**DOT/FAA/TC-23/54**

Federal Aviation Administration
William J. Hughes Technical Center
Aviation Research Division
Atlantic City International Airport
New Jersey 08405

# Assurance of Machine Learning-Based Aerospace Systems: Towards an Overarching Properties-Driven Approach

September 2023

Final report

U.S. Department of Transportation
**Federal Aviation Administration**

## NOTICE

This document is disseminated under the sponsorship of the U.S. Department of Transportation in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof. The U.S. Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the objective of this report. The findings and conclusions in this report are those of the author(s) and do not necessarily represent the views of the funding agency. This document does not constitute FAA policy. Consult the FAA sponsoring organization listed on the Technical Documentation page as to its use.

This report is available at the Federal Aviation Administration William J. Hughes Technical Center's Full-Text Technical Reports page: actlibrary.tc.faa.gov in Adobe Acrobat portable document format (PDF).

**Form DOT F 1700.7** (8-72) Reproduction of completed page authorized

| | | |
|---|---|---|
| 1. Report No.<br>DOT/FAA/TC-23/54 | 2. Government Accession No. | 3. Recipient's Catalog No. |
| 4. Title and Subtitle<br>Assurance of Machine Learning-Based Aerospace Systems: Towards an Overarching Properties-Driven Approach | | 5. Report Date<br>September 2023<br>6. Performing Organization Code |
| 7. Author(s)<br>Saswata Paul, Dan Prince, Naresh Iyer, Michael Durling, Nikita Visnevski, Baoluo Meng, Mike Meiners, Craig McMillan, Udayan Mandal, Kit Siu, Sarat Chandra Varanasi | | 8. Performing Organization Report No. |
| 9. Performing Organization Name and Address<br>GE Research<br>Michael Durling<br>One Research Circle<br>Niskayuna, NY 12309 | | 10. Work Unit No. (TRAIS)<br>11. Contract or Grant No.<br>692M15-22-T-00012 |
| 12. Sponsoring Agency Name and Address<br>Federal Aviation Administration<br>Srini Mandalapu ANG-E271<br>Atlantic City International Airport, NJ 08405 | | 13. Type of Report and Period Covered<br>Phase I Report<br>14. Sponsoring Agency Code |
| 15. Supplementary Notes | | |

16. Abstract

Traditional process-based approaches of certifying aerospace digital systems are not sufficient to address the challenges associated with using Artificial Intelligence (AI) or Machine Learning (ML) techniques. To address this, agencies are evaluating an alternative Means of Compliance (MoC) called the Overarching Properties (OP). The goals for this research are to develop recommendations and assurance criteria and to explore safety risk mitigation approaches for such AI/ML-based software systems. This document outlines a novel foundation for the application of OPs to support the assurance and certification of complex aerospace digital systems consisting of AI/ML-based components. To this end, we first select the use case of a Recorder Independent Power Supply (RIPS) system. We then perform a Functional Hazard Assessment (FHA) to identify a set of hazards associated with the RIPS and design a set of appropriate requirements to mitigate those hazards.
Using the RIPS as a motivation, we present a novel OP-driven approach that can be used for creating certification and assurance arguments for the ANNs used in the RIPS. We also outline potential techniques for generating assurance cases from our OP arguments using an in-house data curation platform called the Rapid Assurance Curation Kit (RACK).
We also provide a thorough discussion on our experience in using OPs for the certification of AI/ML-based hybrid software systems and how the use of OPs influenced our design and certification strategy with respect to the strategies generally used for the design and certification of traditional software systems.

| | |
|---|---|
| 17. Key Words<br>Avionics – Autonomy<br>Avionics – Trustworthy AI<br>Avionics – Requirement Engineering | 18. Distribution Statement<br>This document is available to the U.S. public through the National Technical Information Service (NTIS), Springfield, Virginia 22161. This document is also available from the Federal Aviation Administration William J. Hughes Technical Center at actlibrary.tc.faa.gov. |

| | | | |
|---|---|---|---|
| 19. Security Classif. (of this report)<br>Unclassified | 20. Security Classif. (of this page)<br>Unclassified | 21. No. of Pages<br>88 | 22. Price |

## Contents

## Figures

## Tables

## Acronyms

| Acronym | Definition |
|---|---|
| ACAS | Airborne Collision Avoidance System |
| AEPHM | Aircraft Electrical Power Systems Prognostics and Health Management |
| AFRL | Air Force Research Laboratory |
| AI | Artificial Intelligence |
| ANN | Artificial Neural Network |
| ARCOS | DARPA Automated Rapid Certification of Software program |
| ARP | Aerospace Recommended Practice |
| AST | Adaptive Stress Testing |
| AVSI | Aerospace Vehicle Systems Institute |
| BHM | Battery Health Monitor |
| CC | Constant Current |
| CI/CD | Continuous Integration and Continuous Deployment |
| COTS | Commercial Off-The-Shelf |
| CV | Constant Voltage |
| DDS | Data Distribution Services |
| DAL | Design Assurance Level |
| DNN | Deep Neural Network |
| DST | Dynamic Stress Testing Profiles |
| EASA | European Union Aviation Safety Agency |
| EOL | End-of-Life |
| EPFD | Electric Powertrain Flight Demonstration |
| ESiP | ENSEMBLE Embedded Software integration Platform |
| FAA | Federal Aviation Administration |
| FAN | Friendly Argument Notation |
| FAR | Federal Aviation Regulations |
| FDAL | Function Development Assurance Level |
| FDR | Flight Data Recorder |
| FHA | Functional Hazard Assessment |
| FUDS | Federal Urban Driving Schedule |
| GAN | Generative Adversarial Network |
| GSN | Goal Structuring Notation |
| HLR | High Level Requirement |

