NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A SYSTEM-THEORETIC APPROACH TO ENGINEERING ASSURANCE FOR ARTIFICIAL INTELLIGENCE SYSTEMS

by

Eugene D. Williams

September 2023

Thesis Advisor: Joshua A. Kroll
Co-Advisor: Logan O. Mailloux

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** September 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
A SYSTEM-THEORETIC APPROACH TO ENGINEERING ASSURANCE FOR ARTIFICIAL INTELLIGENCE SYSTEMS

5. FUNDING NUMBERS

6. AUTHOR(S) Eugene D. Williams

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Predictive systems that incorporate neural network models have been deployed in both non-safety-critical and highly safety-critical domains. When the models fail to perform as desired, it is often difficult to identify a root cause. In domains where failures might cause irreparable damage, or harm to life or property, steps must be taken to assure those using these systems that risks have been mitigated by thoughtful analysis during design. This thesis demonstrates the use of system-theoretic process analysis (STPA) as a repeatable approach for selecting and calibrating machine learning development actions to provide assurance during the machine learning development life cycle (MDLC). STPA is a system analysis method that identifies component hazards arising from component-level interactions in safety-critical systems. In this research, STPA is used to assess machine learning development safety, with respect to responsible artificial intelligence (AI) principles, for a system that utilizes a classification model to detect maritime vessels based upon audio signatures. As an outcome of the analysis, recommendations are made that can proactively guide the AI design process so that decisions within each phase of the life cycle are explainable. It is demonstrated that by applying this approach, AI systems are more reliable and safer to deploy.

14. SUBJECT TERMS traceability, real-time data, deep learning, artificial intelligence, AI, supervised learning, active learning, system-theoretic process analysis, STPA, machine learning development life cycle, MDLC	**15. NUMBER OF PAGES** 75
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A SYSTEM-THEORETIC APPROACH TO ENGINEERING ASSURANCE FOR ARTIFICIAL INTELLIGENCE SYSTEMS

Eugene D. Williams
Civilian, DMDC
BS, Georgia Institute of Technology, 2001

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2023

Approved by: Joshua A. Kroll
Advisor

Logan O. Mailloux
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Predictive systems that incorporate neural network models have been deployed in both non-safety-critical and highly safety-critical domains. When the models fail to perform as desired, it is often difficult to identify a root cause. In domains where failures might cause irreparable damage, or harm to life or property, steps must be taken to assure those using these systems that risks have been mitigated by thoughtful analysis during design. This thesis demonstrates the use of system-theoretic process analysis (STPA) as a repeatable approach for selecting and calibrating machine learning development actions to provide assurance during the machine learning development life cycle (MDLC). STPA is a system analysis method that identifies component hazards arising from component-level interactions in safety-critical systems. In this research, STPA is used to assess machine learning development safety, with respect to responsible artificial intelligence (AI) principles, for a system that utilizes a classification model to detect maritime vessels based upon audio signatures. As an outcome of the analysis, recommendations are made that can proactively guide the AI design process so that decisions within each phase of the life cycle are explainable. It is demonstrated that by applying this approach, AI systems are more reliable and safer to deploy.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction **1**

1.1 Problem Statement 6

1.2 Research Questions 6

1.3 Preview 7

2 Background **9**

2.1 Active Learning 9

2.2 Federated Learning 11

2.3 Measuring ML Model Performance 11

2.4 Bayesian Deep Learning (BDL) 13

2.5 Uncertainty 15

2.6 Acquisition Functions 18

3 Methods **19**

3.1 Research Objectives 19

3.2 Systems-Theoretic Process Analysis (STPA). 20

3.3 The System of Interest 23

4 Results **25**

4.1 STPA Step One: Determine Purpose of the Analysis 25

4.2 STPA Step Two: Define Losses and Hazards 28

4.3 STPA Step Three: Build Control Structure 32

4.4 STPA Step Four: Find Unsafe Control Actions 41

4.5 STPA Step Five: Perform Loss Scenario Analysis 46

4.6 STPA Step Six: Propose Constraints and Restraints 48

5 Conclusion **51**

List of References **53**

Initial Distribution List **57**

List of Figures

Figure 1.1 Machine Learning Development Lifecycle 2

Figure 2.1 Epistemic and Aleatoric Uncertainty 17

Figure 3.1 System Theoretic Process Analysis Steps 21

Figure 4.1 System of Interest Diagram 27

Figure 4.2 System Control Structure. 33

Figure 4.3 Sonar Detection System Control Structure. 34

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

Table 4.1 STPA System-Level Losses 28

Table 4.2 STPA Hazardous System States 29

Table 4.3 Hazards and Losses Aligned to RAI Principles 30

Table 4.4 STPA Controlled Process - Data Decisions 35

Table 4.5 Data Decisions - Descriptions 36

Table 4.6 STPA Controlled Process - Model Decisions 37

Table 4.7 Model Decisions - Descriptions 38

Table 4.8 STPA Controlled Process - Monitoring Decisions 40

Table 4.9 Monitoring Decisions - Descriptions 41

Table 4.10 UCA Analysis - Data Decisions 43

Table 4.11 UCA Analysis - Model Decisions 44

Table 4.12 UCA Analysis - Monitoring Decisions 46

Table 4.13 STPA Loss Scenarios 47

Table 4.14 STPA System Constraints 49

Table 4.15 STPA System Restraints 50

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

ACM	Association for Computing Machinery
AI	Artificial Intelligence
ANI	Artificial Narrow Intelligence
AGI	Artificial General Intelligence
ASI	Artificial Super Intelligence
BDL	Bayesian Deep Learning
BNN	Bayesian Neural Network
CA	Control Action
CAST	Causal Analysis based on STAMP
CS	Causal Scenario
DL	Deep Learning
DOD	Department of Defense
DST	Decision Support Tool
FMEA	Failure Mode and Effects Analysis
ML	Machine Learning
MDLC	Machine Learning Development Lifecycle
ML/DL	Machine Learning / Deep Learning
NIST	National Institute of Standards and Technology
RAI	Responsible Artificial Intelligence

SDLC	Software Development Lifecycle
SOI	System of Interest
STAMP	Systems-Theoretic Accident Model and Processes
STPA	Systems-Theoretic Process Analysis
UCA	Unsafe Control Action

Acknowledgments

Special thanks to Professor Joshua Kroll for the time and effort poured into me to communicate the importance of traceability in AI. The insights gleaned from his lectures and time spent discussing the impacts of socio-technical systems on the state of humanity have been immensely enlightening.

I would also like to thank Commander Ed Jatho, PhD for his guidance and support in building out a justification for developing assurance cases for AI systems and for researching alongside me, exploring STAMP, CAST, and the potential for STPA to be a useful tool in this study. I am also grateful that we were able to collaborate as co-authors while performing STPAs similar to that performed here.

Along with Commander Jatho, I thank Lieutenant Colonel Logan Mailloux, PhD whose insight and STPA subject matter expertise informed our analysis and kept us on track while reviewing various systems. I'm grateful for Professor Patrick McClure who, in a single Bayesian Neural Networks class, provided some of the clearest teaching on the intersection of machine learning and mathematics that I've ever experienced; and to Professor Marko Orescanin for introducing me to the challenges of deep neural networks when applied to the sonar system analyzed in this thesis.

Additionally, I thank the Defense Human Resources Activity of the Department of the Under Secretary of Defense for Personnel and Readiness for allowing me to participate in this master's program. The insights gleaned while studying at this great institution (NPS) will aide in my ability to add value and better direction in the programs I support and lead.

Lastly, I thank my wife for being so patient over these three years; her graciousness in dealing with the late nights and hours added to the workday in order to complete this program. You are the love of my life and I am grateful to God for your love toward me.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Software engineering has evolved over the decades since its recognized inception in the 1960's [1]. The practice has played a critical role in shaping and transforming the world. Misra et. al. discuss how this evolution is marked by the transitioning from delivering software using manufacturing practices (waterfall) to Agile approaches, where changes are defined, developed, and delivered through iterative cycles [2]. Historically, applications have been written with requirements that can be directly mapped to functionality. When the state of a system is not consistent with the expectation of stakeholders, taking into consideration the inputs received, system designers are able to incrementally trace those failed outcomes to a failure in the design of a particular method, class (for object-oriented models), or data object used to inform the system. Classically, manual testing drives this tracing activity, a practice that is prone to error due to inconsistency or other issues associated with procedural integrity. In modern practices, automated test cases mitigate those issues and improve traceability by ensuring consistency in test procedures: unit and integration tests created during a test-driven software development cycle are accompanied by functional tests used during subsequent verification and validation cycles, executed as part of an automated build process or pipeline before the application is delivered to an end-user. This is referred to as continuous integration/continuous delivery, and is a key aspect of DevSecOps practices [3]. These pipelines deliver capability to operational environments—dynamic allocation of infrastructure resources, configurations automatically applied, continuous security assessment, and quick implementation of changes to the system and environment. As enhancements in software development and delivery emerge, expansion of the domains in which software is used to solve organizational and mission problems has likewise occurred.

This trend is not isolated to traditional software applications, but also applies to software applications that utilize machine learning (ML) in their core function. Many of these AI systems incorporate neural networks to maximize precision and accuracy. When implemented, these systems serve to either augment or replace human effort with minimal error, consistency, and, when compared to a human counterpart, with more accurate results, and leading to greater efficiency [4]. Wan et. al. discuss how challenging it is, however,

to operationalize these systems [5]. For example, current machine learning/deep learning methods also incorporate learning principles (continual, active, reinforcement, and federated learning) designed to improve the performance of core models through iterative cycles by incorporating new data and/or aggregating model effectiveness.

The Machine Learning Development Lifecycle (MDLC) is a process, similar to the software development lifecycle, which provides a framework for planning, developing, testing, and delivering software [6]. However, although each of these lifecycles incorporate a planning or analysis phase, a testing, and a delivery phase, MDLC expands the development process so that attention is given to the iterative data preparation and model development activities. In [7] these stages are explained in detail, while also explaining the role of data preparation, model refinement, and model evaluation in the overall process. Figure 1.1 depicts these activities.

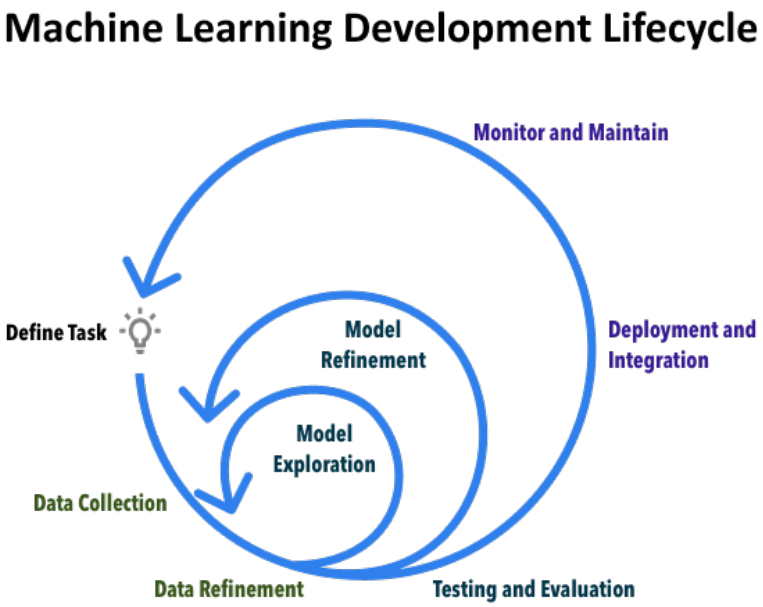


Figure 1.1. The Machine Learning Development Lifecycle (MDLC) consists of seven primary activities/phases in three concentric cyclical activities, each integrated with the other in a cyclical manner. This is to ensure that model improvements and performance measurements can occur in the context of other business requirements or concerns. Adapted from [8].

