software responsible for the SSC behavior. Once these are identified. Move to next step.

B-2. Identify the Software Development Tasks for Each Phase of the SDLC.

SDLC activities and their tasks are clarified by the standards, manuals, programs, plans, and policies followed by developers. The SDLC tasks should be identified and mapped to the to the stage they most directly correspond within the BAHAMAS network. The stages selected for BAHAMAS are concept, design, implementation, testing, and installation & maintenance stages. Even though standards and guidelines may show consistency for when a task is performed, the specifics can vary. Identified tasks shall be recorded in the input file for BAHAMAS (see Table B-1 for the example input file).

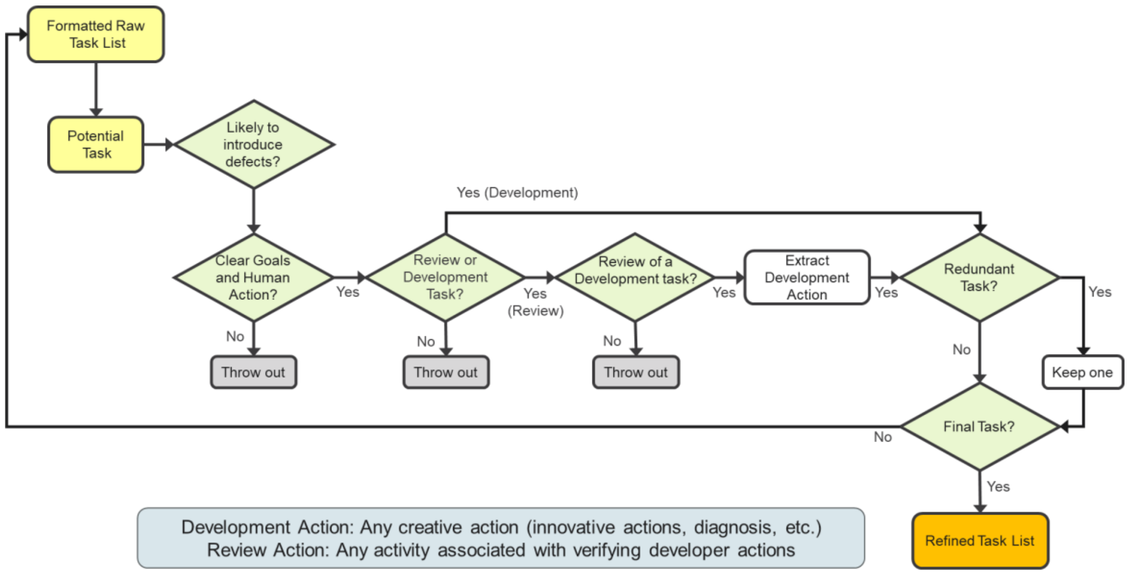


Figure B-2. Flowchart for assisting SDLC task identification.

B-3. Determine the HEP for the Tasks Identified in Step 2.

The SDLC tasks identified in the previous step are to be evaluated for human errors. Figure B-3 provides a general guidance for assigning a mode to each SDLC task. The flow chart will provide for a given task some combination of D1, D2, C, O, and OC. The general distributions for each mode combination are found in A.1.1. The union of task HEPs provides an indication of the probability of defects for each stage per Equation (A.1). Of course, BAHAMAS tool will evaluate Equation (A.1) as part of its automatic calculation. All that is needed for this step is to populate the Task List.xlsx input file.

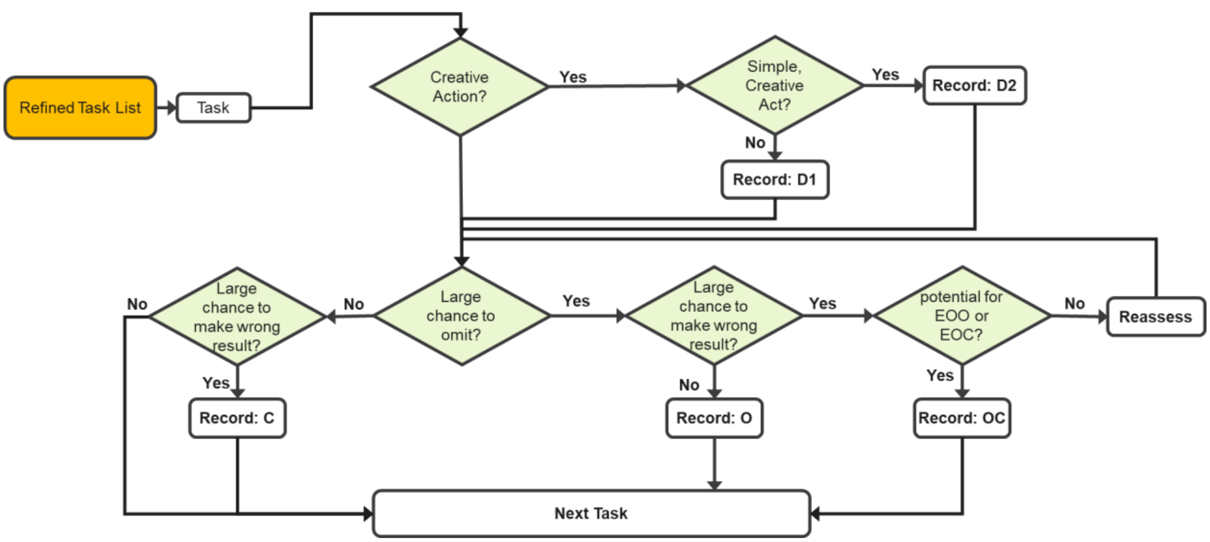


Figure B-3. General process for identifying SDLC tasks to quantify.

Note: as an alternative to scoring the tasks individually. A screening approach can be applied by expert elicitation where the stage HEP distributions can be approximated.

B-4. Identify the Quality of Review Activities Performed for the Tasks Identified in Step 2.

The quality of review efforts influences whether defects will remain in the software. For this work, quality of review is determined by the number of reviews and specific review activities (i.e., trigger coverage) that are employed for the SDLC. Future work may incorporate other concepts. There are two approaches for addressing the quality of review activities. The primary approach is given below. However, as an alternative, a distribution can be specified for trigger coverage and review number. Expert elicited distributions capture the uncertainty of review quality when the task-level analysis proves infeasible.

Trigger Coverage

The specific review activities performed (i.e., trigger coverage) for each task shall be assessed. There are 21 total triggers listed in [16]. Table A-5 shows these triggers and their associated activities adapted from [16]. The stage-level trigger coverage (TC) is determined from the average of each task-level trigger coverage (tc) which is the percent of relevant triggers that have been covered for a task of a particular SDLC stage (see equation (A.2)). The task level trigger coverage (tc) is entered for each task as part of the input file (see Table B-1).

Review Number

The number of reviews employed for each task shall be assessed and recorded in the input file. The average number of times each of the relevant triggers were reviewed is an indication of task trigger coverage. BAHAMAS then calculates the average number of reviews for a particular stage as defined by Equation (A.3), where r is the number of reviews performed for each task and T is the total number of tasks for a given stage.

B-5. Employ BAHAMAS to Calculate Software Failure Probabilities

The BAHAMAS software tool is used to calculate software failure probability. The calculation of software failure probability follows the discussion from A.5. BAHAMAS requires the manual activities of gathering data and providing inputs from which it then can perform an automatic calculation. Output from BAHAMAS is software failure probability, Figure B-4 provides an overview of BAHAMAS tool and its use.

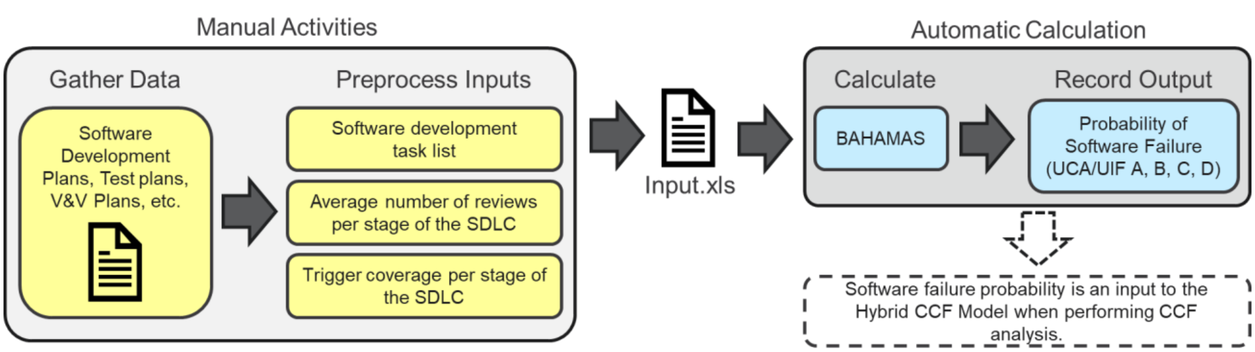


Figure B-4. BAMAHAS user overview.

Requirements:

- MATLAB Version 2019b or later.
- Start MATLAB
- Access to the following files within a known location for example: (…\BAHAMAS-Folder)

- BAHAMAS.m (Primary code)
- HEPcalculation.m (Supporting code)
- HEPData.m (Supporting code)

Input file

- Input file name: Task List
- Input file format: Excel (.xlsx)
- Input file location: Within same file location as BAHAMAS.m for example (…\BAHAMAS Folder\Task List.xlsx)

There are six sheets for the Excel workbook input file, each corresponding to a stage of the SDLC modeled by BAHAMAS. These sheets are named as follows:

- Sheet 1 name: Concept
- Sheet 2 name: Requirement
- Sheet 3 name: Design
- Sheet 4 name: Implementation
- Sheet 5 name: Testing
- Sheet 6 name: Install and Maintenance

The format for each sheet of the Task List.xlsx is shown in Table B-1. Columns 1-5 are mandatory. Any additional column can be used to save useful information such as reference files or notes for each task.

Table B-1. Input file format for BAHAMAS.

Task num	HumanErrorMode	Task Description	Review num	Trigger Coverage
1	D1OC	Define system architecture	2	1
2	D1OC	Assess each candidate architecture against constraints and requirements	2.1	0.975
3	D1OC	team defines interface architecture	2.2	0.999
…	…	…	…	…
end	…	…	…	…

Evaluation Mode 0

BAHAMAS has two evaluation modes that are useful depending on the goals of the assessment. The purpose of this mode is to expedite the identification of the mean software failure probability when compared to mode 1. This approach is an application of the central limit theorem and effectively evaluates the mean value many times.