| | |
|---|---|
| IDE | Integrated Development Environment |
| MBD | Model-Based Design |
| MBSE | Model-Based System Engineering |
| ML | Machine Learning |
| MoC | Means of Compliance |
| NAS | National Airspace System |
| NASA | National Aeronautics and Space Administration |
| ONNX | Open Neural Network Exchange Format |
| OP | Overarching Properties |
| OWL | Web Ontology Language |
| PCoE | Prognostics Center of Excellence |
| RACK | Rapid Assurance Curation Kit |
| RIPS | Recorder Independent Power System |
| RUL | Remaining Useful Life |
| SAE | Society of Automotive Engineers |
| SADL | Semantic Application Design Language |
| SoH | State of Health |
| SoC | State of Charge |
| SRL | SADL Requirements Language |
| UAM | Urban Air Mobility |

## Executive summary

The traditional process-based approaches of certifying aerospace digital systems are not sufficient to address the challenges associated with using Artificial Intelligence (AI) or Machine Learning (ML) techniques. To address this, agencies like the National Aeronautics and Space Administration (NASA) and the Federal Aviation Administration (FAA) are evaluating an alternative Means of Compliance (MoC) called the Overarching Properties (OP). The goals for this research are to develop recommendations and assurance criteria and to explore safety risk mitigation approaches for such AI/ML-based software systems. This document outlines a novel foundation for the application of OPs to support the assurance and certification of complex aerospace digital systems consisting of AI/ML-based components.

To this end, we first select the use case of a Recorder Independent Power Supply (RIPS) system that provides several minutes of backup power to the data recorder when an aircraft loses access to standard power supply. Our RIPS design utilizes two Artificial Neural Networks (ANN) to make predictions that can increase the time between maintenance actions of the RIPS battery by reducing unnecessary battery charge/discharge cycles. We then perform a Functional Hazard Assessment (FHA) to identify a set of hazards associated with the RIPS and design a set of appropriate requirements to mitigate those hazards. The two neural networks are then trained using a publicly available dataset on the performance of Lithium-ion batteries and a novel approach is proposed for the design, development, integration, and deployment of the ANNs using Model-Based System Engineering (MBSE) and Model-Based Design (MBD) principles. Using the RIPS as a motivation, we present a novel OP-driven approach that can be used for creating certification and assurance arguments for the ANNs used in the RIPS and outline potential techniques for generating assurance cases from our OP arguments using an in-house data curation platform called the Rapid Assurance Curation Kit (RACK). We also provide a thorough discussion on our experience in using OPs for the certification of AI/ML-based hybrid software systems and how the use of OPs influenced our design and certification strategy with respect to the strategies generally used for the design and certification of traditional software systems.

# 1 Introduction

Current systems, software, and electronic hardware assurance processes for certifying aerospace digital systems are based on explicit, detailed, prescriptive approaches, using “objectives” that must be satisfied to demonstrate compliance with applicable regulations for aircraft certification. These processes do not address the assurance aspects of new technologies such as AI/ML implementations, tools or Commercial Off-The-Shelf (COTS) products proposed for use in civil aircraft development. For example, the traditional DO-178C/254 (DO-178C Software Considerations in Airborne Systems and Equipment Certification, 2011) (DO-254 Design Assurance Guidance for Airborne Electronic Hardware, 2000) process for development and certification of software does not have a clear approach to support the assurance of AI/ML-based components that are developed and qualified using data instead of traditional methods. Additionally, standards and guidance for safe use in civil aircraft do not exist for implementation of methodologies using AI, autonomy, and non-determinism. The National Aeronautics and Space Administration (NASA) recently published a roadmap (Brat, et al., 2023) that identifies potential AI/ML and autonomous functions that may be used in future aerospace applications along with validation and verification gaps. The Federal Aviation Administration (FAA) and NASA have been working on developing a domain independent and technology independent alternate means of compliance called the Overarching Properties (OP) (Holloway, 2019) that are intended to be a sufficient set of properties that can be used for making approval decisions for any entity that is to be used in an aircraft. If the entity can be shown to possess the OPs, then approval can be granted for its use in an aircraft. This document proposes an OP-based methodology for assurance of aerospace applications that include AI/ML-based components.

The goals for this research are to develop recommendations and assurance criteria and to explore safety risk mitigation approaches for such AI/ML-based software systems (Paul, et al., 2023). As a use case to motivate our study, we selected a Recorder Independent Power Supply (RIPS) system that provides several minutes of backup power when an aircraft loses access to standard power supply. Our hybrid RIPS design is augmented with a battery health monitor (BHM) that uses artificial neural networks (ANN) to make predictions. These predictions can increase the time between maintenance actions of the RIPS battery by reducing unnecessary battery charge/discharge cycles. Using the RIPS as a motivation, we present a novel OP-driven approach that can be used for creating assurance arguments for AI/ML-based components. We develop a set of argument structures that consider the non-traditional design parameters of AI/ML-based components and use appropriate strategies to generate supporting evidence for different Design Assurance Levels (DAL). We describe our experience in developing appropriate requirements

for the RIPS system via an iterative requirements-engineering process which initiates as a traditional approach and is gradually informed by the design aspects of the ANNs, such as the data used for development and qualification. In addition, we also demonstrate a unique, advanced approach for the design, development, integration, and deployment of AI/ML-based systems using Model-Based System Engineering (MBSE) and Model-Based Design (MBD) principles.