Whereas the traditional software development lifecycle (SDLC) aims to introduce new or improved functionality into software through iterative planning, engineering, building, validation, deployment, and monitoring phases, the goal of the machine learning development lifecycle is to iteratively improve performance of a predictive model given a dataset which may evolve over each iteration. In SDLC, design and engineering are generally guided by changes in requirements or the need to correct observed defects so that end-users are satisfied. These iterations may end when all functionality is implemented. In MDLC, design and engineering are also guided by requirements, yet the focus of each iteration is to produce a version of a model that better predicts outcomes than prior versions. These iterations continue as new data is obtained and are, essentially, ceaseless since a perfect predictor is unattainable.

Machine learning development requires guardrails to ensure unsafe practices are not employed [9]. Data decisions require guidance; processing steps must be traceable and explainable [10]. Questions must be asked, such as, "is there enough data to achieve the goals established for the system?" and "are we able to reproduce the results observed on repeated trials?" [11] These questions are important in traditional software system development, and all the more critical in ML systems [5]. The set of failure modes described by Selbst et al. [12] are those that arise when ML designers do not take into consideration the problems which arise when technical solutions intersect with society. These intersections occur in contexts such as justice systems where law enforcement rely on predictive software to guide investigations or verdicts [13] or in prescription drug monitoring systems discussed in [14] that can be used to limit access to certain patients based upon ML-driven decisions. Kleinberg et al. [13] argue for considering the risk of, not only self-contained ML products, but that of incorporating them as portable solutions, usable in larger systems whose requirements not only stem from hardware or software needs, but also from matters pertaining to humanity and community. These are referred to as socio-technical systems. For these systems, a change in social constructs may lead to unexpected societal or mission impacts, even though the application may adhere to functional requirements. Similarly, when operators of the system transition to new duties, transferring operational responsibilities to new personnel, a complete understanding of known insufficiency is not always communicated. When these new operators expand use of the system for purposes which it was not intended, undesired outcomes may arise.

MDLC decisions should be informed by possible ethical and safety consequences. [15] highlight the difficulty of achieving this since even humans have difficulty reconciling ethical norms. Feature selection ought to be guided by these concerns and the potential hazards that those features induce. Belitz et al. [16] propose an approach in feature selection which elevates fairness of features to that of accuracy in performance. This work also expresses the importance of scrutinizing optimization procedures.

The deployment, inference, and evaluation phases of MDLC are those where various learning techniques are considered to enhance the system. Each of these techniques present their own challenges: Continual and Active Learning can introduce Catastrophic forgetting [17]; Reinforcement Learning can introduce rater bias (not all ratings are giving the same care) [18]; Federated learning processes can be susceptible to contamination, privacy concerns, data integrity, or other security risks [19].

With a mission-driven desire to build pre-trained systems capable of generating outcomes that fulfill the expectations of users that will rely upon them, there is an inherent risk that they may produce unexpected results which cannot be accounted for or traced by root cause analysis. This is due to the fact that their architectures rely upon obfuscated calculations during training where the primary goal is to improve the accuracy and/or performance of a classification, transformation, or regression task (modeling the relationship between a set of variables through an equation). Wan et al. [5] demonstrate in their survey of software development practices that, although reproducing test results is not a challenge with traditional software systems, ML systems do not provide the same assurances. Based upon outcomes of their survey this is due to the stochastic measures taken during design. Respondents indicated that reproducibility is challenging because of design practices that introduce randomness in data, observation order, weight initialization, and optimization parameters. As a result, it is difficult to assess or mitigate risk.

When outcomes cannot be reproduced during testing, the system is unpredictable. Unpredictable systems are difficult to control since the control measures that are generally implemented to affect undesired states may not prevent unknown states. When unknown, ungoverned states arise, those controlling the system may not have mechanisms for preventing damage or harm. This concern is compounded when users over rely upon AI system components that are not adequately understood; as humans are taken out of the control loop

leading to scenarios where problems are not promptly identified. Indiscriminate deployment of AI to perform self-governing tasks can lead to a lack of awareness of what a system does and why, particularly if managers or stakeholders of these systems grow to inherently trust the AI after a period of loosely monitored operational use [20]. Inappropriate expansion is also a concern. This occurs when use of an AI component is migrated into domains that were not intended by the creators; Selbst et al. [12] refer to this as "The Portability Trap." If those operating the system have limited knowledge of its purpose or assume capability based upon an incomplete understanding about its purpose, there is a risk that the AI might be deployed to perform tasks which may be similar to its design intent, but, ultimately, for which it is not suited and thereby could present ethical, social, or physical harm.

This thesis focuses upon achieving the principles of responsible AI (RAI), a field that has evolved into a discipline of its own [21]. It is motivated by a desire to identify a repeatable process for implementing RAI principles in the design and deployment of DOD AI systems so that the safety, security, and integrity of the DOD mission is maintained. In an era where designers of AI systems focus, primarily, on performance, RAI practitioners evaluate AI solutions to assess whether their implementations can be trusted. [21] [22], where the aim is to develop standards which prioritizes accountability over accuracy. Although multiple attempts have been made to summarize the goals of RAI into a set of principles, the Department of Defense, has chosen five which apply to both battlefield and non-combat applications: 1) Exercise proper judgment during development, deployment and use [responsible]; 2) take measures to reduce unintended bias [equitability]; 3) incorporate a means of auditing methods, data sources, design procedures and documentation [traceability]; 4) implement measures to ensure testing can be performed through all stages of the system's life-cycle [reliability]; 5) incorporate mechanisms to detect, avoid, and halt unintended consequences when the system is operational [governability] [9].

1.1 Problem Statement

Standards for measuring the compliance of AI systems through the use of DOD RAI demands are not commonplace, though there are efforts underway to make them so. Current deficiencies with measuring compliance stem from the reality that agencies are eager to quickly utilize artificial intelligence and are accelerating AI adoption without waiting for policies to evolve. The machine learning development lifecycle phases are focused on engineering models that perform well—optimizing an objective function to generate outcomes (predictions) that satisfy a statistical ideal [12]. The MDLC conclusions reached during the data and model refinement phases are not always easily traceable. It is not directly apparent how the amalgamation of adjustments to the data or the model contribute either detrimentally or favorably to a system's behavior. For this reason, to prevent undesired outcomes that emerge as a result of rapid implementation, a framework is needed which can be used as part of the MDLC to help ensure responsibility and assurance are engineered into DOD AI systems.

1.2 Research Questions

This work provides a case study analyzing the development of an ML model used to identify maritime targets (biological or man-made) utilizing passive sonar [23], henceforth referred to as the system of interest (SOI). Within the study, the thesis examines the feasibility of aligning DOD RAI principles with the MDLC process, while affording the practitioner (the AI system designer) the necessary space to deliver a valuable product. The fundamental intuition is that ML developers operating within an MDLC prefer to discover problems as soon as possible and, within DOD, want to ensure that the AI system operates in a lawful and ethical manner. The steps involved in the analysis aims to answer the following:

- What considerations are important when evaluating DOD's development of Artificially Intelligent systems? This understanding is gleaned, in part, from the concerns identified in the Responsible AI Strategy memo and implementation pathway [9].
- How can the MDLC practitioner effectively trace unacceptable losses, hazardous system states, and unsafe control actions to activities performed during MDLC?
- What are the tradeoffs, recommendations, and auditable criteria during ML development that leads to effective governance for these operational AI systems?

1.3 Preview

This thesis includes four additional chapters, organized as follows: Chapter 2 provides an explanation of core concepts, necessary to understand when reviewing the methodology and results. The methodology chapter, Chapter 3, introduces System Theoretic Process Analysis (STPA) and a target system (the system of interest) that utilizes a machine learning model to identify maritime vessels based upon their emitted sonar signature. Chapter 4 provides the outcome of a case study where the steps of STPA are performed against the system of interest. Here, the purpose analysis, losses and hazards, control structure, unsafe control actions, loss scenario, and proposed constraints and restraints are provided. The final chapter, Chapter 5, concludes the research by summarizing how RAI goals are addressed in the outcomes of the analysis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Background

The system of interest (SOI), a passive sonar maritime target classification system, for this research incorporates learning strategies in the design and enhancement of the classification models that require review. This work investigates the learning process to discern whether certain activities performed during model design and development introduce unsafe practices or enable an environment where hazards can manifest. Although their work does not explicitly address or engage RAI ideals, research by Fischer et. al. [23], exploring the SOI, experiments with concepts outlined in this chapter, while attempting to build models that perform well at predicting the presence of a particular vessel class based on underwater soundscapes. This paper extends upon that work by aligning the following ML principles with RAI principles.

In the development of the SOI, two common learning approaches are implemented: active learning and federated learning. Performance of the system is assessed by measuring accuracy, precision, and recall. The prediction models are devised using Bayesian Deep Learning methods. These methods rely on uncertainty estimation and applying procedures to reduce uncertainty.

2.1 Active Learning

Active machine learning involves updating an ML model based upon information provided by a user or other system with knowledge (an oracle) of the labels for new data being incorporated into the training process. An ML system can be improved (i.e., decreasing uncertainty regarding a particular prediction/classification) when otherwise unknown observed data is labeled and added to the learned dataset. Active learning reframes the modeling paradigm, adding an "update" step to incorporate new data into a trained model. To achieve successful active learning, it is important to decide which data to include in the update process. This involves deciding what new data is beneficial for the model's knowledge base and which is rejected. This approach is especially useful in systems where the initial dataset is small and there is high epistemic uncertainty [23], [24].

There are three common, though not exhaustive, approaches for capturing new data in active learning [24]. In the first, *Membership Query Synthesis*, the system generates new data samples and presents them to the oracle for labeling. This approach is useful when it is difficult or costly to capture new samples in the wild (i.e., the unstructured and uncontrolled environment where the model is deployed). The second approach, *Pool-Based Sampling*, involves selecting a sample that lies within a pool of unlabeled samples based upon a ranking algorithm that quantifies its benefit to the model. A final approach, *Stream-Based Selective Sampling*, the approach that aligns more closely with the SOI, involves gathering a sample observed while the system is in operation (samples being streamed to the system) and determining whether to submit to the oracle for labeling. That determination is based on identifying whether including the sample would be beneficial or most informative. The informativeness of the sample is determined via a querying strategy. [25] discusses some common approaches. The simplest and most common is *Uncertainty Sampling*. In short, this Bayesian approach involves selecting the sample for which the predictor is least confident. The general equation used for classification tasks is:

$$x^*_{LC} = \underset{x}{\text{argmax}} \, [1 - p(\hat{y} \mid \theta, x)] \tag{2.1}$$

This is to say that the least confident sample x^*_{LC} can be identified by evaluating the complement of the probability p of the most probable prediction ($\hat{y}$) given the model, represented by its parameters θ. Bouacida and Mohapatra [25] identify other approaches that do not disregard other less probable class labels. Another querying strategy, *Query-by-Committee*, requires multiple versions of the model predicting the class label for a particular sample (see [26] for more detail on this approach). The most informative sample is that which the *committee* disagrees about most. Other query strategies include *Expected Model Change*, which involves selecting the sample that most significantly affects model performance, and *Expected Error Reduction*, measuring how introducing the sample would reduce generalization error in expectation [27]. Both these approaches can be expensive (require undesirable time and resource allocation) and are more theoretical than practical. *Variance Reduction* is a less expensive approach which involves selecting the sample which reduces generalization error by way of minimizing variance. None of these strategies, however, are as efficient as the first, uncertainty sampling. See [25] for a detailed discussion.

2.2 Federated Learning

Federated learning (FL) is an approach whereby a version of the model (the base model) is deployed on various nodes in disparate environments [19]. This decentralized approach to learning is particularly useful in contexts where the nodes are actively capturing new data that would be beneficial to the entire cluster once the information is aggregated and incorporated into the base model. By consolidating their knowledge (each node delivering their updates to a centralized aggregation system), overall performance is improved [28]. In the SOI implementation of FL, the aggregation involves consolidating weights from each decentralized model to achieve a new set of parameters that can be deployed to the edge nodes during an update process. This is, in general, the process involved in *Weight-Shared Federated Learning* [29].