Settings within BAHAMAS.m

- ModelEvalMode = 0
- numSamples specifies the number of samples used to generate Stage level average HEPs. This is in reference to equation() from section A.1. The numSamples default is 2000.

- K_index specifies the number of samples of the mean software failure probability that will be evaluated by BAHAMAS. Note that this will be an approximate. Default is 20000.
- set_N specifies the size of each of the K_index sample sets. Default is 60.
- TC_Rn_UQmode specifies whether the average number of reviews per stage and the average trigger coverage per stage of will be calculated directly from the input file (i.e., TC_Rn_UQmode = 1) or that they will be accounted for by expert elicitation (i.e., TC_Rn_UQmode = 0).
- For case when TC_Rn_UQmode = 0
  - RevNum is function for specifying the distribution that accounts for the average number of reviews for a given stage. This is a beta distribution that will be shifted to a desired range.
    - RevNum requires four user inputs: alpha, beta, Rn_min, Rn_max.
    - alpha and beta are shape parameters for the beta distribution.
    - Rn_min is the minimum number of reviews that the expert elicitation deems applicable for the given stage.
    - Rn_mas is the maximum number of reviews that the expert elicitation deems applicable for the given stage
  - TriCov is the function used to specify the distribution for the average trigger coverage for a given stage. This is a beta distribution.
    - TriCov has two user inputs: alpha, mean.
    - mean is the expected value for trigger coverage.
    - alpha controls the shape of the distribution to reflect the uncertainty of the elicitation.

Evaluation Mode 1

This evaluation mode provides a view of the impact of changing sampling size. In other words, this provides a view of convergence. This mode outputs the distribution for software failure probability for each of the UCA/UIF modes given the final number of samples employed. Specifically, the software failure probability evaluated from 1 set of samples. This contrasts with mode 1 which provides the results for a mean software failure probability based on n sets (K_index) of size N (set_N).

Settings within BAHAMAS.m

- ModelEvalMode = 1
- Numbersamples is a user supplied value. Default is 40K. It is used by mode 1 to generate set_N, which is the size of each set that will be used to calculate mean results from BAHAMAS. Initially, the final set size is equal to Numbersamples. However, BAHAMAS truncates the sets, capping out at approximately 70% of the initial value for Numbersamples (e.g., for 40K input, the set size for the final term of set_N will be 28K). Ultimately, should the user desire more samples, they should just increase the Numbersamples starting input value.
- K_index specifies the number of samples of the mean software failure probability that will be evaluated by BAHAMAS. Note that this will be an approximate. In mode 1, K_index is not a user supplied value
- set_N specifies the size of each of the K_index sample sets. In mode 1, set_N is not a user supplied value
- numSamples specifies the number of samples used to generate Stage level average HEPs. This is in reference to equation() from section A.1. The numSamples default is 2000.

- TC_Rn_UQmode specifies whether the average number of reviews per stage and the average trigger coverage per stage of will be calculated directly from the input file (i.e., TC_Rn_UQmode = 1) or that they will be accounted for by expert elicitation (i.e., TC_Rn_UQmode = 0).
- For case when TC_Rn_UQmode = 0
  - RevNum is function for specifying the distribution that accounts for the average number of reviews for a given stage. This is a beta distribution that will be shifted to a desired range.
    - RevNum requires four user inputs: alpha, beta, Rn_min, Rn_max.
    - alpha and beta are shape parameters for the beta distribution.
    - Rn_min is the minimum number of reviews that the expert elicitation deems applicable for the given stage.
    - Rn_mas is the maximum number of reviews that the expert elicitation deems applicable for the given stage
      - TriCov is the function used to specify the distribution for the average trigger coverage for a given stage. This is a beta distribution.
        - TriCov has two user inputs: alpha, mean.
        - mean is the expected value for trigger coverage.
        - alpha controls the shape of the distribution to reflect the uncertainty of the elicitation.

Run Script

To run BAHAMAS and calculate the software failure probability for the failure of interest, the users do the following:

1. Verify Task List.xlxs is complete.
2. Verify Task List.xlxs, BAHAMAS.m, HEPcalculation.m and HEPData.m are in same folder.
3. Open BAHAMAS.m with MATLAB.
4. Select the evaluation mode.
5. Input settings corresponding to the evaluation mode chosen.
6. Select the run icon in the MATLAB editor environment.
7. Collect results.

Appendix C
Reliability Analysis of ML-integrated Control Systems

Page intentionally left blank

APPENDIX C – RELIABILITY ANALYSIS OF ML-INTEGRATED CONTROL SYSTEMS

C-1. Overview

Significant research has been conducted on the integration of machine learning (ML) methods into various information and control systems. ML has been applied to enhance plant diagnostics [66, 67, 68], automate the scheduling of maintenance tasks [69, 70, 71], enable autonomous control [72], develop digital twins (DTs) [73, 74], etc. In such data-driven models, a training dataset typically defines a model's function by learning the latent correlation between the input and target values. The function realized through training is governed by the multiplication of nondescript weights and biases that is generally difficult to interpret. The model is also intended to generalize over a range of cases extending beyond the discrete points in the training data. Thus, ML model validation and verification are typically conducted to assess generalizability using holdout testing sets, which are samples not previously seen by model but have similar attributes to the training data (e.g., distribution, features, skewness, range). K-fold cross-validation [75] is an example of holdout data training and validation. From K-fold, the predictive accuracy on the holdout set is the assumed accuracy in post-training operation. This is based on a closed-world assumption [76] in which new input data are assumed to be drawn independent and identically distributed (i.i.d) relative to the training data.

Although ML model methods for training, validation, and verification have advanced significantly, data-driven models can experience major performance reductions when applied to real-world operational environments [77]. In addition to reduced predictive accuracy, poor-performing ML models can also present safety risks. For example, misclassifications in self-driving vehicle algorithms [78] have led to public losses. The root cause of ML model failures may originate from regressional inconsistencies [79], inherent distributional rigidness [80], metric optimization failures [81], and unintended adversarial examples [82]. Thus, if these models are to be adopted in real world systems, it is paramount that their reliability and trustworthiness be guaranteed.

Reports by the National Institute for Standards and Technology [83] and the U.S. Nuclear Regulatory Commission [84] have also indicated that trustworthiness in ML presents a critical barrier to its adoption, and will play a vital role in the safe, accountable, and secure operation of data-driven ML systems.

One possible way to develop trust in ML predictions is to assess how well-matched operational data is to training data. When operational and training data are identical (i.e., prefect recall), the closed-world assumption holds, and we can inductively reason that there is a basis for trusting the model's predictions via training performance. In such cases, samples in the training database may serve as specific points of evidence to support the generalized conclusions made by the ML model. However, realistically speaking, operational data will rarely match the corresponding training data. For instance, many published research projects on data-driven ML-based condition monitoring are based on datasets that artificially manufacture defects and degradations [85]. Artificially generated datasets are useful in the preliminary development of ML models as they focus on the problem assessed. However, these datasets do not accurately reflect realistic scenarios which may vary in measurement sensitivity, operating condition, anomalies, and component types, all contributing to uncertainties in the model prediction [85]. Furthermore, training data is cleaned by removing noise, statistical anomalies, etc., so as to focus the learned function on a specific latent correlation. Thus, the closed-world assumption on realistic data does not always hold true. The degree to which the assumption holds depends on how "far away" new samples in the operational environment are to the training data. However, determining when (i.e., at what distance) a new sample is to be considered in-distribution of the training data (i.e., an interpolation task) is challenging, thus creating the basis for out-of-distribution (OOD) [76] detection research.

Fundamentally, it is well documented that ML models excel at interpolation (or near-interpolation) tasks, but falter at extrapolation. Identification and separation of interpolation and extrapolation tasks are the fundamental problems in OOD detection. In general, OOD detection is intended to identify any test sample that is (a) a statistical anomaly, (b) beyond the range of the training data, (c) a class novelty, or (d) a statistical outlier [76]—any of which may lead to prediction failure.

With that in mind, the present work aims to determine "how far" a new sample must be in order to be considered OOD, and how this value can be used to instill trust in ML-integrated control systems. We establish a method of evaluating the reliability of model predictions by comparing them against the training data and utilizing the closed-world assumption. The fundamental assumption of this work is that training data can be used as inductive evidence to justify model predictions. An OOD detection method referred to as the Data Auditing for Reliability Evaluation (DARE) is developed and can be used to assess the reliability of predictions made by any data-driven, time-invariant (or memoryless) model. We demonstrate the DARE methodology on a DT developed for simulated loss-of-flow transients in the Experimental Breeder Reactor II (EBR-II) [86]. We also provide a use case covering how DARE can be integrated into a real-time ML-integrated control system to filter out incorrect predictions, based solely on the training data.

In this work, we propose DARE, a black-box real-time OOD method, to determine the reliability of individual ML predictions. Reliability determination is based on whether test samples can be represented by a limited set of local training points. The local training points act as inductive evidence to support (or refute) ML model predictions. The intent is to develop a highly interpretable methodology for assessing prediction reliability. We demonstrate how this methodology can be used to construct a temperature map of the local training data upon which the prediction is based. In addition, we show how DARE can be used to decrease the false positive rate when models are (a) poorly trained relative to the operational environment, (b) trained on sparsely populated data, or (c) experience a distributional shift in the input space. Overall, our framework can be used to assess the reliability of ML predictions for the detection of various OOD problems enabling real-time operational awareness of potential failures in ML model predictions.

C-2. Methodology

In this section, we build the mathematical formulation behind our construction of the reliability function for data-driven ML models. To begin, for an ML model, their required functionality is the classification or regression task assigned for an application. For instance, the function of an image classifier is to correctly label images; the function of a regressor is to predict a specific value within an acceptable criterion (e.g., temperature). In this case, the functionality of a ML model does not specify how it completes a task, but whether or not the model output aligns with the developer's goals for creating the model.

Next, recognize that the functionalities of data-driven neural-network-based models are assigned through training data, not through the explicit architecture of the model (e.g., nodes, activation function, and layers). The same neural network configuration can be retrained on any equivalent task (i.e., classification), indicating that architecture plays a minor role in the intended functionality. Thus, developing a representative training dataset to the target operational environment is critical for the intended functionality of an ML model. In this respect, we assume that the functionality of a data-driven model is primarily based on the training data, and that model architecture plays an insignificant role in the intended functionality of the model.