# 2 Use case selection

This section describes the process followed, decision criteria, options considered, and the aerospace AI/ML application use case option selected for Phase 1 of the GE led FAA sponsored Assurance of AI/ML Research project.

## 2.1 Decision criteria

Listed below are the criteria that we used for choosing the use case for Phase 1:

- **FAA relevance**

  Our goal was to choose an application that would be relevant to the Federal Aviation Administration's (FAA) initiatives towards ensuring safe aircraft operations in the US National Airspace System (NAS). Given the recent advancements in the aviation industry towards greener hybrid-electrical propulsion systems, our selected use case is important because the degradation of battery health in such systems can lead to sudden or gradual loss of propulsion power that can lead to catastrophic loss-of-thrust situations (Sripad, Bills, & Viswanathan, 2021). The FAA has been involved in taking steps for the certification of electric aircraft engines that may rely on battery power (FAA, 2021) to prevent such incidents from happening.

- **Alignment to GE Aviation's interests**

  We wanted to select a use case that would be well-aligned with the current work being done in GE Aviation to revolutionize the future of flight. In 2021, GE Aviation entered a partnership with the National Aeronautics and Space Administration (NASA) for developing integrated hybrid-electric powertrain systems as a part of NASA's Electrified Powertrain Flight Demonstration (EPFD) project (GE, 2021). This makes our selected use case pertinent to GE Aviation's business interests.

- **Data pedigree**

  Neural networks models are designed by training them on data. Therefore, data quality plays a very important role in the quality of the models that are created. This criterion was considered to ensure that we train and validate our neural network model with data that can be traced back to a trusted source so that we can have high confidence in the quality of the data. Therefore, we selected an existing dataset on Lithium-Ion (li-ion) battery health collected and published online by NASA (Kulkarni, Hogge, Quach, & Goebel, 2007) and previously used in the literature for developing battery health-prediction ML models (Hogge, Bole, Vazquez, & Celaya, 2015).

- **Model complexity**

  The complexity and randomness associated with machine-learning models make it challenging to formally verify properties about them. Therefore, another consideration behind our use case selection was to analyze a neural network model that would be simple enough to be amenable to formal analysis, but complex enough to justify the use of a neural network for the task. As a starting point for model development, we studied an existing neural network model (Sanabria, 2019) that was designed using battery dataset published by NASA.

- **EASA AI/ML autonomy level**

  In this phase of the project, our objective was to select an application that aligns with the European Union Aviation Safety Agency's (EASA) AI Roadmap Level 1 that is intended to assist crew in performing tasks to prepare for flight. Our selected use case of a neural network-based system for battery health monitoring can be used by the crew for estimating the battery health during pre-flight performance analysis of an aircraft and aligns well with EASA AI Roadmap Level 1. In the future, the use case can potentially be enhanced to monitor the battery health during flights to assist human and autonomous controllers make important operational decisions by calculating flight-parameters such as the remaining flight time.

- **Hazard consequence level**

  Another criteria we considered for our use case selection in Phase 1 was to investigate an aviation-related application which will have low consequences in case of a failure. The selected use case for a pre-flight battery health monitor has a Hazard Consequence Level

"Minor" and a Design Assurance Level (DAL) "D". In the future, we plan to consider use cases with higher failure consequences.

## 2.2 Options considered

Based on the stated decision criteria, we considered the following alternatives as use-case options to pursue in Phase I of the effort:

- **Lithium-ion battery health monitoring**

  The high energy density and efficiency make li-ion batteries the most promising candidates for electrification of aviation functions, including propulsion. The criticality of this use-case revolves around loss of power to aircraft that is propelled by li-ion batteries. This use-case will involve development of an AI-based, data-driven model for the estimation of characteristics of a battery that directly relate to its ability to meet the power demand over a future time-period. Multiple such characteristics have been explored in literature, including state of health (SoH) (GE, 2021; Sanabria, 2019), state of charge (SoC) (Ravaioli, et al., 2022; SafeRL, n.d.) and the prediction of remaining useful life (RUL) under cycling degradation (Litjens, et al., 2017; Dean A. Pomerleau, 1992). A good survey of around 30 datasets that are available to be used for this use-case is provided in Cofer et al. (2020) – some of these are publicly available, others require a request for data, and few others are not publicly available.