2.3 Measuring ML Model Performance

For the DOD, trusting ML-enabled systems is not a matter of merely trusting how well the technical implementation performs based on traditional performance metrics, but how governable the system is, how well the user community is trained on the system's capabilities or known deficiencies, and how well the warfighter can accomplish the mission in a manner consistent with the ethical principles that guide the Nation [9]. In classical performance analysis, stakeholders deem outcomes favorable when several metrics demonstrate acceptable results. Accuracy is one of these indicators. For classification problems, accuracy is measured by simply calculating the ratio of correctly classified examples (i.e., true positive and true negative) to the total number of examples classified (whether true positive, true negative, false positive or false negative):

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \tag{2.2}$$

Precision is also a common measure of performance, calculated as the ratio of true positives to the sum of true positives and false positives:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{2.3}$$

Recall measures how well a model can identify true positive instances while exhibiting fewer false negatives. Recall is measured by taking the ratio of true positives to the sum of true positives and false negatives, a criterion useful when false negatives are more detrimental than false positive predictions, such as safety-critical systems where a false negative would prevent a safety measure from activating (i.e., braking, turning, slowing, or halting):

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2.4}$$

Yet another measurement of performance is the F-score (or F1 score) – a measurement used to quantify the balance between precision and recall, the harmonic mean of the two (2 x (precision x recall)/(precision + recall)). The calculation resolves to a value between 0 and 1. Ideally, the computation would be greater than .5 or 50%. Closer to 1 or 100% indicates good harmony.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision\ + Recall} \tag{2.5}$$

Great emphasis is placed on optimizing these metrics in aggregate. One can infer how challenging it is, in the pursuit of improving them, to incorporate the added work of tracking their incremental changes as data augmentation, model training, parameter tuning, and other neural network design activities are repeatedly performed. However, accountability is a chief objective in responsible AI – those designing systems must do so in a manner that harmonizes improvements in performance with a continual aim of accountability in design, recognizing that doing so will engender greater trust in the system and improve explainability. The more explainable an AI solution is, the more likely that RAI goals are achieved.

2.4 Bayesian Deep Learning (BDL)

When traditionally training neural networks, a single value is learned for each model parameter using the training data [30]. This approach leads to a deterministic model. This is not optimal since datasets do not, generally, represent all potential outcomes that a model would need to predict. Bayesian deep learning introduces a method of incorporating uncertainty into the parameter estimation process. This provides a means of quantifying uncertainty so that different versions of the model can be compared not merely based upon their ability to make accurate predictions but also based on their lack of certainty regarding samples for which a reliable prediction cannot be made. In Bayesian neural networks, the goal is to learn the probability that particular weight values would be used to model a particular set of data after new information is obtained—this is referred to as the posterior distribution. Using Bayes theorem, this posterior is evaluated by utilizing prior knowledge and a likelihood, namely:

$$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \tag{2.6}$$

In this equation, w is a model's weight values. $\mathcal{D}$ represents the observed data (a non-infinite set of x, y data pairs). These data pairs are the training samples used to generate the model.

$$\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\} \tag{2.7}$$

$p(w)$ is the prior belief (a probability distribution) about the model weights before data is observed. $p(\mathcal{D}|w)$ is the probability distribution (or likelihood) of observing a particular data set given particular weight values. $p(w|\mathcal{D})$ is the posterior probability that is evaluated using Bayes' theorem; it is found by using the prior belief and the observed data [30].

2.4.1 Maximum Likelihood and Maximum a Posteriori Estimation

Maximum Likelihood Estimation (MLE) is a procedure for finding a point estimate, a single value, that maximizes the likelihood that data will be fit by particular weight values $p(\mathcal{D}|w)$ [31]. This is to say, it is the maximum probability that data will be observed when particular model parameter weights are provided. Determining MLE is achieved by

maximizing the combined probabilities of achieving a certain outcome given a specific set of parameters:

$$\underset{\theta}{\text{argmax}} \;\; p(\mathcal{D}|\theta) = \underset{\theta}{\text{argmax}} \prod_{i=1}^{n} p(y_i|\theta, x_i) \tag{2.8}$$

An equivalent estimation can be achieved using the negative log likelihood [32]:

$$\underset{\theta}{\text{argmin}} \sum_{i} -\log(p(y_i|\theta, x_i)) \tag{2.9}$$

The MLE solution, however, does not take into account prior information. Maximum a Posteriori (MAP) estimation, on the other hand, does. The updated loss function is formally written as:

$$\underset{\theta}{\text{argmin}} \sum_{y} -\log(p(y_i|\theta, x_i)) - \log(p(\theta)) \tag{2.10}$$

which results in the particular weight values that maximize the posterior. A more detailed discussion regarding these methods, along with detailed proofs, can be found in [32]. [31] provides summary guidance on the concepts.

2.4.2 Variational Inference

Although maximum likelihood estimation (MLE) and maximum a posteriori (MAP) estimation provide a means of arriving at a point estimate to approximate the posterior for a model, a more robust Bayesian estimate can be achieved via Variational Inference (VI). This approach approximates the intractable posterior distribution $p(w|\mathcal{D})$ with an optimization technique that identifies a distribution $q_\theta(w)$, which is an observable and tractable distribution in a family of probable posteriors Q, that most closely represents the actual posterior. θ refers to the learnable parameters of the variational distribution. Identifying $q_*(w)$ that most closely estimates $p(w|\mathcal{D})$ is achieved by minimizing the Kullback-Leibler (KL) divergence between the two, a procedure that estimates how different two probability distributions are

from one another. This estimation is achieved via a continuous summation (integration):

$$\mathbb{KL}[q_\theta(w)||p(w|\mathcal{D})] := \int_\Omega q_\theta(w) \log \frac{q_\theta(w)}{p(w|\mathcal{D})} d(w) \tag{2.11}$$

where Ω refers to all weights from the neural network from which w is obtained. Per [33], this can be rewritten as:

$$\mathbb{E}_{q_\theta(w)}[-\log p(y_i|\theta, x_i)] + \mathbb{KL}[q_\theta(w)||p(w|\mathcal{D})] \tag{2.12}$$

In variational inference, the desired variational distribution is that which minimizes the objective, where the first term (the expected value) estimates how closely the variational distribution predicts the data, and the second (the KL term) indicates how closely the variational distribution is to the prior.

2.5 Uncertainty

Traditional neural networks can suffer from overfitting—a condition where the model performs well on training data samples, but does not when presented new samples [33]. This leads to overconfidence in predictive performance, may provide false assurance to stakeholders, and can result in dangerous outcomes. Bayesian networks address this concern by introducing an uncertainty measurement for the weights within each neural network layer. As [33] explains, applying Bayesian techniques for weight estimation in neural networks is a very difficult (intractable) problem, since the number of parameters is large. Variational inference can make estimating a solution tractable.

Uncertainty in Bayesian deep learning (BDL) is observed from two perspectives - that uncertainty which can be reduced with more data (epistemic uncertainty) and that which cannot (aleatoric uncertainty) [34]. The irreducible aleatoric uncertainty is that which exists due to randomness in outcomes. The reducible, epistemic uncertainty, comes about as a result of estimation with limited data [35].

Total uncertainty, is quantified by taking the summation of aleatoric and epistemic uncertainty:

$$\frac{1}{T}\sum_{t=1}^{T} diag(\hat{p}_t) - \hat{p}_t{}^{\otimes 2} + \frac{1}{T}\sum_{t=1}^{T}(\hat{p}_t - \bar{p})^{\otimes 2} \tag{2.13}$$

2.5.1 Aleatoric Uncertainty

Randomness and noise within a dataset is difficult, and most times impossible, to control. Each time a sample is selected for incorporation into training a model, it will be accompanied by environmental elements which lead to uncertainty in the model. This type of uncertainty is considered aleatoric [36]. Rather than attempting to improve aleatoric uncertainty, or reduce it (which is not done in BDL training and model refinement), the standard approach is to estimate it so that such calculations can be factored into the overall uncertainty assessment of the model when inference is performed. Ideally, aleatoric uncertainty would be quantified so that its value may be used to assess and improve upon the quality of a predictor. For ML applications performing classification tasks, [35], referencing work by [37], suggest the following quantification of aleatoric uncertainty:

$$\frac{1}{T}\sum_{t=1}^{T} diag(\hat{p}_t) - \hat{p}_t{}^{\otimes 2} \tag{2.14}$$

a summation where T represents the number of Monte Carlo samples taken while calculating the difference between the diagonal matrix representation of the c-dimensional vector of probabilities $\hat{p}_t$ (c being the number of classes to predict) and the tensor product of the same c-dimensional vector of probabilities with its transpose ($\hat{p}_t{}^{\otimes 2} = \hat{p}_t\hat{p}_t{}^{\top}$) [35].

2.5.2 Epistemic Uncertainty

When a model is first trained there may not be a sufficient representation of data for the model to accurately label all examples that might exist along a classification spectrum (Figure 2.1). This implies that there is uncertainty as it pertains to those areas where little is known, since the training dataset did not include enough examples to inform the model. This uncertainty is known as epistemic uncertainty [33]. Like aleatoric uncertainty, epistemic uncertainty can be calculated using the following equation proposed by [37]:

$$\frac{1}{T}\sum_{t=1}^{T}(\hat{p}_t - \bar{p})^{\otimes 2} \tag{2.15}$$

where $\bar{p}$ is the mean of all Monte Carlo sample probabilities for each class and the tensor product evaluated as follows:

$$(\hat{p}_t - \bar{p})^{\otimes 2} = (\hat{p}_t - \bar{p})(\hat{p}_t - \bar{p})^{\top} \tag{2.16}$$

Unlike aleatoric uncertainty, epistemic uncertainty can be improved through the introduction of data to inform the model about those areas where it lacks knowledge. Together, epistemic and aleatoric uncertainty reveal total uncertainty that exists in a model. Figure 2.1 is a toy chart depicting the relationship between aleatoric, epistemic, and total uncertainty [38]. In this diagram, note that the epistemic uncertainty bands shrink where more examples (data) are available, indicating that total uncertainty is more affected by aleatoric uncertainty, which remains relatively consistent across all predictions.

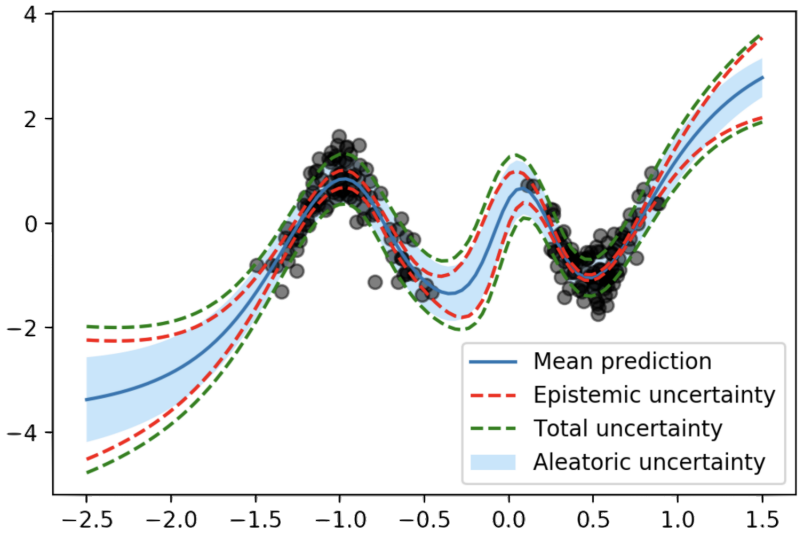


Figure 2.1. Uncertainty can be measured from the perspective of data sufficiency or model efficiency [38]. This toy depiction demonstrates how aleatoric uncertainty is greatest where fewer data examples exist. For continual/active learning, one aim is to discern the sufficiency of the data as compared to the reliability of the model.