When constructing the representative training dataset (whether through synthetic or simulation means), the developer confirms that the target of the training set is indeed the intended function of the model. For instance, in a temperature regressor, the training dataset contains inputs that correlate with a specific output temperature. The model performance (e.g., accuracy) does not necessarily need to be perfect, so long as the prediction made by the model is what was intended by the developer. This second assumption implies that the model can achieve (either partially or wholly) its intended functionality when it predicts on the training set. The second assumption can be formulated as follows. Let $\vec{x}$ denote a single training sample within the training set X and corresponding prediction y. Let $\vec{x}'$ denote a random input test sample and the corresponding model prediction, y'. The model training performance on X is denoted by Q_X and may be represented by the F1 score or other metrics that describe the predictive correctness of a model on a given task. Given $\vec{x}' \in X$, the likelihood that a model will succeed (and complete its intended function) for a single sample is approximated as $r = Q_X$, assuming that the distribution of $\vec{x}'$ is i.i.d in X. Here, r is the probability of a model achieving its intended function for a single instance—in other words, instantaneous success rate. The instantaneous success rate is equal to one minus the instantaneous hazard rate (i.e., $\lambda = 1 - r$) as $1 - Q_X$ describes the probability of model failure on a single input instance.

While useful, this approximation overlooks two critical aspects. The first aspect is where $\vec{x}' \notin X$, and all intermediate cases where $\vec{x}' \approx \vec{x}$. In most applications, $\vec{x}' \in X$ is an unrealistic assumption, as it undermines the intended generalizability of the model. Therefore, we assume that a model can achieve its functionality on test input samples that are nearly identical to the training set. This space may be described as $X \pm \epsilon$, the interpolable range where a model is postulated to be able to achieve its functionality. ϵ signifies the extension of the training dataset per dimension and varies from feature to feature. The exact degree of capability on the interpolable space is determined by the proximity and density of training data, as well as the magnitude of ϵ. Typically, ϵ is difficult to determine and must be approximated, especially for higher dimensions.

To address this issue, a secondary term to instantaneous success rate was developed to represent proximity: the condition $\vec{x}' \approx \vec{x}, \vec{x}' \in X \pm \epsilon$ and denoted by $K(\cdot)$, degree of congruency (DoC), with a range between zero and one. When $K(\cdot) = 0$, the test sample is OOD, meaning that no training samples exist or are nearby that can support a model's prediction. When $K(\cdot) = 1$, the model is predicting on a training sample (i.e., perfect recall) and has a Q_X probability of achieving its intended function. $K(\cdot)$ signifies the degree that the closed-world assumption is true for a test input relative to the training set and is described by a kernel function used to determine the overlap between the test and existing training samples via point-wise comparison. The default kernel utilized by DARE was adapted from the normalized 1D Laplacian distribution from [87], given in Equation (C.1):

$$K(\vec{x}, \vec{x}') = \exp(-2|\vec{x}' - \vec{x}|) \tag{C.1}$$

In this formulation, the question of whether a test sample is in-distribution is determined based on the absolute distance between the two points. For a distance of zero (e.g., $\vec{x}' = \vec{x} \in X$), the sample is definitively in-distribution. This does not imply the model prediction is correct (as illustrated by Q_X). The distance function may be modified to incorporate additional expert knowledge on the training dataset as per Equations (C.3) and (C.4), which dictate the decay rate and bandwidth of $K(\cdot)$. Note that the data do not need to be normalized to calculate $K\left(\vec{x}, \vec{x}^{'}\right)$.

$$D(\vec{x}', \vec{x}) = \sqrt{(\vec{x}' - \vec{x})^{\mathrm{T}} V^{-1} (\vec{x}' - \vec{x})} \tag{C.2}$$

$$V = \begin{pmatrix} \beta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \beta_n \end{pmatrix} \tag{C.3}$$

$$K(\vec{x}', \vec{x}, V) = \exp(-2 \cdot D(\vec{x}', \vec{x})) \tag{C.4}$$

where V is a diagonal matrix, referred as the covariance structure matrix, specifying the decay rate per feature axis. It is not calculated from feature variance. As the inverse of a diagonal matrix of any size is simply the reciprocal of the diagonal, it does not suffer from the computational delays associated with large inverse matrices. Each feature has a unique decay rate, β_i, that dictates the interpolable range (i.e.,

$\pm\epsilon$) of that feature in that axis. β_i does not signify feature importance. The β_i are calculated based on the desired range of interpolation, rather than through optimization as seen in Equation (C.5). This algorithmic choice makes the method as interpretable as possible to developers.

$$\beta_i = \frac{{x_{d,i}}^2}{(\ln L)^2} \tag{C.5}$$

Two hyperparameters serve to determine the decay rate, but only one, $x_{d,i}$, is actively modified by the user. It designates the bandwidth of the kernel in the ith feature axis. This parameter is referred to as the interpolation length, as it allows users to manually specify how far away from a training point credible interpolations can be made (i.e., $x_{d,i} = \epsilon$). It also directly shapes the decay curvature of the kernel. The parameter L specifies the level at which the interpolation length applies. By default, the level is set to 20% (or 0.2) DoC. Though other levels can be used (e.g., 10%), a level of 0% is not recommended, as it is an asymptote for many kernels. In Figure C-1, a 1D exponential kernel is shown to demonstrate how the hyperparameters influence the decay rate and bandwidth. The kernel is arbitrarily centered at $x = 0.5$, under the assumption that a training point exists there. In the example, an interpolation length of $x_d = 0.4$ is specified, implying $x = 0.5 \pm 0.2$. The manual assignment of x_d provides a more intuitive assignment of feature relevance and may be used to identify data coverage inadequacies in the training data.

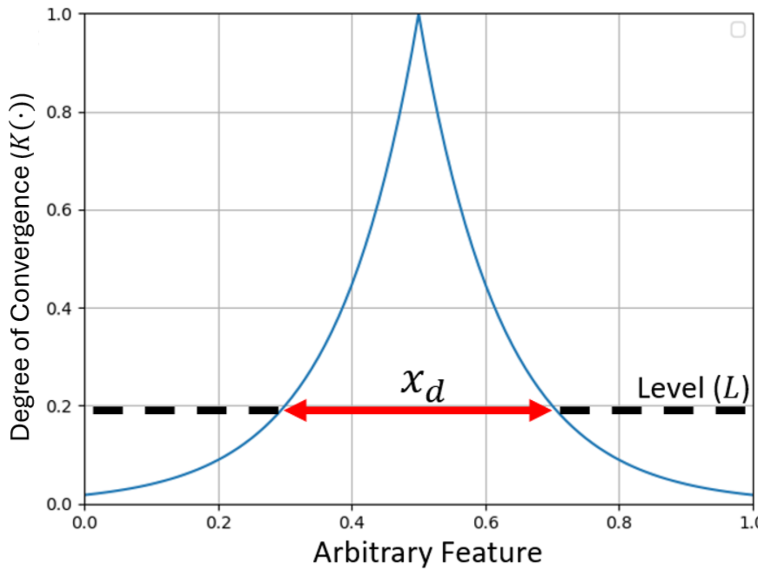


Figure C-1. 1D exponential kernel with $x_d = 0.4$ @ L= 0.2, arbitrarily centered at $x = 0.5$.

The second aspect relevant to the prediction is the local density of data points near and around the test sample. The existence of a single training point near a test sample may be insufficient evidence to justify prediction reliability. Dataset outliers, for instance, are anomalies that render the model incapable of achieving its intended function. Multiple training points near a test sample indicate greater evidence to support a model prediction. To incorporate the presence of multiple training points, the DoC of nearby points is averaged over N training samples. The full equation for the instantaneous success rate is given in Equation (C.6):

$$r(\vec{x}') = Q_X \cdot \frac{1}{N}\sum_{i=0}^{N} K((\vec{x}', y'), (\vec{x}_i, y_i)) = Q_X \cdot \mu \tag{C.6}$$

where N, is user-specified and dictates the number of nearby points required for a prediction to be deemed reliable. The larger the N, the more training data is required to support a prediction's reliability in achieving the intended functionality. The term μ is the proximity averaged degree of congruence (PADoC) of a prediction over N training samples.

Kernel Selection

Other popular kernels are implemented in the DARE methodology—namely, cosine, tophat, linear, and Gaussian. The kernel equations and feature decay calculations are given in Table C-1. Note that decay calculations are automatically determined by DARE; only $x_{d,i}$ is specified by the user. Figure C-2 shows all available kernels with convergent interpolation lengths mapped on a training point at $x = 0.5$. The kernels differ only in terms of spread and decay rate per feature axis. In preliminary tests, no one kernel was determined to be superior to another and it is anticipated that different kernels will prove optimal for different applications.

Table C-1. Kernel functions for degree of congruency.

Name	Kernel Function ($\boldsymbol{K(x)}$)	Decay ($\boldsymbol{\beta_i}$)
Exp. (default)	$\exp(-2D(x))$	$\frac{x_{d,i}^2}{(\ln L)^2}$
Cosine	$\begin{cases} \cos\left(\frac{\pi}{2}D(x)\right), D(x) < 1 \\ 0, D(x) \geq 1 \end{cases}$	$\frac{\pi^2 x_{d,i}^2}{(4\cos^{-1}(L))^2}$
Tophat	$\begin{cases} 1, D(x) < 1 \\ 0, D(x) \geq 1 \end{cases}$	$\frac{x_{d,i}^2}{4}$
Linear	$\begin{cases} 1 - D(x), D(x) < 1 \\ 0, D(x) \geq 1 \end{cases}$	$\frac{x_{d,i}^2}{4(1-L)^2}$
Gaussian	$\exp(-D(x)^2)$	$\frac{x_{d,i}^2}{-4\ln L}$

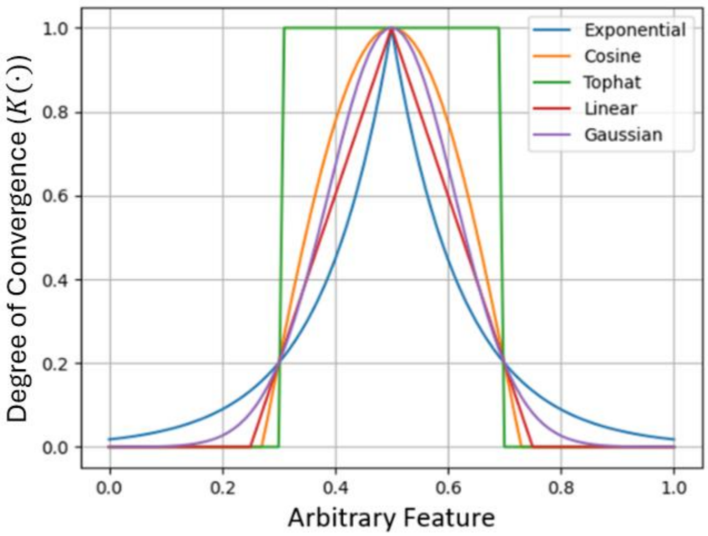


Figure C-2. Available kernels with $x_d = 0.4$ @ L=0.2, arbitrarily centered at $x = 0.5$.