  Initially, we looked at the CALCE Battery Research repository, specifically at the dataset mentioned in Chen et al. (2015) which deals with the problem of estimating state-of-charge (SoC) of LiFePO4 batteries as a function of its previous voltage, current and temperature measurements. Simulated dynamic stress testing profiles (DST), that capture variability of real-world loading conditions were used to collect the measurements in a laboratory, to use as training data. This was used for developing an Artificial Neural Network (ANN) based model for predicting SoC of the battery, while also validating it using data that capture benchmark driving schedules like the US06 driving schedule and the federal urban driving schedule (FUDS). However, this dataset does not capture the ground-truth of the SoC directly as a field and would have required its analytical estimation as a precursor to the model development activity. In response, we then evaluated the battery datasets repository made available by the Prognostics Center of Excellence within the NASA Ames Research Center (Julian, Kyle D., Mykel J. Kochenderfer, and Michael P. Owen, 2019). The dataset evaluated consisted of a set of four li-ion batteries, that were run through three different operational profiles at room

temperature. Charging was carried out in a constant current (CC) mode at 1.5A until the battery voltage reached 4.2V and then continued in a constant voltage (CV) mode until the charge current dropped to 20mA. Discharge was carried out at a constant current (CC) level of 2A until the battery voltage fell to 2.7V, 2.5V, 2.2V, and 2.5V for batteries 5, 6, 7, and 18, respectively. The repeated charge and discharge cycles result in accelerated aging of the batteries; the experiments were stopped when the batteries reached end-of-life (EOL) criteria, which was a 30% fade in rated capacity (from 2Ahr to 1.4Ahr). The goal for the current effort would be to use this dataset to build an ANN model that can predict SoH for a battery.

- **AFRL control function benchmarks**

  Machine learning is being exploited to approximate control laws in the aerospace context. One of the use cases we reviewed under this project is a benchmark for 2D Spacecraft Docking published by the Air Force Research Laboratory (Ravaioli, et al., 2022; SafeRL, n.d.). The idea is that an active deputy spacecraft utilizes a neural network to learn to dock with a passive chief spacecraft. At each step, the ML-based controller generates force to propel the deputy spacecraft towards the chief. The deputy is considered successfully docked when its distance to the chief is less than a predefined threshold. In the meantime, the deputy must also adhere to a velocity safety constraint that dynamically changes as it approaches the chief. However, due to the lack of relevance to FAA and GE Aviation, it is not selected as the use case for this project.

- **Perception**

  Neural networks are commonly used to read and classify content in images. Examples include character recognition, disease detection (Litjens, et al., 2017), object classification, and vision for autonomous vehicles (Dean A. Pomerleau, 1992). In the aerospace domain, researchers have explored using neural networks-based vision to perceive alignment with the center line of a runway (D. Cofer et al, 2020). They may also be used to perceive safe landing areas, their location, and identify potential obstacles such as other aircraft, birds, structures, and terrain. Neural network-based perception is a critical function in the autonomous automotive domain (Chen, C.; Seff, A.; Kornhauser, A.; Xiao, J., 2015) and appears to have great potential to enable autonomous and reduced crew in the aerospace domain. Therefore, assurance for neural network-based perception is identified as a high priority application for this research. Further, it is understood to be a very challenging technical topic.

- **Collision avoidance**

  Airborne Collision Avoidance System (ACAS) is a family of aircraft collision avoidance systems for unmanned aircraft systems. It uses large numeric lookup tables to generate both horizontal and vertical maneuver guidance in order to reduce the risk of mid-air collisions or near mid-air collisions between aircrafts. Recent research work leveraged a deep neural network to approximate the compress collision avoidance tables for ACAS Xu (Julian, Kyle D., Mykel J. Kochenderfer, and Michael P. Owen, 2019) and ACAS sXu ( Irfan, Ahmed, Kyle D. Julian, Haoze Wu, Clark Barrett, Mykel J. Kochenderfer, Baoluo Meng, and James Lopez, 2019) which showed comparable performance to the original lookup tables with significantly less storage space. The safety properties about the neural networks have been extensively studied and verified in various research activities The Deep Neural Network (DNN) representation of the ACAS Xu is also publicly available (ACAS Xu DNN Repository, 2017). The ACAS Xu has seven input parameters (range to intruder, bearing angle to intruder, relative heading angel of intruder, ownship speed, intruder speed, time to loss of vertical separation, previous advisory) and produces scores for five advisories, which are Clear of Conflict, Weak Left, Weak Right, Strong Left, and Strong Right. There are 50 neural networks tainted for ACAS Xu, one for each combination of previous advisory and time to loss of vertical separation to reduce runtime to evaluate the networks. However, this use case does not align with GE Aviation's interest, thus is not selected.

- **Propulsion system health monitor**

  We looked for existing available datasets that emulate critical failures for propulsion systems of electric aircraft, but did not find many instances, although publications indicate that this is an area of enquiry that is accelerating. In (Palanisamy, Rajendra prasath & Kulkarni, Chetan & Corbetta, Matteo & Banerjee, Portia, 2022) (Kulkarni, C. S., Corbetta, M., & Robinson, E., 2020), diagnostic frameworks for fault isolation in electric powertrains of unmanned aerial vehicles are explored, where FMEA and physics-based simulations are used to analyze the failures of interest and use them for fault isolation. While this body of work itself did not produce data that can be used to develop an AI-based model for the effort, it provides an insight regarding how a good simulator can be leveraged to generated ample data required for the development of an AI-based model. The Aircraft Electrical Power Systems Prognostics and Health Management (AEPHM) program being worked by Air Force Research Laboratories (AFRL), and Boeing and Smiths Aerospace has published work (Keller, K., Kevin Swearingen, J J

Sheahan, Michelle D. Bailey, Jonathan Mark Dunsdon, Katarzyna Przytuła and Brett Jordan, 2006). It describes prognostics and health management (diagnostics, prognostics and decision aids) for electrical power systems including electric actuation, fuel pumps/valves and wiring, including the generation of test data to characterize degraded systems, and the development of algorithms for system health management to support maintenance and operational decision-making. We will explore if this dataset is publicly available for use in our program. When considering electric motors outside the aviation industry, we anticipate abundance of datasets to be available related to the failure and health of components of motors in terrestrial applications both within and outside GE (Cheng Wang, Tongtong Ji, Feng Mao, Zhenpo Wang, Zhiheng L, 2021). Goebel & Saha (2015) provide surveys related to PHM for electric vehicles and propulsion. If this use-case is selected for further investigation, we will further explore these repositories for use in this program.