2.6 Acquisition Functions

Epistemic uncertainty can be reduced through the acquisition of useful data. This is data that informs the model in classification areas where it lacks confidence. Deciding which new data to incorporate requires a systematic and quantitative process. A discussion on the active learning approach used to identify where new samples are sourced is provided in [24], Settles [39] along with Houlsby et al. [40] explain how this is achieved through the use of mathematical calculations (or a function) that quantifies the benefit afforded by the incorporation of a particular sample into training—an acquisition function [41]. The three proposed calculations are 1) max entropy assessment, 2) Bayesian active learning by disagreement (BALD), and 3) random sampling (making use of a uniform distribution to randomly select from a pool of samples).

CHAPTER 3:
Methods

Bearing in mind the challenges of ensuring accountability in design, balancing performance objectives with principles for responsible AI, this thesis proposes a set of repeatable practices (a framework) that can be implemented during the ML/DL system development phase that might serve as a means of engineering safety, accountability, and, overall, assurance into MDLC. This will be accomplished by examining whether actions taken during MDLC are attributed to undesired outcomes that are observed when an AI system is operational.

Although several variations of RAI principles have been proposed in various domains, this analysis adapts DOD RAI fundamentals as the basis for developing the assurance framework, since the SOI is a DOD AI system [22]. The results of this analysis are transferable, however – whether to DOD systems, other Government systems, or commercial systems. Furthermore, this analysis can guide development of safety-critical systems (those that might cause environmental harm, severe damage to property, or serious injury or death to humans when they malfunction), weapons systems (those used to defend property and life from adversarial entities), or other AI systems that do not operate in safety-critical environments.

3.1 Research Objectives

This thesis considers how such a systematic approach might impact the creative process in the design phase. Frameworks and decision support systems often restrict the natural flow of experimentation, taking control away from those making creative decisions during development. By contrast, the framework proposed here presents recommendations in a manner that synergizes creativity, performance, and explainability cohesively [10].

3.2 Systems-Theoretic Process Analysis (STPA) [1]

A key assertion in this research is the proposition that system theory provides techniques or methodologies that can be adapted to address RAI concerns. One such technique proposed by Leveson [42] is System Theoretic Process Analysis (STPA) which introduces procedures that guide hazard analysis. Adapting this framework to address the concern for engineering responsible AI provides a means of producing a repeatable process for achieving governance, reliability, and traceability in AI systems.

A strength of system safety engineering frameworks is that they connect abstract safety policies, which are difficult to make actionable through technical means alone, to implementable requirements. Tools from this domain further have the advantage of being regularly applied in high consequence domains, well studied, and providing a strong basis on which to systematize efforts to identify social and ethical risks [43] [44]. Such tools include traditional safety-through-reliability techniques like fault-tree analysis (FTA) [45] and Failure Mode and Effects Analysis (FMEA), quantification-oriented approaches used for decades to reduce the number of failures in systems under analysis [46]. By contrast, Leveson's Systems Theoretic Accident Model and Process (STAMP) [47] explicitly rejects the notion that reducing failures improves safety, noting that safety is a property of systems not components. STAMP instead models hazardous states that could lead to defined losses as insufficient control within an entire sociotechnical system.

3.2.1 Prior MDLC STPA Research

Qi et al. [48] acknowledge the usefulness of STPA for learning enabled systems. In this analysis, they identify a number of examples where STPA was used to evaluate the ML development lifecycle. It is acknowledged in their study that, although several attempts have been made to adapt STPA to machine learning contexts, it is still not yet determined whether there exists an approach that can be generalized across all ML systems. In fact, in reviewing the 20+ papers that make the attempt to apply systems safety methods to identify causal paths that lead to undesired or unsafe system states for learning systems, many take slightly and, sometimes, significantly different approaches. In their paper, the propose a new form of process analysis which they label "DeepSTPA"—an adaptation which 1) focuses upon

[1] The summary introducing this section, 3.2, is captured from the NeurIPS Workshop paper [14], evaluating the use of STPA for ML-based systems

identifying hazards associated with data-driven activities occurring at all stages of MDLC and 2) identifies the importance of evaluating deep learning neural network layer hazards in order to discover parameter (weight) permutations that contribute to failures. In the end, however, much remains unexplored. Although they identify the need for a standardized approach and highlight intriguing studies, the work is ultimately a survey of the potential that exists.

3.2.2 STPA Procedure

STPA ordinarily proceeds in four steps. This thesis proposes a modified six step process for two reasons. Firstly, the standard process combines defining the system and identifying losses and hazards into a single step. Doing so risks losing insights needed during root cause analysis, since consolidating these elements may reduce their singular importance. For large, complex systems, care must be taken in clearly defining what components are of primary concern and why. Second, classic STPA concludes by identifying causal scenarios without proposing changes to the system that would prevent these scenarios or the losses that arise when they are encountered. The updated six step approach is depicted in Figure 3.1 and is described as follows:

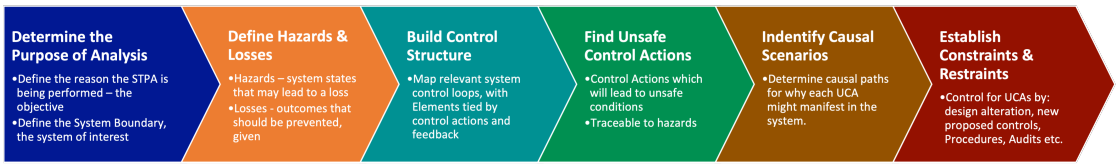


Figure 3.1. The proposed 6-step STPA process. Adapted from [42].

1. Determine the purpose of analysis, this includes an explanation of why the STPA is being performed, what the system of interest is and what system boundary will be placed upon the analysis. Due to the inherent risk of scope expansion as the analyst examines component interactions, establishing the analysis boundary is key to ensuring the most useful safety and assurance recommendations are established as an outcome of the activity.

2. Define hazards and losses. Hazards are unsafe system states that, when coupled with worst case scenarios, will lead to an unacceptable loss. Losses are those outcomes that must be prevented and are unacceptable consequences of a component's failure. These are system-wide aspects, not particular to the component or SoI, but to which the component may contribute.
3. Build a control structure, a mapping of how integrated components affect others, the system control loops, as well as how controller inputs and component feedback impact the overall state of the system.
4. Find unsafe control actions (UCAs). A control action (CA) is a procedure, performed by a controller (human or otherwise) that is necessary for each component within the control structure to perform its task or place the system in a particular state. The unsafe control actions are those CAs directly traceable to hazards, ultimately leading to unsafe conditions.
5. Identify causal scenarios (CS). This step involves examining each UCA to discover the paths leading to their manifestation in the system. These are commonly referred to as causal paths. By recognizing them one can identify actions that can either be avoided or prevented or feedback that can be acted upon so that unsafe system conditions might be avoided, which is the ambition in the final step;
6. Establish constraints and restraints to guide system redesign efforts. The STPA is incomplete if, in the end, the outcomes are not sufficient to inform system operators and designers of what changes in behaviors, function, or operating procedures should be made to prevent undesired system outcomes.

3.3 The System of Interest

This research assesses a notional design for a passive sonar maritime system used to predict the presence of certain vessels within a defined radius of a sensor (or group of sensors). Applying STPA in this manner involves identifying the context in which such a system might be used, enumerating the controls, losses, hazards, and modeling a control structure for those hazards, and examining the components within the system to determine how their design might be enhanced to reduce the likelihood of failure and/or help identify the root cause of such failure if/when it occurs.

Performing an STPA prior to completion of development and while requirements might still be under review, should lead to recommendations that will guide the efforts of engineering traceability into data processing activities, active, federated, and continual learning procedures (to reduce the potential for model drift), and the design and application of Bayesian methods used to build the neural networks that govern the predictive processes. Because modern software solutions are designed with Agile, fail-fast philosophies, one can leverage such practices to identify deficiencies earlier in the formation phase of the system. The intuition being that, such analysis would not hinder or slow the scientific design phase, but inform it so that these systems would be more resilient, reliable, sustainable, and, overall, effective when deployed.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Results

This thesis examines how STPA can be used to provide the MDLC practitioner a framework for minimizing unacceptable outcomes and prevent the hazards that lead to them. It also identifies recommendations for process improvement and criteria for auditing AI systems to achieve more effective detection and mitigation of unintended consequences while they are under development. This chapter contains the artifacts generated during STPA, using the steps enumerated in Chapter 3, while analyzing the previously referenced passive sonar system as the case study. What follows are the results arising from the six steps.

4.1 STPA Step One: Determine Purpose of the Analysis

The purpose of this analysis is to investigate the machine learning development lifecycle (MDLC) used to produce a passive sonar acoustic sensing system that utilizes a neural network model to classify seafaring vessels. In this STPA, MDLC is considered a controlled process that must operate in a manner that comports with DOD RAI principles; particularly, the considerations pertaining to the MDLC practitioner's adherence to engineering responsible AI and incorporating an ability to trace unacceptable losses, hazardous system states, and unsafe control actions to activities performed during MDLC. In the final stage of the STPA, tradeoffs, recommendations, and auditable criteria are considered for incorporation into ML development so that the system is governed effectively.

In addition to articulating the purpose of the analysis, a description of the system under evaluation is necessary. The larger system architecture consists of four components:

1. an array of passive sonar vector sensors designed to respond both to scalar acoustic pressure as well as the vector motion of the medium in which they are deployed [23]. These sensors are deployed along with onboard audio filtering components that reduce the feature set for each audio sample to avoid overfitting, and a version of the convolutional neural network (CNN) model trained using Bayesian Deep Learning (BDL) techniques that assign class probabilities to each example—these classes (A-E) consist of A) Fishing Vessel, Tug, or Towing Vessel, B) Pleasure Craft, Sailboat,

Pilot, C) Passenger Ship, Cruise Ship, D) Tanker, Container Ship, Military Ship, Bulk Carrier, or E) No Ship—Background Noise

2. an inference analysis component where input from each of the devices is gathered and aligned with sonar observations either obtained as part of an Automatic Identification System (AIS) data feed or other through other identification sources in use by analysts—AIS is system that uses transceivers located on seafaring vessels that provides information regarding registered vessels within a pre-defined range of a particular sensor
3. A federated learning engine which is used to aggregate model parameters and deploy the improved version of the shared model to the devices dispersed in the target area
4. An active learning engine that facilitates audio sample labeling by human operators and shared model update when epistemic uncertainty improvements are incorporated.

Figure 4.1 depicts the operational view of the complete system:

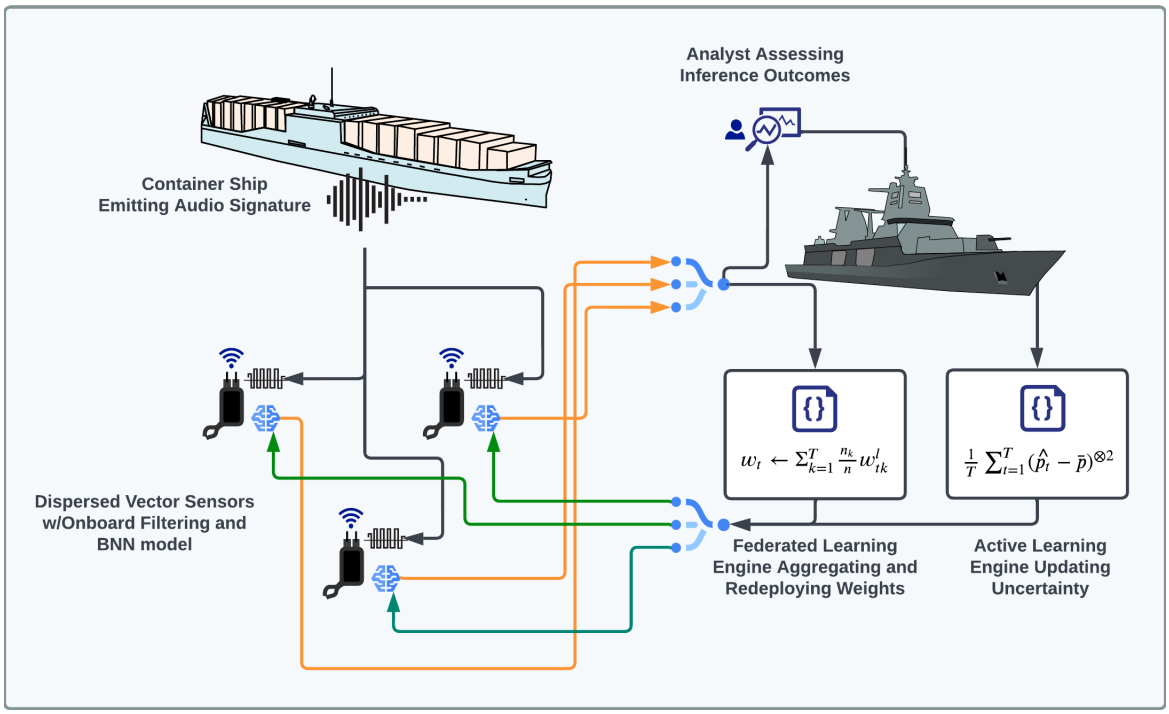


Figure 4.1. Diagram depicting the activities performed within the System of Interest. In this proposed version, vector sensors are deployed along with signal processing capabilities and the neural network model. The local model deployed with each sensor is updated as federated and active learning procedures are performed.