Integration with ML-Integrated Control System

Figure C-3 presents an example use case in which the DARE methodology is integrated into an ML-supported control system. DARE is intended to operate in parallel with an ML model and provides real-time prediction reliability assessments per test sample. Block 1 represents the training data used to develop the ML model and serves as the knowledge base for the DARE calculations. This knowledge base consists of both the training input and training target. In block 2, a data-driven ML model receives inputs from the operational environment and makes a prediction (y'). In block 3, the prediction from the ML model is combined with the input test from the operational environment and compared against the knowledge base. The output of block 3 is the instantaneous success rate of the model for the current input, $\vec{x}'$, and prediction, y', given the training set, X, and the model training performance, Q_X. Block 4 is a decision block for determining whether to keep the current prediction and proceed with the normal anticipated function or to engage auxiliary functions. A prediction is accepted if it attains a certain level of instantaneous success rate denoted by the threshold, ξ. An acceptance boundary (ξ) of 0.5 implies that the current sample and prediction ($\vec{x}', y'$) is on average, across N training points, within 50% of the interpolation length per each feature. Choosing a relevant ξ for an engineering application will depend on the application but is not investigated in this work. In the event a prediction cannot be deemed reliable (even if correct), non-data-driven or auxiliary functions may be engaged as alternate pathways.

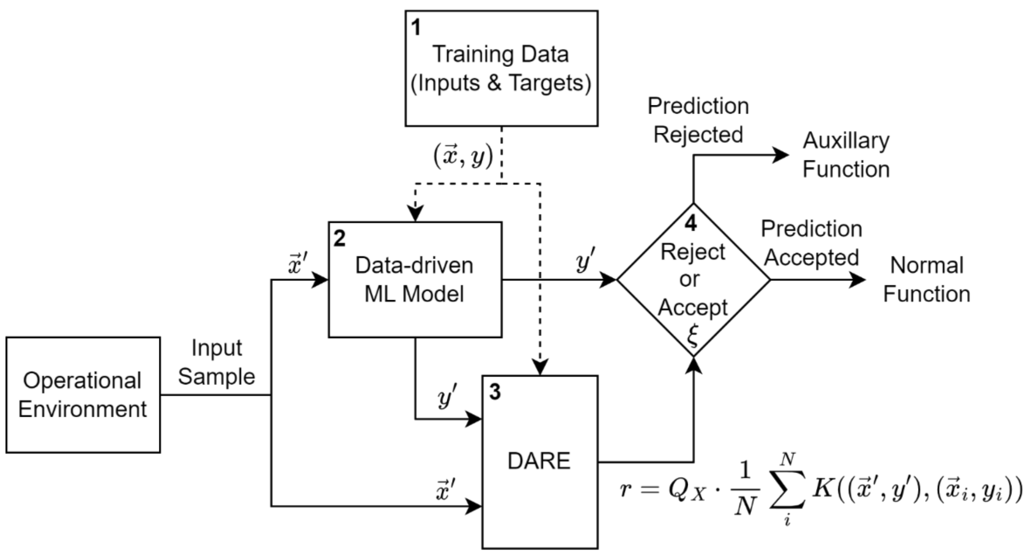


Figure C-3. Basic framework of DARE coupled with an ML-integrated control system.

Development Performance Metrics

This section provides guidance and performance metrics for assessing the efficacy of DARE. The presented metrics are for training and validation purposes only, and cannot be used in deployment, as true prediction targets are unknown. In addition, the metrics are framed from the perspective of the use case presented in Figure C-3 and potential stakeholder concerns.

First, note that in the use case, DARE serves as a data supervisor, rejecting and accepting predictions based on their instantaneous success rate. As the acceptance or rejection of predictions is a binary choice, exactly four possible outcomes stem from each model prediction: (1) the prediction is correct, and DARE accepts the result (true positive); (2) the prediction is incorrect, and DARE rejects the result (true negative); (3) the prediction is incorrect, and DARE rejects the result (false positive); or (4) the prediction is correct, and DARE rejects the result (false negative). With this in mind, we may use the false negative rate (FNR), seen in Equation (C.7), and false discovery rate (FDR), seen in Equation (C.8), to determine performance. FNR measures, from the perspective of the use case, the ratio of correct model predictions rejected by DARE over the total number of correct predictions. It is analogous to a model's degree of performance degradation and is denoted as $f_{degrade}$. FDR, on the other hand, measures the ratio of accepted incorrect predictions over the total number of predictions accepted. It is analogous to the degree of peril as a result of incorrect predictions and is denoted as f_{peril}. Note that FNR and FDR can be used to derive recall and precision scores to calculate an F1 score, as per Equation(C.9).

$$f_{degrade} = \mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{C.7}$$

$$f_{peril} = \mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TP}} \tag{C.8}$$

$$F_1 = \frac{2 \cdot f_{precision} f_{recall}}{f_{precision} + f_{recall}} = \frac{2(1 - \mathrm{FDR})(1 - \mathrm{FNR})}{2 - \mathrm{FDR} - \mathrm{FNR}} \tag{C.9}$$

From a stakeholder perspective, there may be two potential objectives when choosing performance metrics. If the stakeholders are concerned about maximizing safety by using only supported predictions, minimizing f_{peril} may be of higher priority than minimizing $f_{degrade}$ as this minimizes the number of false positive predictions. Conversely, if the stakeholders are concerned about performance without sacrificing safety, maximizing the F_1 score may be top priority as it balances both metrics.

The performance metric of concern will entail different optimization schemes. In DARE, the hyperparameter to optimize is the interpolation length ($x_{d,i}$) per feature axis. Once this is optimized, the use case can be deployed for operational test samples. Note that optimization cannot be conducted on test samples. In the present work, a grid search method was utilized to optimize these parameters, as it is the most straightforward method. However, other optimization methods such as the Sheather and Jones [88] and the Park and Marron [89] methods also exist. Cross-validation methods (e.g., maximum likelihood cross-validation [90]) may also be employed. The variance per feature axis may be used as a preliminary estimate of the interpolation lengths if knowledge of the operational conditions is unknown.

However, recognize that interpolation length optimization also comes with trade-offs—namely, the stability of the f_{peril} and $f_{degrade}$ metrics, depending on which one was minimized. For instance, assume that optimal interpolation lengths were found that first minimize f_{peril}, followed by $f_{degrade}$. Due to this minimization, no margin of error is provided for the reliability determination, potentially resulting in a non-zero f_{peril} when the use case is deployed. Thus, from a developer perspective, it may be preferrable to establish slightly narrower interpolation lengths than the optimized ones so as to allow for a margin of predictive error. This optimization challenge is discussed in greater detail in further sections.

C-3. Case Study

The DARE framework (see Figure C-3) was applied to a diagnostic DT model [66] developed for the Near Autonomous Management and Control System (NAMAC) [72] I&C system. The implemented NAMAC workflow can be seen in Figure C-4. Additional details regarding the workflow can be found in [72, 91]. The DT is a deep feedforward neural network developed for partial loss-of-flow accident (LOFA) scenarios to predict peak fuel centerline temperatures (T_{FCL}) and is indicated by the red box in Figure C-4. The DT predicts unobservable safety significant factors that are used to guide operator control actions in transient scenarios based on real-time simulated sensory data.

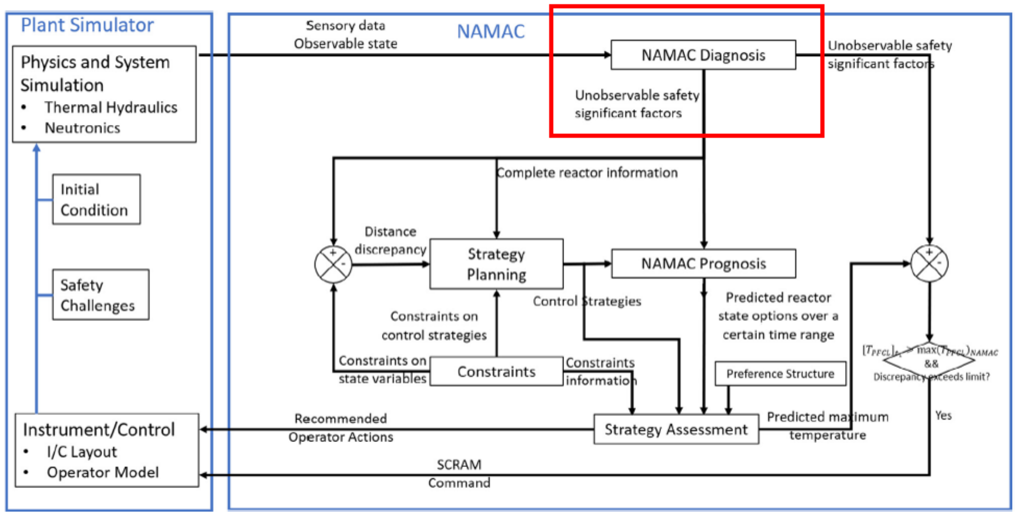


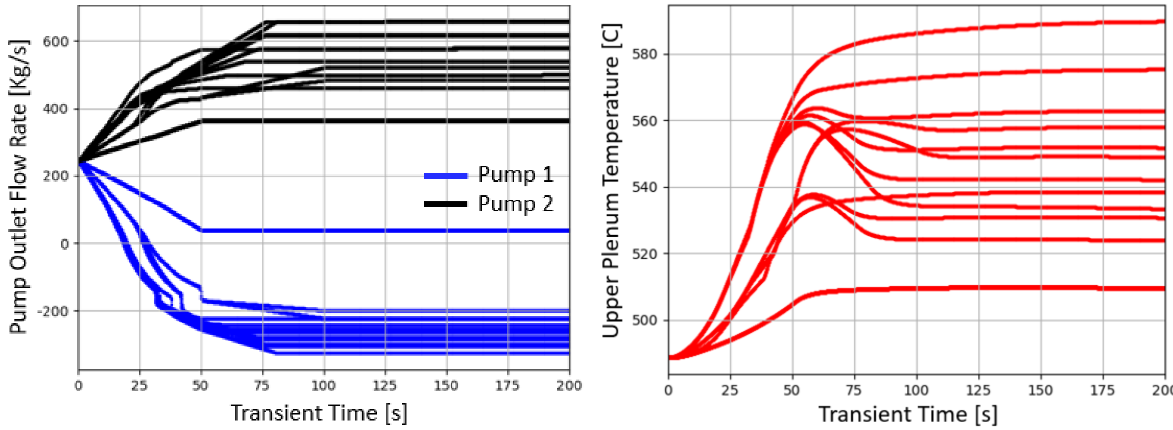
Figure C-4. Operational workflow of the NAMAC system for controlling EBR-II [72].