## 2.3 Selected option

The team considered the options shown below in Table 1. For the first year/phase of the research, the battery health monitor seemed best. It had high scores in terms of relevance to the FAA and GE, data pedigree (from NASA published source), and is a non-safety critical Level-1 AI function. The propulsion system health monitor appears to be a good choice for the next phase of the work. It is also well aligned with the FAA and GE interests and is more complex than the battery health monitor because it is a system that includes a diverse set of components. It may also be considered a Level-2 system if the use case includes feedback to human pilots during flight. The most challenging and thus the final application will be a perception function. AI/ML functions have been used to classify and identify objects in images. A perception function in aerospace may be used to identify potential objects such as mountains, other aircraft, birds, and runway lines. The team will build up the baseline capabilities to argue assurance for less complex and safety critical functions at the beginning of the project, then use the experience to develop an approach for the more challenging perception function.

Table 1. Comparison of the selected and considered use cases

| | **Selected Use Case (Phase 1)** | **Considered Alternative 2** | **Considered Alternative 3** | **Considered Alternative 4** | **Considered Alternative 5** |
|---|---|---|---|---|---|
| **Decision Criteria** | Battery Health Monitor | Perception Function | Collision Avoidance Function | Control Function | Propulsion System Health Monitor |
| **FAA Relevance** | 9 | 9 | 9 | 1 | 9 |
| **Alignment to GE Aviation's Interests** | 9 | 9 | 1 | 1 | 9 |
| **Data Pedigree** | 9 | 9 | 9 | 9 | 9 |
| **Model Complexity** | 1 | 9 | 5 | 9 | 5 |
| **AI Autonomy Level** | 1 or 2 | 2 or 3 | 2 or 3 | 3 | 1 or 2 |
| **Hazard Consequence Level** | Varies by Application | Varies by Application | Catastrophic, Hazardous, or Major | Varies by Application | Varies by Application |

# 3 Use case: the Recorder Independent Power System (RIPS)

The RIPS is an alternate power source that supplies direct current voltage to the flight data recorder (FDR), for 10 minutes whenever the primary aircraft power is removed. The desired behavior is to ensure continued recording following a loss of primary power associated with an aircraft emergency. For this design, the aircraft supplies power to the RIPS. The RIPS provides this aircraft power to the FDR, when available, and backup battery power to the FDR, when aircraft power is not available. In this way, all power to the FDR passes through the RIPS, so the primary system function of the RIPS is simply to "Supply Power to the FDR". This function is captured in Table 2.

Table 2. Function list

| Function | Rationale |
|---|---|
| Supply Power to FDR | Primary function of a RIPS defined in TSO-C155b Section 3.a. |

The system in its operational context is shown in Figure 1, including a high-level representation of the external interfaces to the system. The dashed line indicates the boundary of the system to which this assessment will be applied.

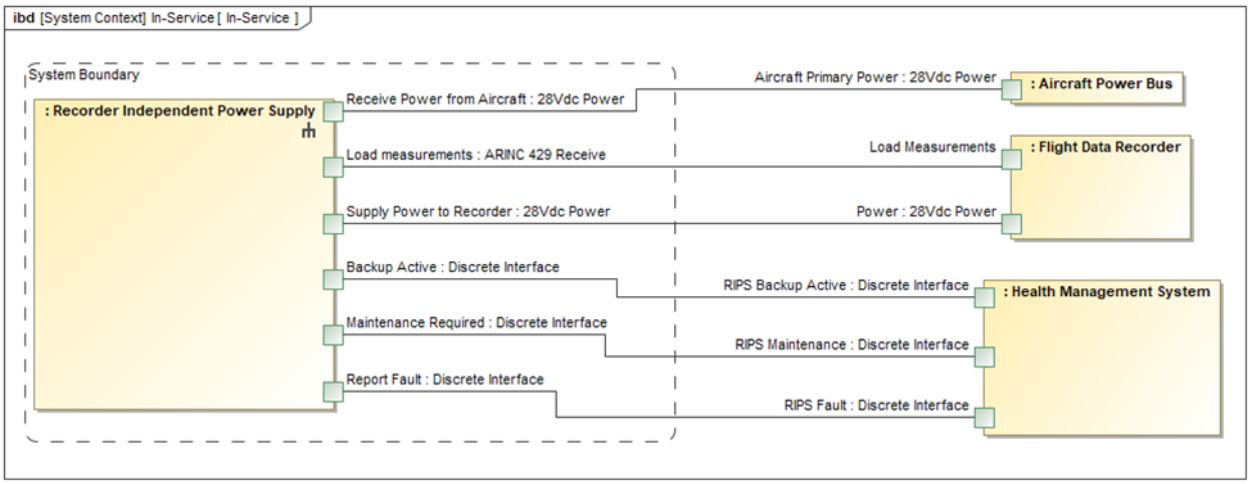



Figure 1. RIPS system scope

The system model (General Elelctric Aviation Systems LLC (US), 2022) was used to create components and component interactions that would allow system requirements to be satisfied. Through this process, the system was subdivided into the components shown in the simplified architecture in Figure 2 and its equivalent internal block diagram representation from the systems model is shown in Figure 3. Of these components, this program will focus on the assessment of the "Battery Health Monitor" component.