4.2 STPA Step Two: Define Losses and Hazards

In system theory, losses are those accidents that system designers should aim to avoid by implementing measures to reduce the likelihood the system enters a hazardous state, from which losses can occur [42]. Table 4.1 enumerates the losses identified for this SOI:

Table 4.1. System-Level Losses

Losses		
L1	Loss of Accountability	Refers to the inability to assign responsibility to an individual, component, or process while pursuing an explanation for a given outcome. Also includes an inability to prevent the system from ceasing undesired actions
L2	Loss of Property	Pertains to the critical equipment (the defense systems or other resources) that are unacceptable to lose, or costs associated with those losses that extend beyond desired expenditures when the failure occurs
L3	Loss of Life	Refers to casualties that arise from preventable hazardous system states

Hazards are system states that, when coupled with the worst case conditions (causal factors), can lead to unacceptable losses [42]. When referring to the state of the system, it is necessary to recognize the distinction between the state of overall system and that of the individual components belonging to the system. The hazardous states enumerated in step two are those that pertain to the system as a whole rather than to its individual components. With hazards, the objective is to describe system-level conditions that must be prevented or avoided through the use of constraints placed upon procedures performed by persons or entities exercising control over the system (i.e., a controller). Table 4.2 lists the hazardous states for the system aligned with the losses that could arise as a result:

Table 4.2. Hazardous System States

Hazards				
		L1	L2	L3
H1	Misidentification of vessel—friendly when hostile	X		X
H2	Misidentification of vessel—hostile when friendly	X	X	
H3	Failure to identify vessel (Not Seen When Present)	X		X
H4	Classification & Identification capability degrades over time (Catastrophic Forgetting)	X		
H5	(Unrecognized/Hidden) bias in predicting a particular class of vessel, whether too often or not often enough—overclassification of a particular vessel—overfitting and non-generalized model	X	X	
H6	Compromised System State: Either System enters an unexpected state and cannot recover (the unknown unknown) or bad actors taint inputs or outputs to yield unreliable results (classification outcomes)	X	X	X

Although there is no requirement within STPA to formally identify a relationship between a set of system principles or defined standards and the artifacts produced during the analysis, this thesis purposes to demonstrate how the losses and hazards identified in STPA step two align with the concerns of the DOD RAI principles [9]. Table 4.3 highlights the relationship between the RAI principles and losses and hazards that have been identified for the SOI.

Table 4.3. Hazards and Losses Aligned to DOD Responsible AI Principles

DOD RAI Principles		
	Losses	Hazards
Responsible	L2, L3	H6
Equitable	L1	H5
Traceable	L1, L2	H1, H2, H3, H4, H5
Reliable	L1	H1, H2, H3, H4, H6
Governable	L1, L2, L3	H1, H2, H3, H6

The *responsible* RAI principle defines the need to exercise proper judgment during development, deployment, and use so that a loss of property and life can be avoided. Failures in judgment can lead to misapplication (or non-application) of the measures necessary to prevent a system from being compromised—a hazardous system state arising from the controller exercising poor judgment while designing the system or utilizing it. The SOI is compromised when tainted data is used during training, adversarial examples (audio signatures purpose-built to deceive the neural network) are allowed to deceive the system, or bad actors are able to intercept operations and manipulate outcomes.

When appropriate measures are not taken to reduce unintended bias, the system will not operate in an *equitable* manner. Although this principle generally applies to fairness for persons that may be affected by the outcomes of the system, for the SOI, bias can refer to scenarios where one vessel class is predicted too frequently or not enough. Consequently, a loss of accountability emerges since, when class prediction bias occurs, it may not be immediately apparent which system component is responsible for producing this bias. This compromises traceability. Accountability requires that outcomes be explainable by those producing the outcomes. The SOI will not serve its stakeholders well if it does not equitably predict the vessels upon which it was trained. Measures should be taken during model

refinement to ensure that such inequity does not arise.

The *traceability* principle requires that designers incorporate a means of auditing the methods, data sources, design procedures, and documentation used during development and when deploying their systems. Those responsible for the SOI must document why particular datasets are chosen, the contribution the data makes to the overall performance of the model, why a particular Bayesian neural network method of parameter estimation is selected, the limitations introduced when choosing one method over another. They must also be able to explain the weight aggregation methodology used during federated learning, as well as the data acquisition and labeling approach used when active learning techniques are implemented. They must document how uncertainty is quantified and how such quantification enhances or degrades system performance. Outcomes must be reproducible either by the same team or an independent body, with the same artifacts utilized by the designers. Inferential reproducibility [11] is a useful approach to achieve assurances that outcomes are traceable.

A system that cannot be controlled or for which hazardous behavior cannot be detected is a system with a potential for producing all losses that have been identified for that system. The controller must possess an ability to *govern* the SOI's behavior. This requires that designers understand the hazardous system states and unsafe control actions that lead to these states. Without control exercised during development or operation, the system will fail to identify the vessels on which it has been trained. Failed identification will yield poor classification outcomes and an unusable system.

Accountability is compromised when measures are not taken to provide a means of auditing the system in this manner. Resources may be lost if the auditing mechanisms are inefficient due to failed controls—more effort may be required to establish an Authority to Operate (ATO). More computing central processing unit (CPU) and memory resources may be necessary when efficiencies are not incorporated early; a state that could develop when proper care is not taken or adequate consideration applied during the design and development phase. Each of the misidentification and classification bias hazards are at risk since decisions made during data selection and preparation and model design cannot be justified or explained. A root cause of catastrophic forgetting (a condition where a model which previously performed well during inference on a particular sample, degrades in predictive

capability after new data is introduced during active learning) may not be discernible when measures are not taken to incorporate a means of auditing system activities that pertain to active learning.

The SOI must be resilient, able withstand the rigors of independent verification and validation (IVV), whether these IVV activities assess performance, security, safety, or other business requirements. This resilience would assure stakeholders that the system is *reliable*. Failed resilience under validation leads to a loss of accountability, as errors found in testing will cause developers to reevaluate and modify errant decisions that are deemed acceptable during system refinement, prior to deployment. Failures in validation would also demonstrate that the system will not be successful when deployed, leading to hazards associated with misidentification, misclassification, and potential system compromise due to vulnerabilities discovered during validation.

4.3 STPA Step Three: Build Control Structure

Control actions are used to manage conditions so that the system does not enter a potentially hazardous system state. During this step of the STPA, the control structure is modeled in an effort to emphasize the relationship between the controlled process of concern (i.e., the MDLC) and the larger system in which that controlled process exists. As shown in the diagrams that follow, both the overall system and the controlled processes have control actions (commands sent by the controller) and feedback (outcomes arising from those commands) that can be assessed empirically or subjectively.

Figure 4.2 depicts the overarching system. In this diagram, the controllers are made up of the hardware used to gather and process sonar data, the algorithms used to make predictions, the operators and analysts that interact with the deployed equipment, and the developers that design the predictive components constituting the core of the SOI's capability. Each of these controllers perform actions and receive feedback. The actions are applied to the controlled processes summarized in the diagram: detecting, recognizing, identifying, tracking and/or targeting vessels, and the machine learning development lifecycle (i.e., the MDLC) used to generate the predictive ML model.

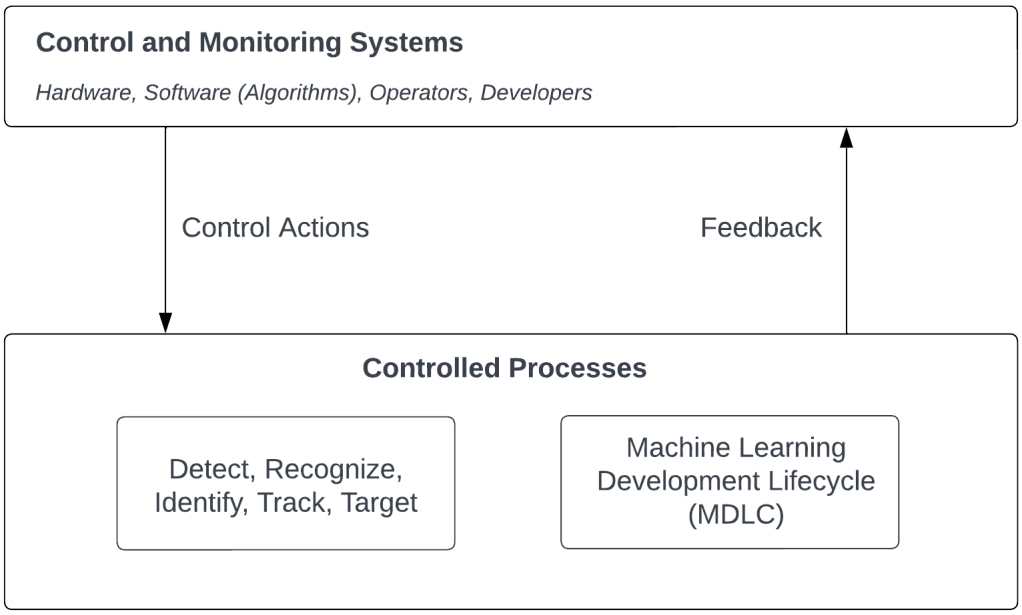


Figure 4.2. Sonar Detection System Control Structure

A challenge in STPA is that of identifying boundaries upon which to scope the analysis. Because most systems consist of numerous components and subcomponents, it can be difficult to avoid secondary and tertiary considerations when the primary area of concern touches upon activities performed within adjacent controlled processes. In this STPA, the losses, hazards, unsafe control actions, and loss scenario analysis is confined to the MDLC controlled process. Figure 4.3 provides a more detailed overview of how MDLC is viewed within the context of STPA; as a set of controlled processes with particular control actions and feedbacks.