The datasets were gathered from a plant simulation of EBR-II using a GOTHIC™ model and benchmarked against experimental data [86]. In this simplified EBR-II model, two separate primary centrifugal sodium pumps (designated P1 and P2) provide coolant flow through a low-pressure lower plenum, high-pressure lower plenum, the active core, an upper plenum, and an intermediate heat exchanger. The simulation models coolant drawn from the sodium tank and injected into the lower plenums. From there, coolant within the high-pressure plenum flows through the active core to cool the subassemblies, control rods, and inner reflector and exits in the upper plenum. The hot coolant joins the "Z-pipe" from the upper plenum to the intermediate heat exchanger before re-entering the sodium tank. Boundary conditions impose temperature of the sodium on the inlet side of the sodium tank [72]. Virtual probes are specified in the simulator that collect data on various parameters at different locations. In total, fourteen different measurements are collected throughout the progression of each transient, namely: flow rate (3), temperature (5), pump rotational speed (2), pump head (2), and reactivity (2). Before developing the diagnostic DTs, feature selection was conducted to determine the most relevant measurements to T_{FCL} via the Pearson correlation. The upper plenum temperature (T_{UP}) and total core flow rate (v_T), or the sum of P1 and P2 flow rate, were selected due to their strong correlation to T_{FCL}. No other pre-processing steps were conducted during the development of the DTs.

To generate data, the EBR-II model is first modelled in steady state to establish the normal operational regime of the plant. From there, a LOFA scenario is simulated by manually ramping down one of the two pumps' rotational speed, decreasing the overall coolant flow through the core block. The objective of the NAMAC system is to re-establish normal flow conditions by ramping up the rotational speed of P2 to compensate. For this dataset, it is assumed that the ramp up time of P2 occurs at the same instance as when P1 begins to ramp down (i.e., transient start time). Throughout the scenario, a diagnostic DT monitors plenum temperatures and flow rate and predicts the fuel centerline temperature of the subassemblies. The scope of the numerical demonstration can be represented by the time-dependent curve of P1's rotational speed, per Equation (C.10):

$$\omega_1(t) = \omega_0 \left(1 - \frac{1 - (\omega_1)_{end}}{T_1} t_0\right) \qquad \text{(C.10)}$$

where ω_0 is the nominal pump speed, T_1 is the ramp-down duration, $(\omega_1)_{end}$ is the normalized P1 end speed, and t_0 is the transient start time. By varying the pump end speed, from -20% to -80%, and ramp-down duration, from 20 to 80 seconds, different accident profiles can be simulated. The ramp-up rate of P2 is also varied from 100% to 150% as the compensatory action when P1 ramps-down. Note that pump rotational speed and flow rate are not linearly proportional. The parameters v_T, T_{UP}, and T_{FCL} are uniformly sampled for 2000-time steps (or 200 seconds) per transient. In Figure C-5, ten different LOFA transient profiles are shown as an example of the data generated with corresponding P1 and P2 flow rate and T_{UP}. The diagnostic DT uses two input features—v_T and T_{UP}—to predict the T_{FCL} for the current time step.



(a) (b)

Figure C-5. (a) Ramp-down and -up flow rate of P1 and P2 during a LOFA, (b) corresponding upper plenum temperature derived from EBR-II.

The full dataset details are provided in Table C-2. Column 1 of the table provides the dataset reference label, column 2 shows the number of simulated transients within each dataset, and columns 3–5 show the range of each collected input feature. Figure C-6 plots the target training and testing data together to show how they differ from each other. Important to note is that the test data contain transients in which the T_{FCL} is both hotter and colder than in the training data.

Table C-2. Size of training and testing sets used.

Label	**# of Simulated Transients**	**Range v_T**	**Range T_{UP}**	**Range T_{FCL}**
dTrain	125	263–488	488–569	606–698
dTest1	255	263–488	488–569	606–698
dTest2	255	230–488	488–589	606–728
dTest3	255	399–495	487–509	605–630

As only dTrain is used to train the DT, only dTrain transients are included in the DARE knowledge base. All three parameters from the training dataset are used to calculate the instantaneous success rate within DARE.

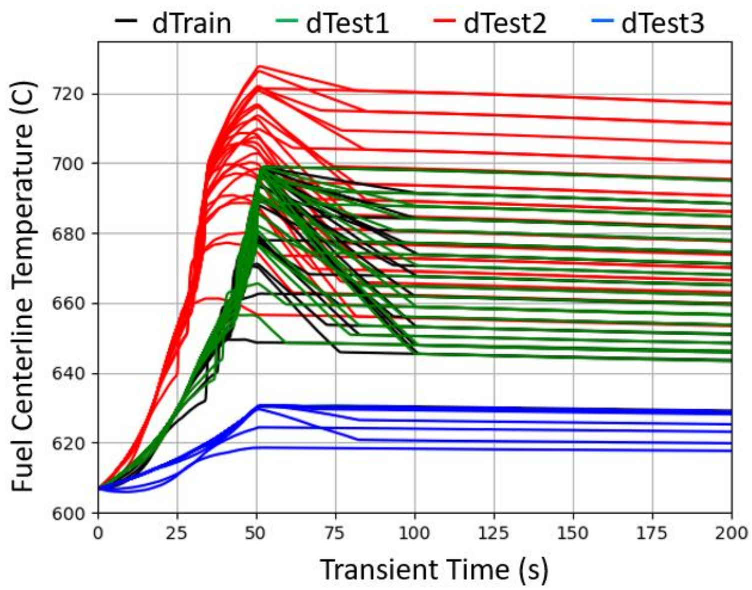


Figure C-6. DT training (black) and testing (colored) data.

The dTest sets are used to gauge DARE's performance under different scenarios. dTest1 consists of unseen transients that share similarities to dTrain. dTest2 contains transients with hotter T_{FCL} temperatures, and dTest3 contains transients with colder T_{FCL} temperatures. These transients were generated by increasing and decreasing the degree of P1 degradation respectively.

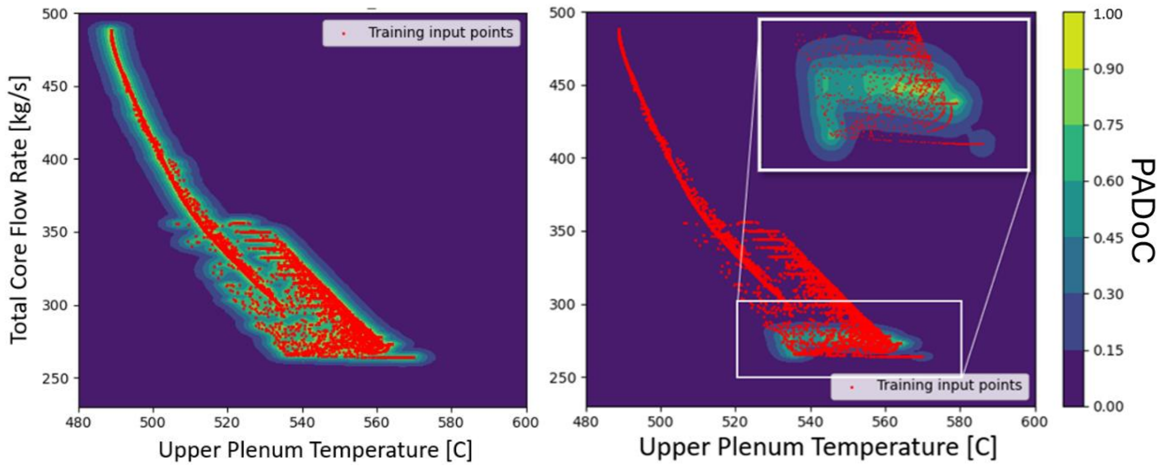
The objective of the diagnostic DT is to predict T_{FCL} within $\pm 10^{\text{o}}$C of the simulated temperature, a value that serves as the acceptance criterion for predictions. An arbitrary reject/accept boundary was set at $\xi = 0.5$ to examine the performance metrics developed for the use case in Figure C-3. T_{FCL} is generally unmeasurable, as obstructions in the fuel bundle limit thermocouple placement and may be used to calculate the core damage frequency. Anticipating the reliability of model predictions is thus critical to safety. In this case study, stakeholders may prefer to minimize the peril metric rather than the degradation metric. Also, note that the DT's post-training performance in this case study is irrelevant, as the intent is to assess the DARE methodology's capability on the instantaneous success rate. Models with poor predictive capabilities will have a high rejection rate, and vice versa. To investigate the instantaneous success rate of single predictions, Q_X is set to one such that $r = \mu$ (see Equation (C.6)]).

Table C-3 provides two arbitrary settings for DARE hyperparameters in order to illustrate how they impact performance metrics (i.e., $f_{degrade}$, f_{peril}, and F1-score). These two settings are intended to demonstrate how an increased interpolation length can lead to more generous calculations of the instantaneous success rate—and subsequently a decrease in model degradation ($f_{degrade}$)—but at the cost of an increase in false positive predictions (f_{peril}).

Table C-3. User-specified DARE parameters.