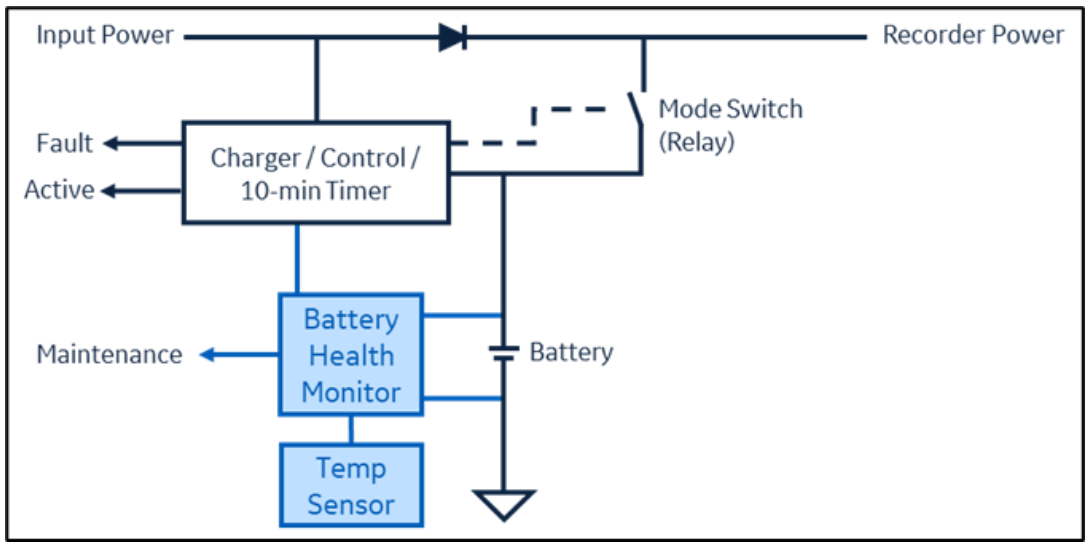



Figure 2. Simplified block diagram

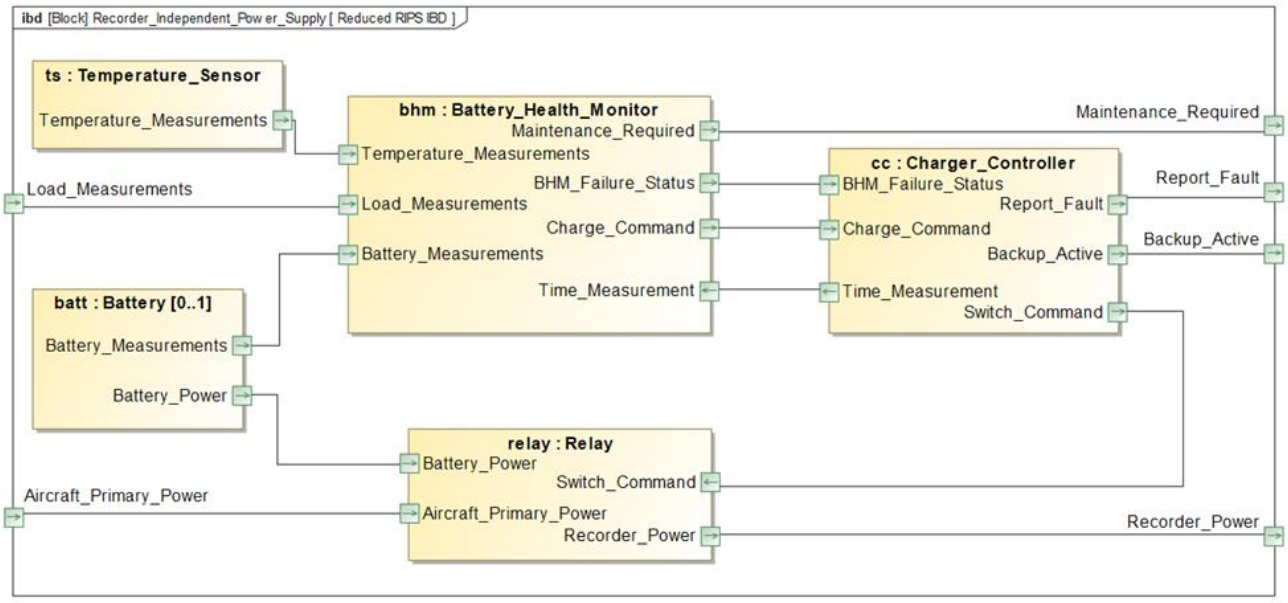



Figure 3. RIPS internal block diagram

## 3.1 Battery Health Monitor (BHM)

This Functional Hazard Assessment (FHA) is performed at the system level to match the certification guidance provided in TSO-C155b. However, the neural network that is being assessed is being evaluated for the implementation of the BHM subcomponent, therefore, this

assessment will additionally describe the BHM subcomponent and its contributions to the hazards.