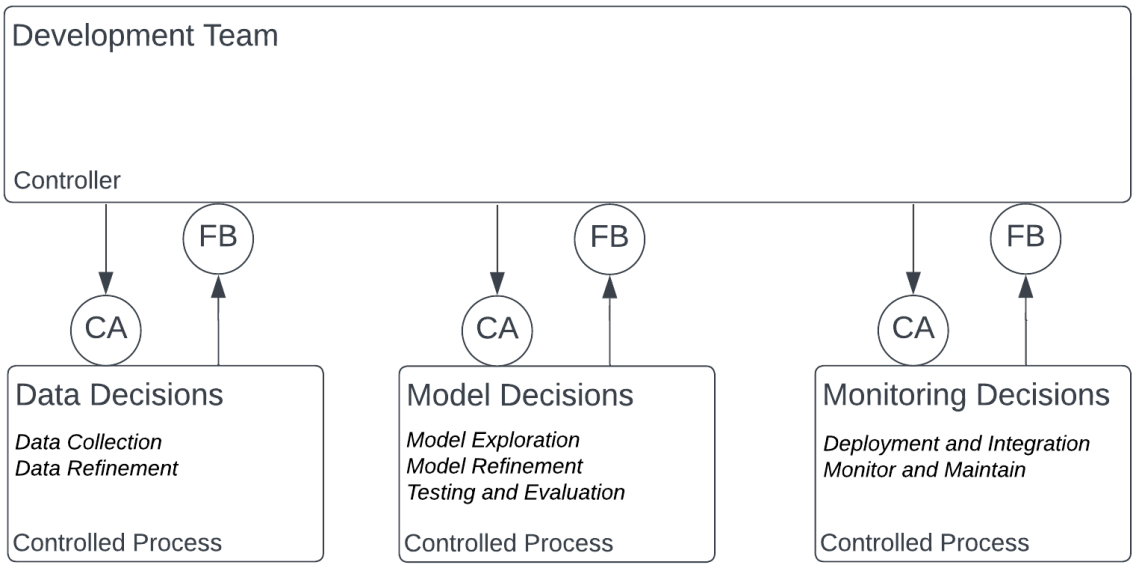


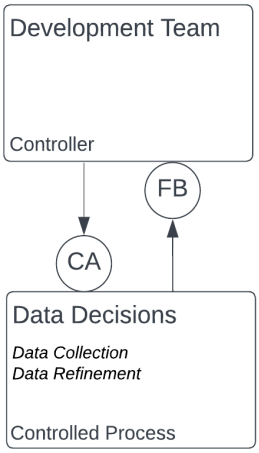
Figure 4.3. Machine Learning Development Lifecycle Process

Within each MDLC controlled process, the development team acts as the controller - this team consisting of the business analysts, data scientists, machine learning engineers, sonar experts, and validation personnel that work together to produce an operational solution. Data decisions are those that involve gathering, refining, and labeling the audio data needed to inform the system during training. Model decisions are the neural network design activities undertaken to produce reliable outcomes during testing and inference. Monitoring decisions are actions taken after the model is deployed to identify and roadmap areas of improvement by means of federated and active learning. The control actions and feedbacks for each of these controlled processes is further elaborated within Tables 4.4, 4.6, and 4.8.

4.3.1 MDLC Data Decisions

The data decisions controlled process represents the measures taken by the development team (i.e., the controller) to identify data appropriate and of a quantity necessary to train the system. For the SOI, this data must provide the needed detail to build a neural network capable of classifying audio signatures within the number of designated ship classes required. When the appropriate data is identified, additional actions are taken to refine the data, removing characteristics that do not contribute to or make it difficult to achieve the desired performance in the ML model. Table 4.4 enumerates the control actions (CA) and feedback (FB) that occur when the controller is implementing this process followed by Table 4.5 which provides brief explanations for each CA and FB: [2]

Table 4.4. Data Decisions Controlled Process

Control Actions		Feedback
1. Dataset Selection/Omission 2. Audio Sample Period 3. Frequency Filtering 4. Ship Class Label Determination 5. Test, Train, Validation Split 6. Data Augmentation		1. Data Sufficiency 2. Class Representation (Under/Over) 3. Data Credibility 4. Noise Presence/Absence

[2]For each of the control actions and feedback tables that follow, note that control actions and feedback do not map 1:1, they are not directly associated with one another

Table 4.5. Data Decisions Controlled Process Control Action and Feedback Descriptions.

Control Actions		**Feedback**	
CA-1	Dataset Selection/Omission *Determining which source will provide appropriate audio samples for training*	FB-1	Data Sufficiency *Identified whether there is enough data to adequately train the neural network*
CA-2	Audio Sample Period *Deciding the length of time and short-time Fourier transform (STFT) boundaries for segmentation of audio samples used for training*	FB-2	Class Representation *Observed whether each class is adequately represented in each test, train, validation set*
CA-3	Frequency Filtering *Applying filtering mechanisms to reduce the number of dimensions (i.e., features) used during training*	FB-3	Data Credibility *Discovered whether the source data can be relied upon for training*
CA-4	Ship Class Label Determination *Selecting how many and which vessels should be included in classes for which predictions are made*	FB-4	Noise Presence/Absence *Observed the level of noise present in the samples obtained*
CA-5	Test/Train/Validation Split *Determining the proportion of and which samples are included in the test, train, and validation sets*		
CA-6	Data Augmentation *Efforts taken to increase the training dataset by artificially producing new data through modifying existing data*		

4.3.2 MDLC Model Decisions

The judgment that guides ML designers through the process of engineering a reliable model, capable of consistently predicting the class of vessel associated with an audio signature while also adhering to the RAI principles that must inform each decision, constitute the model decisions controlled process. In this stage, feature engineering occurs, whereby determinations are made as to what characteristics of the data are most pertinent for inclusion into the model. Additionally, neural network exploration occurs, wherein Bayesian parameter estimation methods are examined to yield the most suitable model. Model refinement also takes place in this process. This includes tuning of hyperparameters such as regularization, weight decay, learning rate, and batch size in an effort to reduce overfitting and improve generalizability of the network. Following refinement and tuning, to assist in making decisions about the viability of the model, performance testing occurs, providing developers valuable insights needed to pursue additional refinement. Table 4.6 provides an overview of the control actions performed and feedback encountered while undertaking this process. Table 4.7 describes the CA and FB:

Table 4.6. Model Decisions Controlled Process

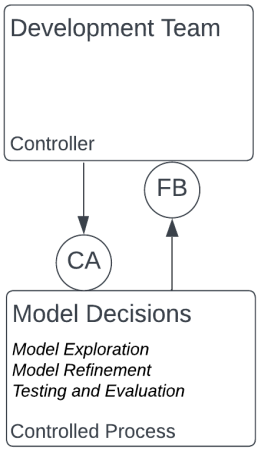
Control Actions		Feedback
1. Hyperparameter Tuning (HT) 2. Neural Network Weight Approximation Method 3. Network Architecture 4. Uncertainty Incorporation 5. Acquisition Function Determination 6. Federated Learning Approach 7. Active Learning Procedure 8. Multiclass vs. Multi-Label Decision 9. Feature Engineering		1. Hyperparameter Tuning Observations 2. Classification Outcomes (Inference) 3. Performance Measurements 4. Uncertainty Measurements

Table 4.7. Model Decisions Controlled Process Control Action and Feedback Descriptions

Control Actions		**Feedback**	
CA-1	Hyperparameter Tuning (HT) *Adjustments made to hyperparameters to improve generalizability*	FB-1	HT Observations *Observed changes to neural network when HT values are modified*
CA-2	Parameter Approximation Method *Deciding whether Variational Inference or Markov Chain Monte Carlo yields the most suitable neural network weights*	FB-2	Classification Outcomes *Observed class predictions based upon weights existing in network*
CA-3	Network Architecture *Decisions pertaining to network configuration such as dropout layers, type of network used - convolutional or recurrent, number of hidden layers, etc.*	FB-3	Performance Measurements *Calculated class prediction accuracy metrics such as precision, recall, and F1-score*
CA-4	Uncertainty Incorporation *Model refinement in BDL includes procedures for incorporating observations made when calculating uncertainty based upon probability distribution for estimated NN parameters/weights*	FB-4	Uncertainty Measurements *Calculated uncertainty for weights selected during model refinement*
CA-5	Acquisition Function Determination *Selecting which method to use (i.e., max entropy, BALD, random sampling) when deciding whether a particular sample will improve epistemic uncertainty*		

Table 4.7 – continued from previous page

Control Actions		Feedback	
CA-6	Federated Learning Approach *Applying a method to aggregate weights for the distributed models deployed on the distributed sonar detection nodes*		
CA-7	Active Learning Approach *Selecting which method to use during active learning to identify new data for incorporation into model refinement (i.e., membership query synthesis, pool-based sampling, stream-based selective sampling, or uncertainty sampling*		
CA-8	Multiclass vs. Multilabel Decision *During model exploration, determining which classification method optimizes performance—multiclass: assigning one label from multiple classes, multilabel: assigning zero or more labels from multiple classes*		
CA-9	Feature Engineering *Decisions are made to establish what aspects of the examples, provided as data, are important to incorporate into the model in order to produce reliable predictions*		

4.3.3 MDLC Monitoring Decisions

Assessing model performance through the controlled process of presenting data and measuring class prediction outcomes (inference) is the phase of MDLC where the design team gathers valuable input to achieve improved performance in subsequent MDLC cycles. In this stage the model is deployed to the SOI nodes where sound signatures are gathered, processed, and assessed. Activities such as predicting vessel classification and ingesting changes to model parameters based upon federated and active learning activities are undertaken. In addition, monitoring and maintenance activities occur in this step. For example, system managers determine whether nodes should be removed, relocated, or the entire system decommissioned as a result of outcomes discovered once deployed to an operational environment. Table 4.8 provides an overview of the control actions performed and feedback encountered while monitoring decisions are made. Table 4.7 describes the CA and FB:

Table 4.8. Monitoring Decisions Controlled Process

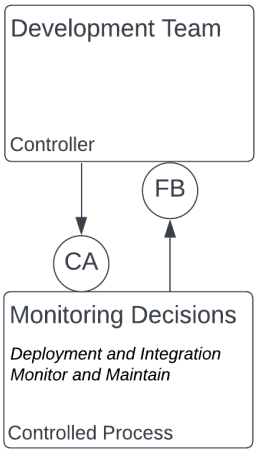
Control Actions		Feedback
1. Sensor/Node Placement 2. Monitoring Interference and Noise 3. Analyzing Model Drift 4. Processing Distributed Weights		1. Node Improvement or Degradation 2. Model Improvement or Degradation 3. Class Expansion 4. System Failure

Table 4.9. Monitoring Decisions Controlled Process Control Action and Feedback Descriptions

Control Actions		Feedback	
CA-1	Sensor/Node Placement *Deciding where best to place nodes on sea floor to achieve best performance for SOI federated node network*	FB-1	Node Improvement/Degradation *Observed sensor performance based upon placement of node*
CA-2	Monitoring Interference and Noise *Monitoring the level of noise contributed by objects in the environment which are not of concern for SOI operation*	FB-2	Model Improvement/Degradation *The performance change observed, based upon active and federated learning activities*
CA-3	Analyzing Model Drift *Actions taken to detect whether oracle labeling provided during active learning has caused previous model performance to degrade*	FB-3	Class Expansion *Identified vessels which are not included in existing class definitions*
CA-4	Processing Distributed Weights *Collecting weights from nodes distributed in the environment so that federated learning procedures may be performed*	FB-4	System Failure *An observed state whereby the system no longer functions and must be decommissioned*

4.4 STPA Step Four: Find Unsafe Control Actions

When documenting the artifacts of STPA in classic scenarios, control actions are purposeful commands issued to the system by a controller as part of a controlled process. In this study the controlled process is the ML development lifecycle, where "commands" issued in this context are the control

actions enumerated in step three above. In step four, control actions are inspected to determine whether performing them, not performing them, performing them in the wrong order, or ceasing the action too early could result in a hazardous system state. Performing this step allows designers, operators, system administrators, and other stakeholders to ascertain unsafe system states based upon when and whether control actions are exercised (i.e., unsafe control actions (UCAs)). This is an important step in determining where system constraints and restraints must be applied.

Traditionally the UCA analysis requires a "wrong order" assessment which includes a "too early" or "too late" determination and is primarily useful for systems in motion; knowing MDLC is not in motion, per se, this concern is simplified whereby "wrong order" is not further elaborated (e.g., this STPA does not inspect scenarios when selecting a neural network weight approximation method occurs "too late" or "too early," but only instances where this might take place before or after the "Model Decisions" phase in the lifecycle). To further elaborate, when the Data Decisions controlled process is evaluated and the control actions are investigated for safety, wrong order would refer to whether the action is performed either prematurely or delinquently, such as in an MDLC stage that is not appropriate for the action 1.1. In a similar sense, the traditional review of assessing whether stopping an action too soon or applying it too long would cause a hazard is simplified to "Not Completing." This is to say that a hazardous state can emerge when an action is not completed prior to proceeding to the next stage of the development cycle.