Parameter	Setting (a)	Setting (b)
x_{d,v_T}	$5\ kg/s$	$20\ kg/s$
$x_{d,T_{UP}}$	$5^o C$	$20^o C$
$x_{d,T_{FCL}}$	$5^o C$	$20^o C$
N	3	3

In Figure C-7a, a PADoC map is generated for input features only, to indicate areas in which a high degree of input training evidence exists to support model predictions. Realistically, DARE also requires the model prediction to make relevant assessments. In Figure C-7b, the training T_{FCL} is also considered, and a 2D slice of the 3D PADoC map for when $T_{FCL} = 690^{o}\mathrm{C}$ is shown. Figure C-7b illustrates the range of the inputs required to derive a non-zero PADoC for a specific prediction outcome. For instance, suppose a test input has an upper plenum temperature of $T_{UP} = 520^{o}\mathrm{C}$ and a total core flow rate of $v_T = 350\ kg/s$. First, input training data exist near that test input. Using only Figure C-7a, we may observe an PADoC of 0.75–1. However, if the model prediction is $T_{FCL} = 690^{o}\mathrm{C}$ given $(T_{UP}, v_T) = (520, 350)$, no nearby training points correlate to that temperature; thus, the PADoC is 0, as seen in Figure C-7b. Only inputs within the range of $T_{UP} \in (530, 570)$ and $v_T \in (260, 280)$ can possibly correlate to $T_{FCL} = 690^{o}\mathrm{C}$. The enhanced box area in Figure C-7b illustrates which input data points strongly correlate to $T_{FCL} = 690^{o}\mathrm{C}$.



(a) (b)

Figure C-7. (a) PADoC maps for training data inputs. (b) PADoC maps for training data at $T_{FCL} = 690^{o}\mathrm{C}$.

Figure C-8 shows the DT's T_{FCL} prediction for an unseen single loss-of-flow transient taken from dTest1. The T_{FCL} values from dTest1 are not part of the DARE knowledge base. The model prediction is segregated by prediction outcome: true positive (green line), true negative (blue line), false positive (red line), and false negative (purple line), based on the acceptance criterion of $\pm 10^{o}\mathrm{C}$. The ground truth is the simulated T_{FCL} (black line). In areas of poor predictions (i.e., 25–75 seconds), the instantaneous success rate (yellow line) plummets. Thus, in these areas, the model outputs should be (and are) rejected, as they diverge too significantly from the training data.

However, as a data supervisor, DARE is imperfect, so areas in which the model is capable (i.e., 23–25 seconds) can also be rejected. For the transient reflected in Figure C-8, 31.5% of correct predictions (i.e., ones within $\pm 10^{o}\mathrm{C}$ of the simulated T_{FCL}) were rejected, suggesting the model performance was degraded. The recall (i.e., $1 - f_{degrade}$) of the model was 78.9%, indicating that 78.9% of all predictions within the acceptance criterion had an instantaneous success rate over 50%.

Figure C-9a shows the DT predictions for a randomly selected transient from dTrain2. Recall that dTrain2 transients have a higher T_{FCL} than dTrain1 transients, and that the two datasets do not significantly overlap. In this example, 92.3% of all correct predictions were rejected, as their instantaneous success rate was below 0.5. Only the beginning data points (i.e., at 0–25 seconds) were similar enough to the training data to be deemed reliable. In Figure C-9a, the predictions in the red box fall within the acceptance criterion and should be accepted. However, in Figure C-9b, it is immediately evident that no training points were similar to the test sample. As no training samples existed in this region to support ML model predictions, the instantaneous success rate of these predictions was near zero. This example also illustrates the following dilemma: if a solution is correct but the data used to obtain that solution are wrong, should the solution still be accepted?

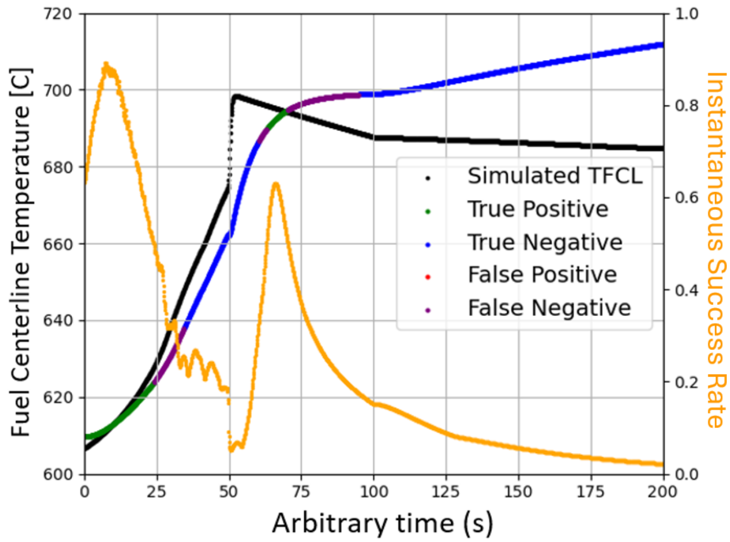


Figure C-8. Model predictions on dTest1, compared against instantaneous success rate.

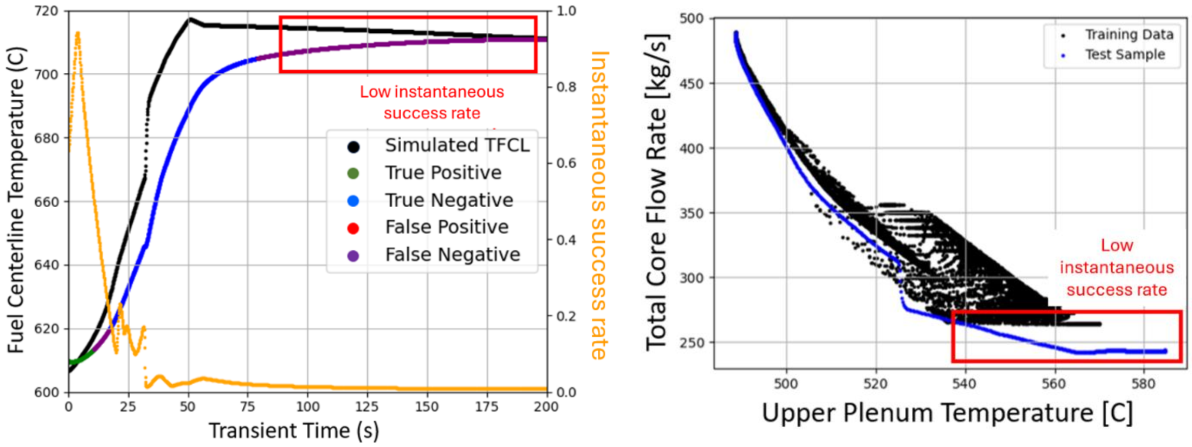
Table C-4 shows the performance metrics for the single transients seen in Figure C-8 and Figure C-9. The first column indicates the transient source dataset. For both settings, the same transient per dataset was selected when calculating the performance metrics. Columns 2–3 present the performance metrics for the DARE use case; column 4 provides the F_1 score result with the DARE framework; and column 5 gives the unfiltered F_1 score (denoted as F_{max}). F_{max} is based on the same equation as F_1, but is calculated only from model predictions and does not use DARE to filter the predictions. It serves as a baseline for comparing F_1 scores.

Note that peril is zero for all the test cases, indicating the model rejected all incorrect predictions and prevented any false positive identifications. However, for the Degrade and F_1 scores, the datasets significant differ with each other. More specifically, as the dTest2 transients significantly differ from those in the training set, models undergo significant degradation, as most of the predictions are rejected due to lack of training evidence. This does not imply that the model is incapable on dTest2. In fact, the F_{max} score suggests that the model may have some generalizability on the new dataset. However, the cost of that generalizability is an increased number of false positive predictions, which may be detrimental to the system operation. Another interesting observation is that for Setting (b), dTest1, the achieved F_1 score is actually higher than the F_{max} score because the number of false positive identifications in the filtered model is zero, whereas it is non-zero in the unfiltered test. As the F-score weighs false positives, its presence naturally leads to a lower performance score in the unfiltered case.

Table C-4. Performance metric for transients in Figure C-8.

Sample	**Setting (a)**			
	Degrade ($f_{degrade}$)	Peril (f_{peril})	F_1	F_{max}
dTest1	0.315	0	0.813	0.893
dTest2	0.980	0	0.040	0.817
	Setting (b)			

dTest1	0.098	0	0.949	0.893
dTest2	0.923	0	0.144	0.817



(a) (b)

Figure C-9. (a) dTest2 model predictions. (b) Scatter plot of training data vs. current test sample.

In the DARE use case, all transients from the three test sets were assessed to provide a more wholistic picture of model performance across the different datasets. Dataset details can be found in Table C-2. Recall that each test dataset contains 255 transient episodes, each with 2000 time steps—a total of 510,000 test samples. Setting (b) (see Table C-3) was used to gauge instantaneous success rate. Table C-5 presents the performance of the use case, Table C-6 presents the distribution of false and true, positive and negative predictions with DARE, while Table C-7 provides a control for an unfiltered ML model (no DARE).

dTest1 had the highest F_1 score. This is expected, as it had the most similarities with the training set. Most accepted predictions (383,643) fell within the acceptance criterion (see Table C-6). A total of 80,130 predictions fell outside the acceptance criterion were also correctly rejected. However, 46,227 predictions were rejected spuriously, contributing to a performance degradation of 10.8%.

dTest2 contained transients with a hotter T_{FCL} than in dTest1, though there was some degree of overlap, as seen in Table C-7. First, a model performance degradation of 68.4% is reflected in Table C-5, correlating to 348,923 out of 510,000 test samples being rejected despite falling within the acceptance criterion. On the other hand, no false positive predictions were accepted.

Lastly, dTest3 contained transients with a colder T_{FCL} than in dTest1. From Figure C-6, it may appear that dTest3 shares no overlap with dTest1; however, Figure C-10 makes it clear that the test input and training data share a degree of similarity. The tests in dTest3 demonstrate how a model may maintain generalizability within the interpolative length of training data. In Table C-6, 209,976 samples were correctly predicted with an instantaneous success rate over 50%.

Table C-5. Performance metrics on all test samples.

	Performance Metric			
	Degrade ($f_{degrade}$)	Peril (f_{peril})	F_1	F_{max}
dTest1	0.108	0	0.943	0.910

dTest2	0.684	0	0.480	0.881
dTest3	0.460	0	0.701	1.00

Table C-6. Positive-negative detection rate for predictions.

	Detection Rate				
	True Positive	True Negative	False Positive	False Negative	Total
dTest1	383,643	80,130	0	46,227	510,000
dTest2	161,077	0	0	348,923	510,000
dTest3	209,976	121,195	0	178,829	510,000

Table C-7. Detection rate for the unfiltered ML model.