The BHM is intended to increase the time between maintenance actions for the RIPS battery by reducing unnecessary battery charge/discharge cycles. The performance of this function cannot interfere with the RIPS ability to supply sufficient backup power when needed. To perform this intended behavior, the BHM must provide a charge command when the battery's charge has depleted to the point where it can no longer supply power when needed. Also, the BHM must provide a maintenance indicator to signal that battery health has deteriorated to a point where it needs to be replaced. Both behaviors can have an impact on the hazards identified in section 6.1.

For this application, the BHM will estimate the state of charge (SoC) of the battery and command a full recharge if the SoC of the battery has reached 30% or less of the nominal charge capacity. The BHM will indicate that maintenance is required when the state of health (SoH) of the battery is 70% or less. Definitions for SoC and SoH will be provided in later sections of this report.

Based on this understanding of the component behavior, loss of BHM estimation could result in the inability to charge the battery or the inability to indicate that a replacement is necessary. Also, incorrect estimation of either the state of charge or state of health could produce the same result. In either case, the battery may be unable to supply power when needed. Therefore, loss of BHM estimation and incorrect BHM estimation of state of charge and state of health will be assessed for this application.

# 4 Recorder Independent Power System (RIPS) Functional Hazard Assessment (FHA)

## 4.1 Purpose and scope of the FHA

As part of the GE Assurance of Artificial Intelligence (AI) research program, a Functional Hazard Assessment (FHA) was conducted for the selected use case to support the application of an overarching properties assurance analysis. This FHA was initially conducted in accordance with the guidance provided in the SAE ARP 4761 (1996), then specific aspects of the assessment were formalized in the Semantic Application Design Language (SADL) (2022).

The use case selected is the design of a 777-2 Recorder Independent Power Supply (RIPS) (ARINC, 2010), where a new Battery Health Monitor component has been added. The BHM makes use of a neural network to increase the time between maintenance actions for the RIPS

battery by reducing unnecessary battery charge/discharge cycles and accurately predicting the remaining battery life.

The FHA is intended to provide identification of RIPS failure conditions and an assessment of their effects on the aircraft, the crew, and the occupants. The results of the FHA establish the quantitative and qualitative system safety objectives drive from the failure condition classification (i.e., hazard classification), as well as the Function Development Assurance Level (FDAL) for the RIPS design. These results are fed back into an aircraft-level assessment as an iterative process to ensure a mature design and complete analysis

The FHA considers two modes of failure for each function identified. These failure modes include loss of function and malfunction (detected and undetected erroneous function). Likewise, the FHA considers internal system functions as well as those functions provided by, or provided to, other systems (i.e., exchanged functions).

## 4.2 Hazard assessment approach

The analysis portion of the FHA is captured in an FHA worksheet. The worksheet provides the details necessary to conduct this assessment. Refer to Appendix A for details of the FHA worksheet. The worksheet considers all functions provided by the system and assesses the failure effects on the aircraft, the crew, and its occupants.

In order to ascertain the effect and severity of the system level functional hazards, the analysis considers the following:

1. **Id**: A unique, sequential identifier for each of the functions and failure conditions provided.
2. **Function**: Identifies the action of a system to perform an operational capability. Both functional (internal functions and exchanged functions) failure conditions and external (environmental/emergency/abnormal) failure conditions should be considered. The FHA should also consider failures of an entire function and failures of a portion of a function.
3. **Failure Condition (Hazard Description)**: This is the failure mode of the particular function and includes failure modes of the loss-of or undetected erroneous type. In addition, single and combinational failures are considered. Each hazard within the FHA worksheet is preceded by an alphanumeric identifier (e.g., A1) to distinguish the hazard from others related to the same function. The alpha character represents the specific hazard being investigated (described below). The numeric is a sequential number quantifying the number of hazards of that specific type associated with that

function. The following two hazard types are considered for each function being investigated.

4. **Loss of Function**: Identifies total loss of the function or where applicable partial loss.
5. **Undetected Malfunction**: Used to identify undetected erroneous behavior.
6. **Phase**: The flight phases considered in this analysis are defined in Table 3. Each failure condition within the FHA is assessed in each of these flight phases when establishing the failure's effect on the aircraft, crew, and occupants. This ensures that all phases are addressed when considering the failure condition. If it is determined that a failure condition has more than one severity classification based on the failure effect, the analysis is extended to a new line such that each flight phase with a differing severity classification is evaluated independently.
7. **Effect of Failure Condition on Aircraft /Crew/Occupants**: A description of the hazard effects on the airplane, crew, and occupants onboard. This description should be detailed enough to allow the hazard to be classified with certainty based on the terminology provided in the applicable guidance material (e.g., AC 25.1309 or MIL-STD-882). Consideration is given as to whether the failure may be annunciated or unannunciated, and possible mitigating actions that could be taken by the crew.
8. **Failure Condition Classification**: This is the severity classification for the failure condition identified. Where applicable, regulatory and industry guidance is used in determining the severity classifications for the hazards as captured within the FHA worksheet.
9. **Min FDAL**: This is the minimum functional development assurance level proposed for the function based on the failure condition severity classification identified. Refer to section 5.3 for more information on assigning the FDAL.
10. **Failure Condition Classification Justification**: Identifies specific regulatory requirements or guidance for establishing the severity classification for the specific hazard addressed. In some cases, there may be other means for providing the failure condition justification which includes aircraft architecture, analysis, or engineering judgment.
11. **Verification Method**: Identifies the planned verification method for the safety objectives associated with the failure condition (e.g., Specific analyses, test, etc.).
12. **Remarks**: This is a comments section for adding additional text that applies to any aspect of the functional analysis.