The following UCA tables describe the potentially hazardous system states along with conditions that arise when the control actions, identified for each MDLC controlled process, are engaged. The hazards in brackets (e.g., [H1, H2]) represent the hazardous states enumerated in step two. Tables 4.10, 4.11, and 4.12 lists the UCAs for the MDLC controlled processes:

Table 4.10. Unsafe Control Action Analysis—Data Decisions Process

Data Decisions (DD)					
CA#	Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Order Causes Hazard	Not Completing Causes Hazard
1	Dataset Selection/Omission	Not providing dataset selectivity criterion may cause system to false classify due to lack of data quality [H1, H2]	Data required for effective training may be excluded [H1, H2]	Not exercising discretion in dataset selection prior to training leads to poor model performance [H1, H2, H3, H5]	Not completing the task of discriminating datasets leads to poor model performance [H1, H2, H3, H5]
2	Audio Sample Period	Not defining a period may lead to excessive noise [H1]	Defining the period that is too long may yield unwanted audio signatures [H1]	Performing this after or before DD leads to wasted time retraining [H3]	Not being thorough in analyzing appropriate periods may lead to unexplainable results [H3, H5]
3	Frequency Filtering & Dimension (Feature) Reduction	Not providing may lead to overfitting, bias, mislabeling [H1, H2, H3, H5]	Providing may lead to wrong features being considered or prioritized [H1, H2, H3, H5]	Outcome same as CA2 [H3]	Outcome same as CA2 [H3, H5]
4	Ship Class Label Determination	Not labeling causes unknown classification [H1, H2, H5]		Same as CA4 “Not Providing” [H1, H2, H5]	Same as previous [H1, H2, H5]
5	Test/Train/Validation Split	Not performing makes training (next MDLC step) impossible [H6]		Same as CA5 “Not Providing” [H6]	Partial completion leads to unreliable model [H1, H2]

Table 4.11. Unsafe Control Action Analysis—Model Decisions Process

Model Decisions (MD)					
CA#	Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Order Causes Hazard	Not Completing Causes Hazard
1	Model and Input Feature Hyperparameter Tuning	Not providing leads to poor model performance [H1, H2]		Tuning some hyper parameters before others may lead to unreliable outcomes [H1, H2]	Incomplete tuning leads to same [H1, H2]
2	NN Weight Approximation Method (VI or Markov Chain Monte Carlo)	Not selecting a method causes poor network weights [H1, H2, H3, H5]		Selecting after MD leads to poor model performance [H1, H2, H3, H5]	
3	VI Distribution Selection (Mean-Field Approximation, MC Dropout with Bernoulli Prior)	Not calculating KL for VI causes poor or unreliable model performance [H1, H2, H3, H5]		Selecting after MD leads to poor model performance [H1, H2, H3, H5]	
4	Network Architecture	Number of layers, Number of Neurons, Dropout, CNN, Fully Connected [H1, H2, H3, H5]		Selecting after MD leads to poor model performance [H1, H2, H3, H5]	
5	Measuring and Incorporating Uncertainty	Not incorporating uncertainty leads to poor model performance [H1, H2, H3, H5]			
6	Acquisition Function Decision	Not deciding upon an acquisition function causes unreliable active learning [H6]		Same as CA5 “Not Providing” [H6]	
7	Federated Learning, Weight Aggregation Approach	Not implementing a federated learning approach causes distributed models to poorly perform [H1, H2, H3, H6]		Same as CA6 “Not Providing” [H1, H2, H3, H6]	

Table 4.11 – continued from previous page

Model Decisions (MD)					
CA#	Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Order Causes Hazard	Not Completing Causes Hazard
8	Active Learning Procedure (Use of Bayesian Deep Learning by Disagreement - BALD)	Not implementing active learning causes system to not recognize unknown vessels [H1, H2, H3]	Implementing active learning may lead to catastrophic forgetting [H1 - H5]		
9	Opting for Multiclass or Multi-label Classification	Not deciding impacts classification performance and identification [H1, H2]			

Table 4.12. Unsafe Control Action Analysis—Monitoring Decisions

Monitoring Decisions (MnD)					
CA#	Control Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Order Causes Hazard	Not Completing Causes Hazard
1	Sensor/Node Placement	No strategy for placement can lead to poor performance or system failure [H1, H2, H3, H6]			Not placing enough nodes can cause non-identification [H3, H6]
2	Performing interference and noise measurements	Not measuring interference and noise leads to poor data quality [H1, H2, H3, H6]			
3	Assessing model drift (Catastrophic Forgetting) during Active Learning	Not analyzing model drift contributes to catastrophic forgetting [H1, H2, H3, H4, H6]		Same as CA3 "Not Providing" [H1, H2, H3, H4, H6]	
4	Gathering distributed weights	Not gathering weights from distributed nodes degrades federated learning activities [H1, H2, H3, H4, H6]			Partial aggregation leads to unreliable system state [H1, H2, H3, H6]

4.5 STPA Step Five: Perform Loss Scenario Analysis

Under what conditions would an unsafe control action occur? Do they arise when a procedure is not adequately documented, understood, or governed by reactive components engineered to prevent the UCA? What would cause a controller to execute a control action improperly or not execute it at all? STPA step five purposes to evaluate these questions by identifying scenarios that either lead to UCAs or contribute to a controller's negligence in performing an expected action within the controlled process. In some literature, these scenarios are referred to as "causal" scenarios [49], while the originators [50] refer to them as "loss" scenarios, each comprised of causal factors. Both descriptions aim to summarize what must be the outcome of this step in STPA: to highlight circumstances that lead to unsafe control actions and hazardous system states in an effort to better perceive why losses emerge.

For the SOI, losses are established in step two and UCAs are assessed in step four. In step five, loss scenarios are presented in an effort to identify aberrant procedures that hinder successful system operation. Table 4.1 provides a set of loss scenarios (LS) categorized by the circumstance from which it arises (resulting from an unsafe control action or non-compliant execution of a control action) and aligned to the loss conceived as a result of the scenario occurring:

Table 4.13. Loss Scenarios

LS	Scenario	Cause	Loss
LS1	Distributed models degrade during federated learning due to poor calculations of Kullback-Leibler (KL) divergence, while applying aggregated weights	Non-Compliant CA Execution	L1, L2
LS2	Model performs poorly on new examples because epistemic uncertainty measurements are not incorporated during model refinement	UCA	L1, L2, L3

The loss scenarios provide potential failures within the MDLC controlled process that lead to a loss. In the first loss scenario (LS1), the controller's failure to properly calculate the new weights during federated learning procedures will cause the subject nodes to either degrade in their ability to predict proper labels for detected vessels, or not be capable of detecting new vessels belonging to a particular class. This leads to a loss of accountability (L1) since the controller responsible for performing the aggregation may no longer be trusted to do so. It can also lead to a loss of property (L2) since resources must now be commissioned to recalculate, redeploy, and, potentially, recover failed nodes.

The second loss scenario (LS2) arises directly from a UCA; CA-4 within the Model Decisions controlled process (see Table 4.7). This occurs due to a failure on the part of the controller to either integrate new audio data during active learning procedures, or the controller not performing epistemic uncertainty calculations when concluding upon model weights as part of the Bayesian deep learning (BDL) training activities. In either circumstance, each of the three losses (L1, L2,

and/or L3) are at risk of occurring: a loss of accountability arises due to a lack of understanding regarding why the measurements are not incorporated or a lack of trust in the controller's ability to perform expected training procedures; a loss of property, as more human effort (increased effort by controllers in other controlled processes within the system) is required to operate without the class predictions that are anticipated; or a loss of life as those relying upon the class predictions may not be aware of or able to identify unfriendly vessels, thereby risking the lives of those utilizing the SOI's capability.

Although additional loss scenarios may be considered, the aforementioned are proposed as examples of an approach that can be taken as part of a more in-depth analysis for the MDLC system of interest.

4.6 STPA Step Six: Propose Constraints and Restraints

The final step in STPA is to identify parameters that should be applied to the system in an effort to reduce the likelihood of a hazardous system state. These parameters are categorized as either *constraints*, the things that must be done, or *restraints*, things that must not be done. Constraints are the set of activities to enforce while operating the system in an effort to eliminate or mitigate hazards. Similarly, restraints provide wider protections against hazardous system states in that they define restrictions on a controlled process so that certain actions cannot be performed.

RAI traceability is achieved by following procedures for documenting and monitoring the lifecycle of an AI solution from inception to deployment. As part of STPA, constraints and restraints are introduced into the MDLC controlled process upon discovering hazardous states during the design or following deployment of the AI system.

In step two, hazards are traced to one or more losses that may occur when the system enters an undesired state. In step six, constraints are traced to the hazards prevented when they are implemented. There are two types of constraints: 1) those things that must occur during operation to prevent a hazardous state, and 2) the steps to perform that will minimize loss in the event that a hazardous state arises. This thesis focuses upon the former—the set of proposed activities to incorporate into planning and design so that hazardous states can be avoided. Table 4.14:

Table 4.14. Proposed System Constraints

Constraints							
		H1	H2	H3	H4	H5	H6
C1	Incorporate a secondary team to perform independent validation of the model development process to ensure outcomes are consistent using the sonar data obtained	X	X	X		X	
C2	For federated and active learning procedures, utilize automation to the greatest extent possible, documenting requirements, mathematical proofs, algorithms in use, and the expected outcomes through deterministic processes and preconfigured datasets	X	X	X	X		X
C3	Implement a mechanism to quickly revert deployed model to a previously reliable version through the use of a secondary controlled process providing configuration management system functions, maintaining immutable, digitally signed, uniquely identifiable versions of working models, so that, at any time, these versions can be redeployed to the operational environment.			X			X
C4	Abstract hyperparameter tuning and other model refinement activities away from the ancillary tasks required to incorporate these changes into the model for verification. Having an interface that can allow the controller to only modify those specific parameters needed for testing their impact on the model's predictions (inference) will improve traceability as iterative updates are performed	X	X				

In addition to identifying actions that must be taken so that hazardous states are avoided, it is beneficial to consider actions to avoid during design to achieve the same. These restraints are provided in 4.15:

Table 4.15. Proposed System Restraints

Restraints							
		H1	H2	H3	H4	H5	H6
R1	Model Decisions must not be pursued until data acquisition and training thresholds are achieved. These limits might pertain to the number of vessels belonging to a particular class as represented in the dataset, the amount of noise present and the level of filtering required to achieve quality data, the perceived distance of the sound signature from the sensor, etc.	X	X	X		X	
R2	Do not perform federated learning weight aggregation if one or more nodes have exhibited catastrophic forgetting as a result of active and continual learning procedures.	X	X		X		

CHAPTER 5: Conclusion

Artificial intelligence presents promising advantages for processes that require consistent performance, a reduction in the influence of human error, and data-driven predictive behavior that anticipates outcomes based upon observed signals or environmental states. In modern systems, machine learning (or deep learning) enables artificial intelligence that empowers probabilistic methods which are essential to their design. These components are integral during operation. Achieving success when utilizing systems that rely upon machine learning or deep learning based AI requires that performance be measured not only by how well the system achieves classic performance metrics, but how well the solution aligns with responsible AI principles. In this sense, measuring the desired outcome for the system is multi-dimensional; some dimensions align with requirements associated with predictive performance, others dimensions require compliance with established responsible AI (RAI) goals of ensuring proper judgment in design, development, and deployment, equitability when producing results, traceability in procedures and processes, reliability when under validation, and the ability to control or govern behavior when unintended behavior emerges.