	Detection Rate				
	True Positive	True Negative	False Positive	False Negative	Total
dTest1	429,690	N/A	80,310	N/A	510,000
dTest2	401,410	N/A	108,590	N/A	510,000
dTest3	388,805	N/A	121,195	N/A	510,000

The need for risk and reliability analysis in NAMAC is highly relevant and is applicable to other ML-integrated control systems. In [91], software risk analysis is identified as a necessary tool for NAMAC as the consequences of inappropriate actions may significantly affect reactor safety, especially when these actions are relevant to safety-related or safety-significant components. In this respect, DARE is a potential framework to address these concerns.

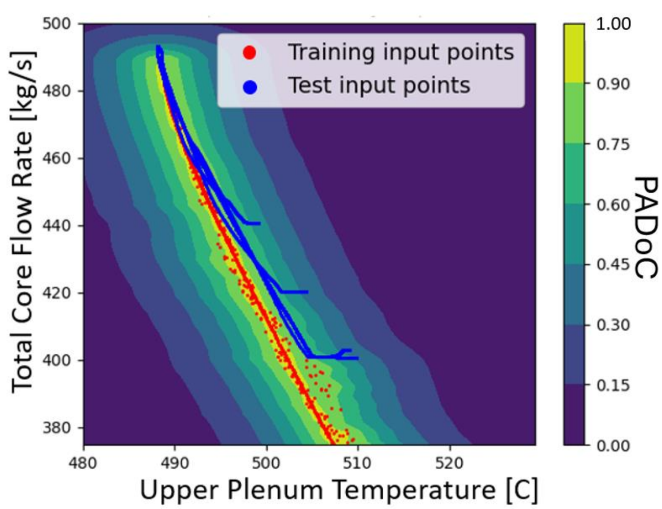


Figure C-10. dTest3 data plotted over a training data PADoC map.

This section discusses the implications and limitations of the DARE methodology when used to evaluate the reliability of individual ML model predictions. First, the need for reliability evaluation is pressing, as ML models are becoming more widespread in information and control applications such as DTs for active component health monitoring [92]. DARE leverages training data as inductive evidence supporting the reliability of ML model predictions. It has also been demonstrated as a real-time reliability evaluation methodology. As the method approaches reliability determination from a black box perspective, theoretically any data-driven memoryless neural network model can replace the feedforward neural networks in the presented use case (Figure C-3). However, certain limitations to the current methodology must be addressed.

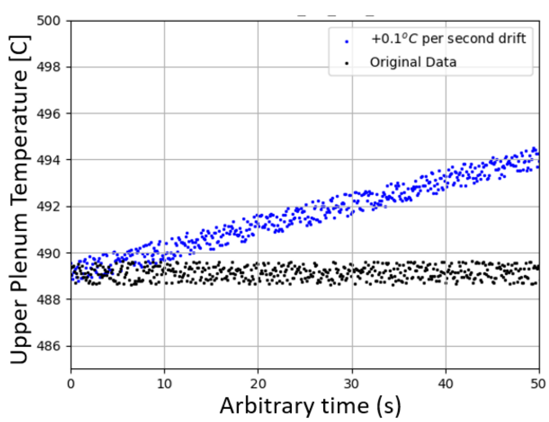
C-4. Discussion

Distributional Drift Detection

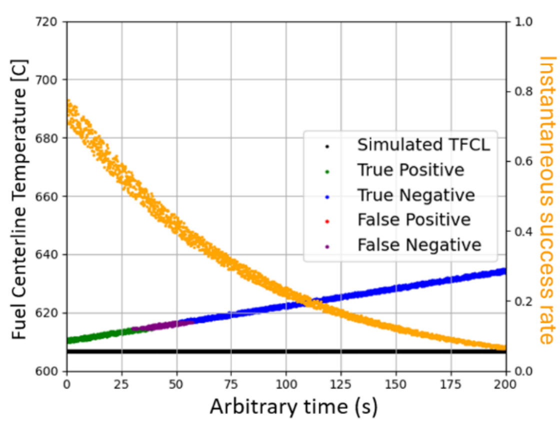
Distributional drift, defined as when the underlying distribution in the operational test data gradually or suddenly shifts, is a serious issue in ML models. It may be a result of sensor de-calibration over time, degradation, or displacement (e.g., a sensor being pushed), resulting in slightly different measured inputs. This can impact a model's prediction over time, increasing the number of false positive or false negative predictions.

DARE mitigates distributional drift by limiting application of the ML model to the domain and range of the training set. We assume that only one input drifts at a time. To demonstrate drift detection, two different cases are presented: (a) a $+0.1^{o}\mathrm{C}$ per second gradual drift on the upper plenum temperature at steady-state conditions (Figure C-11), and (b) a $+0.1^{o}\mathrm{C}$ per second gradual drift during a loss-of-flow transient. Figure C-11a shows the difference between the drifting (blue) and the baseline upper plenum temperature sensor (black). Figure C-11b presents the corresponding fuel centerline prediction with the drifted data. As the upper plenum temperature drifts, so does the fuel centerline prediction. As the degree of drift increases, the test sample distribution deviates further the training data, decreasing the instantaneous success rate of that prediction. In Figure C-11b, all fuel centerline temperature predictions that fall outside the criterion due to a drifting upper plenum temperature sensor are correctly rejected.

In Figure C-12, the drift occurs during a loss-of-flow transient from dTest1. In Figure C-12a, the drift is assumed to begin at the start of the transient and then accumulate as the transient progresses. In Figure C-12b, the sensor drift results in higher fuel centerline predictions; however, by implementing a DARE-informed filter, all predictions that fall outside the acceptance criterion are rejected. Table C-8 shows the detection rate and performance metrics for the transient in Figure C-12b.



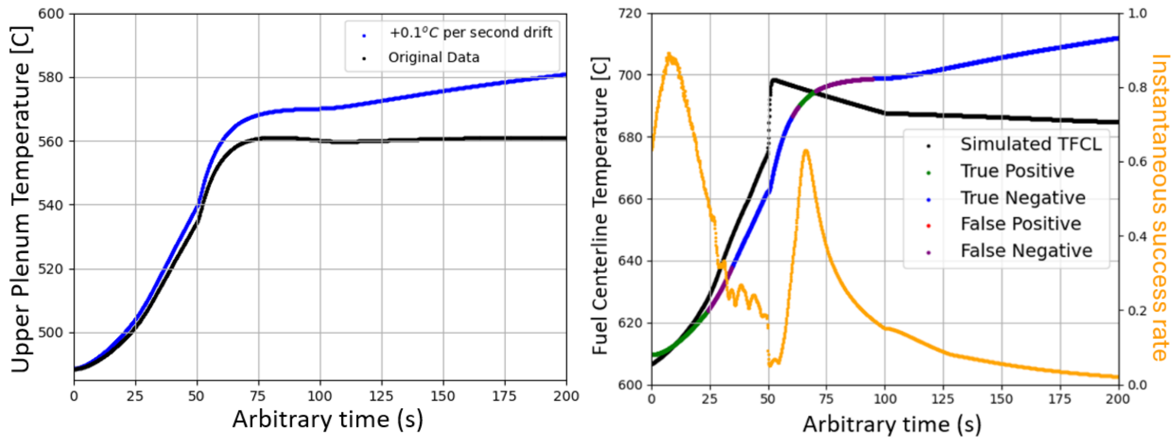
(a)



(b)

Figure C-11. (a) Drift in upper plenum temperature under steady-state. (b) Resulting fuel centerline prediction with drift and DARE filtering.

In general, as sensors drift away from the training data distribution, the degree of PADoC also decreases. At minimum, DARE can be used to filter out false positive predictions when drift has occurred. The two cases demonstrate how DARE can be used to detect, in real-time, distributional shifts on the input.



(a) (b)

Figure C-12. (a) Sensor drift in the upper plenum temperature. (b) The corresponding fuel centerline prediction.

Table C-8. Detection rate and performance metrics for the drift transient.

Detection Rate			
True Positive	True Negative	False Positive	False Negative
315	1,310	0	376
Performance Metrics			
Degrade ($f_{degrade}$)	Peril (f_{peril})	F_1	F_{max}
0.544	0.0	0.626	0.551

Optimization

As observed in Table C-7, the ML model can make false positive predictions that fall outside the acceptance criterion. When Setting (b) was applied in the use case, degradation ($f_{degrade}$) and peril (f_{peril}) rates of 0.108 and 0.0 were observed, respectively. The interpolation lengths for the upper plenum, the total core flow rate, and the fuel centerline temperature, in addition to the number of averaging samples (N), may be optimized so as to minimize the degradation and peril rates. Typically, as interpolation lengths decrease, the peril rate reduces while the degradation rate increases, and vice versa. For stakeholders concerned with safety and intent on minimizing false positive predictions, it may seem intuitive to first minimize peril, then minimize degradation. However, this may introduce instability in the peril metric, which may rise above zero in certain test samples. The situation is analogous to a delicate equilibrium. For instance, when using grid search optimization on dTrain1, the ideal interpolation lengths for the input features were found such that peril and degradation were minimized. Table C-9 presents these optimal lengths and determined performance metrics. When the same settings were applied to dTest1, a non-zero peril rate was found that corresponded to approximately 500 false positive predictions. In this respect, we argue that interpolation lengths should be chosen based on more human-relevant concerns than simply optimization efforts.

One way of interpreting interpolation length is as the margin available until a model starts making erroneous predictions. Implementing conservative interpolation lengths reduces model performance but enables rejection of edge cases that may be unsafe or otherwise irrelevant to the current training set.

From a regulatory perspective on DT development, the DARE methodology may be used to determine the degree of data coverage required to train a model. The interpolation length may specify the minimum grid separation distance for generating training data, such that sufficient training evidence is achieved when data is generated at increments shorter than the interpolation length.

Table C-9. Performance metrics using optimal interpolation lengths derived for dTrain1 and applied to dTest1.

	Performance Metric				
	Degrade ($f_{degrade}$)	Peril (f_{peril})	F_1	x_{d,v_T}	$x_{d,T_{UP}}$
dTrain1	0.059	0	0.968	26.8	22.8
dTest1	0.068	0.02	0.921	26.8	22.8

Limits on Model Generalizability

One significant concern when applying a filter to an ML model is the degradation that occurs when correct predictions are spuriously rejected (as in Table C-6). As ML models are intended to generalize over a range of cases beyond the training data, it seems counterintuitive to limit the model to the range of the training set. But consider the DT used in the case study. It demonstrated some degree of capability on both the hotter and colder transients. However, there were insufficient training data correlating to those predictions, making it difficult to differentiate in a human-interpretable manner whether those predictions were coincidental or, in fact, data informed. We argue that just because a model is capable, this does not necessarily imply that its predictions are trustworthy. Conversely, accepted predictions can be clearly traced back to the data points used to inform those predictions. From a trustworthiness perspective, DARE provides a more interpretable solution to prediction reliability.