This thesis demonstrates that it is possible to establish a reusable model for assuring safe and responsible artificially intelligent systems. Traceability can be achieved by the use of a system analytical process used in system safety engineering. STPA is traditionally used as a means of designing hazardous system states out of safety critical systems. Here it is adapted to systems that utilize machine learning, where system design practices that produce artificially intelligent components may introduce undesired outcomes. When system analytical methods are used to assess hazards, unsafe actions, and potential losses that would occur as these systems are used, there must be a means of applying the same analytical processes across all components.

This thesis demonstrates how STPA can be used to evaluate system components which contain controlled processes employing MDLC, where decision-making about data, model design, and post-deployment monitoring occurs. By applying STPA to the MDLC process, it is shown that STPA is well-suited to address concerns of explainability and overall traceability within a system. Responsible artificial intelligence requires that those designing such systems incorporate methods for ensuring accountability. The techniques demonstrated in this thesis provide a basis for adopting STPA as an accountability framework that can be expanded beyond traditional system safety domains to those that arise due to the emergence of artificial intelligence.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] K. Jushaz, "The History of Coding and Software Engineering," Blog - Galvanize, 2022. Accessed May 12, 2023 . Available: https://www.galvanize.com/blog/the-history-of-coding-and-software-engineering/

[2] S. Misra, V. Kumar, U. Kumar, K. Fantazy, and M. Akhter, "Agile software development practices: evolution, principles, and criticisms," *International Journal of Quality & Reliability Management*, vol. 29, no. 9, 2012 . Available: https://www.emerald.com/insight/content/doi/10.1108/02656711211272863/full/html

[3] "DOD Enterprise DevSecOps Reference Design," Strategy Guide, August 2019 . Available: https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf

[4] M. I. Jordan and T. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, July 2015 . Available: https://doi.org/10.1126/science.aaa8415

[5] Z. Wan, X. Xia, D. Lo, and G. C. Murphy, "How does Machine Learning Change Software Development Practices?" *IEEE Transactions on Software Engineering*, vol. 47, no. 9, pp. 1857–1871, 2021 . Available: https://doi.org/10.1109/TSE.2019.2937083

[6] R. Arora and N. Arora, "Analysis of SDLC models," *International Journal of Current Engineering and Technology*, vol. 6, no. 1, pp. 268–272, 2016.

[7] R. Ranawana and A. S. Karunananda, "An Agile Software Development Life Cycle Model for Machine Learning Application Development," in *2021 5th SLAAI International Conference on Artificial Intelligence (SLAAI-ICAI)*, 2021, pp. 1–6 . Available: https://doi.org/10.1109/SLAAI-ICAI54477.2021.9664736

[8] J. Jordan, "Organizing machine learning projects: Project management guidelines." Jeremy Jordan Website, 2018. Accessed May 12, 2023 . Available: https://www.jeremyjordan.me/ml-projects-guide/

[9] K. Hicks, "Implementing Responsible Artificial Intelligence in the Department of Defense." Official Memorandum, May 2021 . Available: https://media.defense.gov/2021/May/27/2002730593/-1/-1/0/IMPLEMENTING-RESPONSIBLE-ARTIFICIAL-INTELLIGENCE-IN-THE-DEPARTMENT-OF-DEFENSE.PDF

[10] T. Miller, "Explainable AI is Dead, Long Live Explainable AI! Hypothesis-driven decision support," *arXiv e-prints*, p. arXiv:2302.12389, Feb. 2023 . Available: https://doi.org/10.48550/arXiv.2302.12389

[11] S. N. Goodman, D. Fanelli, and J. P. Ioannidis, "What does research reproducibility mean?" *Science translational medicine*, vol. 8, no. 341, pp. 341ps12–341ps12, 2016.

[12] A. D. Selbst, D. Boyd, S. A. Friedler, S. Venkatasubramanian, and J. Vertesi, "Fairness and Abstraction in Sociotechnical Systems," in *Proceedings of the Conference on Fairness, Accountability, and Transparency* (FAT* '19). New York, NY, USA: Association for Computing Machinery, 2019, p. 59–68 . Available: https://doi.org/10.1145/3287560.3287598

[13] J. M. Kleinberg, S. Mullainathan, and M. Raghavan, "Inherent Trade-Offs in the Fair Determination of Risk Scores," *CoRR*, vol. abs/1609.05807, 2016 . Available: http://arxiv.org/abs/1609.05807

[14] E. W. Jatho, L. O. Mailloux, S. Rismani, E. D. Williams, and J. A. Kroll, "System Safety Engineering for Social and Ethical ML Risks: A Case Study," 2022.

[15] T. M. Powers and J.-G. Ganascia, "The Ethics of the Ethics of AI," in *The Oxford Handbook of Ethics of AI*. Oxford University Press, 07 2020 . Available: https://doi.org/10.1093/oxfordhb/9780190067397.013.2

[16] C. Belitz, L. Jiang, and N. Bosch, "Automating Procedurally Fair Feature Selection in Machine Learning," in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society* (AIES '21). New York, NY, USA: Association for Computing Machinery, 2021, p. 379–389 . Available: https://doi.org/10.1145/3461702.3462585

[17] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017 . Available: https://doi.org/10.1073/pnas.1611835114

[18] G. Rizos and B. W. Schuller, "Average Jane, Where Art Thou? – Recent Avenues in Efficient Machine Learning Under Subjectivity Uncertainty," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, M.-J. Lesot *et al.*, Eds. Cham: Springer International Publishing, 2020, pp. 42–55.

[19] N. Bouacida and P. Mohapatra, "Vulnerabilities in Federated Learning," *IEEE Access*, vol. 9, pp. 63 229–63 249, 2021 . Available: https://doi.org/10.1109/ACCESS.2021.3075203

[20] S. Passi and M. Vorvoreanu, "Overreliance on AI: Literature Review," Microsoft, Rep. MSR-TR-2022-12, June 2022 . Available: https://www.microsoft.com/en-us/research/publication/overreliance-on-ai-literature-review/

[21] V. Dignum, *Introduction*. Cham: Springer International Publishing, 2019, pp. 1–7 . Available: https://doi.org/10.1007/978-3-030-30371-6_1

[22] "Responsible Artificial Intelligence Strategy and Implementation Pathway," Strategy Guide, June 2022 . Available: https://www.ai.mil/docs/RAI_Strategy_and_Implementation_Pathway_6-21-22.pdf

[23] J. Fischer, M. Orescanin, P. Leary, and K. B. Smith, "Active Bayesian Deep Learning With Vector Sensor for Passive Sonar Sensing of the Ocean," *IEEE JOURNAL OF OCEANIC ENGINEERING*, 2023.

[24] Datacamp, "Active Learning: Curious AI algorithms," Datacamp, 2018. Accessed May 7, 2023 . Available: https://www.datacamp.com/tutorial/active-learning

[25] B. Settles, "Active Learning Literature Survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009 . Available: http://axon.cs.byu.edu/~martinez/classes/778/Papers/settles.activelearning.pdf

[26] R. Burbidge, J. J. Rowland, and R. D. King, "Active learning for regression based on query by committee," *Lecture Notes in Computer Science*, vol. 4881, pp. 209–218, 2007.

[27] S. Mussmann, J. Reisler, D. Tsai, E. Mousavi, S. O'Brien, and M. Goldszmidt, "Active Learning with Expected Error Reduction," 2022.

[28] J. Qi, Q. Zhou, L. Lei, and K. Zheng, "Federated reinforcement learning: techniques, applications, and open challenges," 2021 . Available: https://doi.org/10.20517/ir.2021.02

[29] A. Khare, A. Agrawal, M. Lee, and A. Tumanov, "SuperFed: Weight Shared Federated Learning," 2023.

[30] L. Roque, "Primer on Bayesian Deep Learning," Towards Data Science, 2023. Accessed May 16, 2023 . Available: https://towardsdatascience.com/primer-on-bayesian-deep-learning-d06e0601c2ae

[31] S. Horii, "MLE, MAP and Bayesian Inference," Towards Data Science, 2019. Accessed May 16, 2023 . Available: https://towardsdatascience.com/mle-map-and-bayesian-inference-3407b2d6d4d9

[32] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning*. United Kingdom: Cambridge University Press, 2020.

[33] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight Uncertainty in Neural Networks," in *Proceedings of the 32nd International Conference on Machine Learning* (Proceedings of Machine Learning Research), F. Bach and D. Blei, Eds. Lille, France: PMLR, 07–09 Jul 2015, vol. 37, pp. 1613–1622 . Available: https://proceedings.mlr.press/v37/blundell15.html

[34] S. Xiang, "Aleatoric and Epistemic Uncertainty in Deep Learning," Towards Data Science, 2022. Accessed May 16, 2023 . Available: https://towardsdatascience.com/aleatoric-and-epistemic-uncertainty-in-deep-learning-77e5c51f9423

[35] Y. Kwon, J.-H. Won, B. J. Kim, and M. C. Paik, "Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation," *Computational Statistics & Data Analysis*, vol. 142, p. 106816, 2020 . Available: https://doi.org/https://doi.org/10.1016/j.csda.2019.106816

[36] iMerit, "A Comprehensive Introduction to Uncertainty in Machine Learning," iMerit ML Data Solutions, blog, Sept. 1, 2022 . Available: https://imerit.net/blog/a-comprehensive-introduction-to-uncertainty-in-machine-learning-all-una/

[37] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds. Curran Associates, Inc., 2017, vol. 30 . Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/2650d6089a6d640c5e85b2b88265dc2b-Paper.pdf

[38] W. R. Clements, B. V. Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, "Estimating Risk and Uncertainty in Deep Reinforcement Learning," 2020.

[39] B. Settles, "Active Learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 6, no. 1, pp. 1–114, 2012.

[40] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian Active Learning for Classification and Preference Learning," 2011.

[41] M. Kuhn, J. Silge, and E. Hvitfeldt, "Acquisition functions," tidymodels, github - vignettes, Feb. 28, 2022 . Available: https://tune.tidymodels.org/articles/acquisition_functions.html

[42] J. Dr. Thomas, "Systems Theoretic Process Analysis (STPA) Tutorial," Presentation, Massachusetts Institute of Technology, 2013 . Available: https://psas.scripts.mit.edu/home/wp-content/uploads/2014/03/Systems-Theoretic-Process-Analysis-STPA-v9-v2-san.pdf

[43] B. Hutchinson *et al.*, "Towards Accountability for Machine Learning Datasets: Practices from Software Engineering and Infrastructure," 2021. _eprint: 2010.13561.

[44] J. Kroll, "The fallacy of inscrutability," *Philosophical Transactions of The Royal Society A Mathematical Physical and Engineering Sciences*, vol. 376, p. 20180084, Nov. 2018 . Available: https://doi.org/10.1098/rsta.2018.0084

[45] L. Xing and S. V. Amari, "Fault Tree Analysis," in *Handbook of Performability Engineering*, K. B. Misra, Ed. London: Springer London, 2008, pp. 595–620 . Available: https://doi.org/10.1007/978-1-84800-131-2_38

[46] N. S. C. o. A. Intelligence, "Final Report," Rep., Mar. 2021 . Available: https://www.nscai.gov/2021-final-report/

[47] N. G. Leveson, *Engineering a safer world: Systems thinking applied to safety*. The MIT Press, 2016.

[48] Y. Qi, Y. Dong, X. Zhao, and X. Huang, "STPA for Learning-Enabled Systems: A Survey and A New Method," 2023.

[49] M. Li, F. Yan, R. Niu, and N. Xiang, "Identification of causal scenarios and application of leading indicators in the interconnection mode of urban rail transit based on STPA," *Journal of Rail Transport Planning & Management*, vol. 17, p. 100238, 2021 . Available: https://doi.org/https://doi.org/10.1016/j.jrtpm.2021.100238

[50] N. G. Leveson and J. P. Thomas, "STPA Handbook," 2018 . Available: https://psas.scripts.mit.edu/home/get_file.php?name=STPA_handbook.pdf

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California