That said, DARE is not a universal solution for all types of models. Physics-informed models [93, 94] for instance, may learn fundamental correlations such that a model's predictive capability extends beyond the training domain. In [95], as an example, models are trained based on heat conduction, Navier-Stokes, and Reynolds-averaged Navier-Stokes equations. During training, the model predictions are compared against these physics equations to determine loss. The resulting model learns the fundamental physics behind the thermal hydraulics problem and can be applied to a range of scenarios. In such instances, implementing a DARE framework would severely degrade performance, as the model is capable beyond just the training set.

Prediction Reliability

In this work, the term *reliability* was used extensively to describe whether an ML model can achieve its functional goals. From a conventional standpoint, reliability is used to measure the "the ability of an item to perform as required, without failure, for a given time interval, under given conditions" [96, 97, 98, 99]. However, it is difficult to quantify a reliability value due to various implementation and deployment differences associated with ML models. Rather than a set-and-forget mentality of reliability, we discussed instantaneous success rate (C.6), used to describe the probability that a model can predict correctly on a single input instant at any given instant. The instantaneous success rate is informed by two critical aspects part of the ML development process: the post-training model performance under the closed-world assumption, and the validity of the training dataset to the operational environment under the closed-world assumption.

Instantaneous success rate can be adapted to work with conventional reliability equations but should be used cautiously. First, recognize that ML models operate per demand on the input and is not subject to time-dependent degradation. As such reliability equations are posed as per demand rather than per unit time. Second, the instantaneous success rate is equal to one minus the hazard rate, λ, defined as the instantaneous rate of failure on demand [99, 97]. The derivation of the reliability equation is thus as follows stemming from [99, 97]:

$$\lambda = 1 - r = 1 - Q_X \cdot \mu \tag{C.11}$$

$$\mu = \frac{1}{N} \sum_{i=0}^{N} K(\vec{x}', \vec{x}_i) \tag{C.12}$$

$$\lambda = -\frac{1}{R(u)} \cdot \frac{d}{du}\big(R(u)\big) \tag{C.13}$$

$$\int_0^u \lambda du = -\int_0^u \frac{1}{R(u)} \cdot \frac{d}{du}\big(R(u)\big) du \tag{C.14}$$

$$\lambda u = -\ln|R(u)| + \ln|R(0)| \tag{C.15}$$

where $R(u)$ is the reliability function with respect to the number of model input demands, u. Note that the hazard rate is independent of the number of demands as the function for PADoC (μ) does not account for the number of inputs that came before it. Further, if we assume that the reliability of a model at zero demands is the training performance (i.e., $|R(0)| = Q_X$), then:

$$R(u) = \exp(-\lambda u + \ln Q_X) \tag{C.16}$$

$$R(u) = \exp(-(1 - Q_X \mu)u + \ln Q_X) \tag{C.17}$$

The PADoC can only be calculated when a new test sample is received, which makes future determination of reliability difficult. A preliminary value for model reliability can be determined by estimating the expectation and variance of PADoC through Equations (C.18) and (C.19).

$$\mathbb{E}(\mu) = \frac{1}{M} \sum_{j}^{M} \frac{1}{N} \sum_{i=0}^{N} K\big(\vec{x}_j', \vec{x}_i\big) \tag{C.18}$$

$$\sigma^2 = \frac{1}{M-1} \sum_{j}^{M} \left(\frac{1}{N} \sum_{i=0}^{N} K\big(\vec{x}_j', \vec{x}_i\big) - \mathbb{E}(\mu) \right)^2 \tag{C.19}$$

where M is the number of samples evaluated in the training dataset. A distribution may also be used if the probability of selecting an in-distribution sample is known. Otherwise, Equation (C.17) gives the full equation for reliability of a ML model and can be used to calculate the reliability at any number of demands. However, this equation should be used with caution as it can lead to overly aggressive and unstable estimates on model unreliability. Observe the hypothetical condition where $Q_X = 1$ and $\mu = N(0.95, 0.05)$. The first condition implies that the model's training performance is perfect, and the second condition assumes that test points are highly supported by training data with a PADoC of at least 0.95.

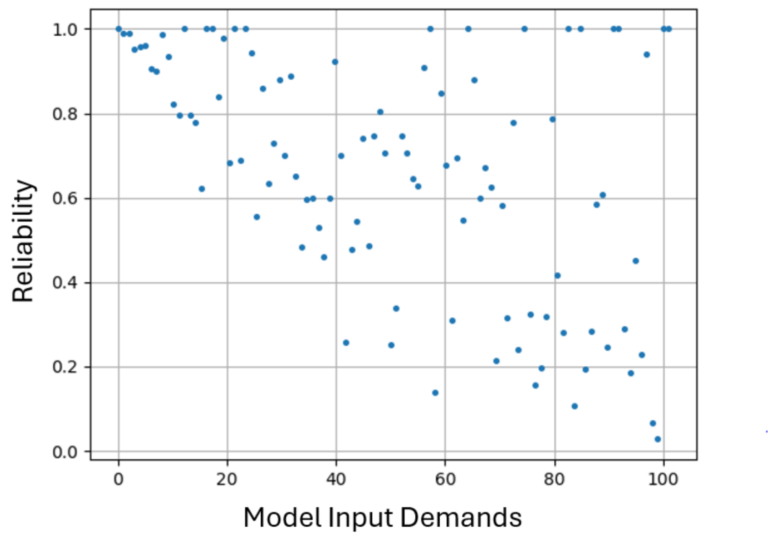


Figure C-13. Estimated reliability using Equation (C.17).

There are two problematic issues with this reliability plot (Figure C-13). First, the plot of the function is highly erratic, and a reliability trend line cannot be determined without significant confidence intervals. The reliability of the model can be 100% at any point regardless of the number of demands. This makes prediction of reliability difficult as a consistent trend cannot be determined. Secondly, the reliability of a model generally degrades with input demand which is generally untrue as software does not degrade over usage.

An alternative estimation of model reliability can be derived from the instantaneous success rate via Equation (C.6). Once again, assume that $Q_X = 1$ and $\mu = N(0.95, 0.05)$. The reliability over M demands using Equation (C.6) is shown in Figure C-14. The trend of the reliability function can be estimated via the expectation of the PADoC as shown in Equation (C.20). This equation is invariant to the number of demands and is unchanging with usage. In addition, Equation (C.20) can reflect the expected reliability of a model's prediction by incorporating an expectation on the test samples drawn from the training data set, the basis for the closed-world assumption.

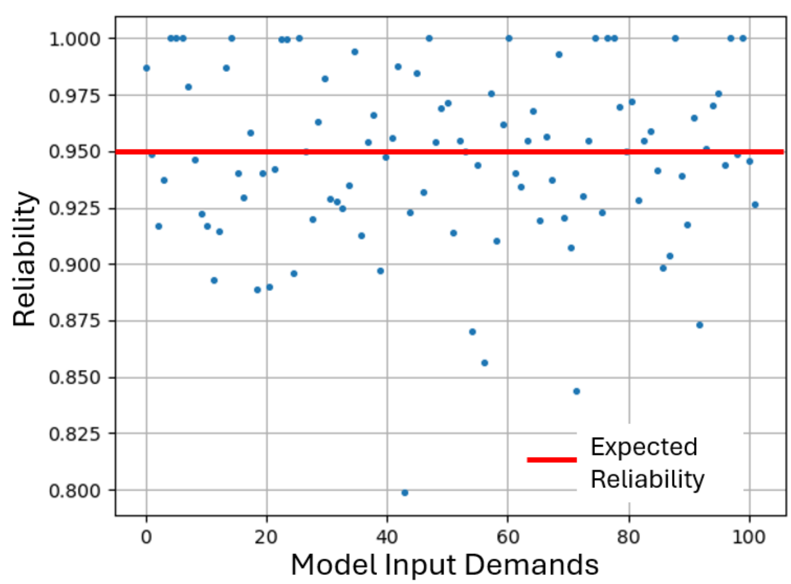


Figure C-14. Estimated reliability using Equation (C-8).

$$R = Q_X \cdot \mathbb{E}(\mu) \tag{C.20}$$

The authors acknowledge that this formulation of reliability may be incomplete as it only incorporates how training performance and data coverage contribute to calculating reliability. As previously discussed, the ability for a model to achieve function is highly dependent on the input feature. Drift and changes in the operational conditions of model can alter the reliability of a model's prediction significantly. These factors should be accounted for dynamically in the reliability equation. However, given only training data, only a preliminary estimate of the reliability of a model can be obtained. Therefore, it is still recommended to utilize PADoC within the DARE framework to evaluate the likelihood a model can achieve its predictive function in real-time scenarios.

Potential Application

DARE can be easily adapted to detect sensor drift in input features. Specifically, if sensor input begins to drift away from the training data, the PADoC (and subsequently the instantaneous success rate) of the predictions will decrease. As training data are intended to emulate an operational condition, any decrease in PADoC may signal an error in the system. In such cases, PADoC may be used as a direct health indicator for component monitoring and may signal when sensors require calibration or maintenance.

DARE may also be combined with software reliability analysis to determine when a system is operating in an unfamiliar regime. In such cases, PADoC serves as a direct measure of whether an ML model can achieve its intended function and may be used to calculate risk whenever a new test sample is received.

C-5. Summary of Work

This work presents a novel framework for determining the reliability of model predictions in relation to the training dataset. Data points in the training dataset are used as inductive evidence for determining when new test samples are in-distribution vs. OOD. A case study was presented to show how DARE can be integrated with data-driven ML-integrated control systems for loss-of-flow transient diagnosis. We demonstrate how the DARE framework can be used to reject unreliable predictions, whether through sensor drift or out-of-distribution, made by an ML model to enhance performance.

We acknowledge that DARE is not a panacea for addressing trustworthiness issues in ML models, however, through DARE we can derive highly interpretable results based on known training data evidence. Future work includes assessing DARE on other types of out-of-distribution failures and development issues such as high dimensionality of input features, varying feature importance over time, and validating DARE on other ML architectures.