## 4.3 Guidance

The RIPS safety objectives generated by this analysis utilizes information and guidance from the applicable regulatory and guidance material from the following:

- Federal Aviation Administration (FAA) Advisory Circulars (ACs)
- Federal Aviation Regulations (FAR)
- Society of Automotive Engineers (SAE) International Aerospace Recommended Practice (ARP) 4754A and 4761

The definition for different phases of flight and the abbreviations used in the FHA are presented in Table 3.

Table 3. Flight phase definitions

| **Flight Phase Abbreviation** | **Flight Phase Name** | **Definition** |
|---|---|---|
| STD | Standing | The aircraft is stationary – prior to pushback or taxi and after arrival at gate, ramp, or parking area. |
| PBT | Pushback/ Towing | The aircraft is moving in the gate, ramp, or parking area assisted by a tow vehicle. |
| TXI | Taxi | The aircraft is moving under its own power on the surface prior to takeoff or after landing. |
| TOF | Takeoff | Application of takeoff power through rotation to an altitude of 35 feet above runway elevation. |
| ICL | Initial Climb | End of takeoff to first prescribed power reduction or until reaching 1000 feet above runway elevation or the Visual Flight Rules (VFR) pattern. |
| ENR | En Route (Cruise) | Instrument Flight Rules (IFR): completion of initial climb through cruise altitude and completion of controlled descent to the Initial Approach Fix (IAF).<br>VFR: completion of initial climb through cruise and controlled descent to the VFR pattern altitude or 1000 feet above runway elevation. |
| APR | Approach | IFR: From IAF to the beginning of the landing flare.<br>VFR: From point of VFR pattern entry or 1000 feet above the runway elevation. |
| LDG | Landing | From the beginning of the landing flare until aircraft exits the landing runway, comes to a stop on the runway, or when the power is applied for takeoff in the case of touch and go landing. |
| ALL | All | All phases of operation. |

In keeping with the guidelines defined in ARP 4761, this FHA contains the following information:

- A list of associated functions to be assessed.
- The scope of the FHA including the system boundary to which the assessment applies and the failure condition classification criteria to be applied.
- An overview of the approach taken.
- A summary of the results of the FHA and recommendations identified during the assessment.
- The FHA worksheet which identifies the functions, their applicable failure conditions, the effects of each failure condition and a severity classification of each failure condition.

## 4.4 Acceptance criteria

A failure condition is defined as a condition with an effect on the aircraft and its occupants, both direct and consequential, caused or contributed to by one or more failures, considering relevant adverse operation or environmental conditions. Hazards are classified according to the severity of their effects (refer to AC/AMJ 25.1309 – Arsenal). For the purpose of this FHA, all failure conditions that may result in injury, illness, or death to personnel; damage to, or loss of a system, equipment, or property; or damage to the environment are considered hazards.

As previously stated, this FHA is used to establish safety objectives for the RIPS. These objectives are defined in terms of failure condition severity classification (leading to a quantitative probability of occurrence) and FDAL. Information within this table has been defined in accordance with AC/AMJ 25.1309 – Arsenal and modified by ARP 4754A to define the associated FDALs.

## 4.5 Functional Hazard Assessment (FHA) summary

This FHA identifies the functional failure conditions at the system (RIPS) level and an assessment of their effects. The system description includes the aspects of the system needed to determine these failure conditions and the behaviors of the BHM that may contribute to them. The analysis that supports this FHA is contained in the FHA worksheet (refer to Appendix A). These failure conditions are grouped according to their failure condition classification. The Failure Condition ID corresponds with the ID of the failure condition identified in the FHA worksheet (just like in Table 4).

Table 4. RIPS minor failure conditions

| ID | Failure Condition |
|---|---|
| A1 | Loss of ability to provide backup power |
| B1 | Backup power provided when not required |

# 5 Recorder Independent Power System (RIPS) requirements engineering

During this phase of the program, a typical top-down design process was followed to construct a set of evidence that can be used in exploring arguments to satisfy the overarching properties. This section will describe the process followed, the constructs of the natural language requirements and associated attributes, then the formalized curation of the requirements.

## 5.1 Requirements development process

### 5.1.1 System requirements development

The first step in the systems engineering process is capturing Stakeholder Expectations. For the RIPS that is being designed as a use case for this program, a set of expectations, as shown in Figure 4, was collected from relevant standards, guidelines, and certification documents. TSO-C155b (FAA, 2015) is the Technical Standard Order that forms the certification basis for this RIPS example design. ARINC 777-2 (ARINC, 2010) contains guidance for the development of a RIPS that is interchangeable and installable in multiple aircraft. ED-112A (EUROCAE, 2013) contains minimum operational performance requirements for a RIPS unit. Finally, DO-160G (RTCA, Inc., 2010) contains a set of environmental conditions and test procedures that are used to constrain the conditions in which the RIPS will operate. Additionally, there was a product expectation on this RIPS based on a business case that maintenance costs could be decreased by reducing battery replacement frequency. It is this expectation that eventually drives the need for the neural network designs to more intelligently manage battery charging and maintenance actions.

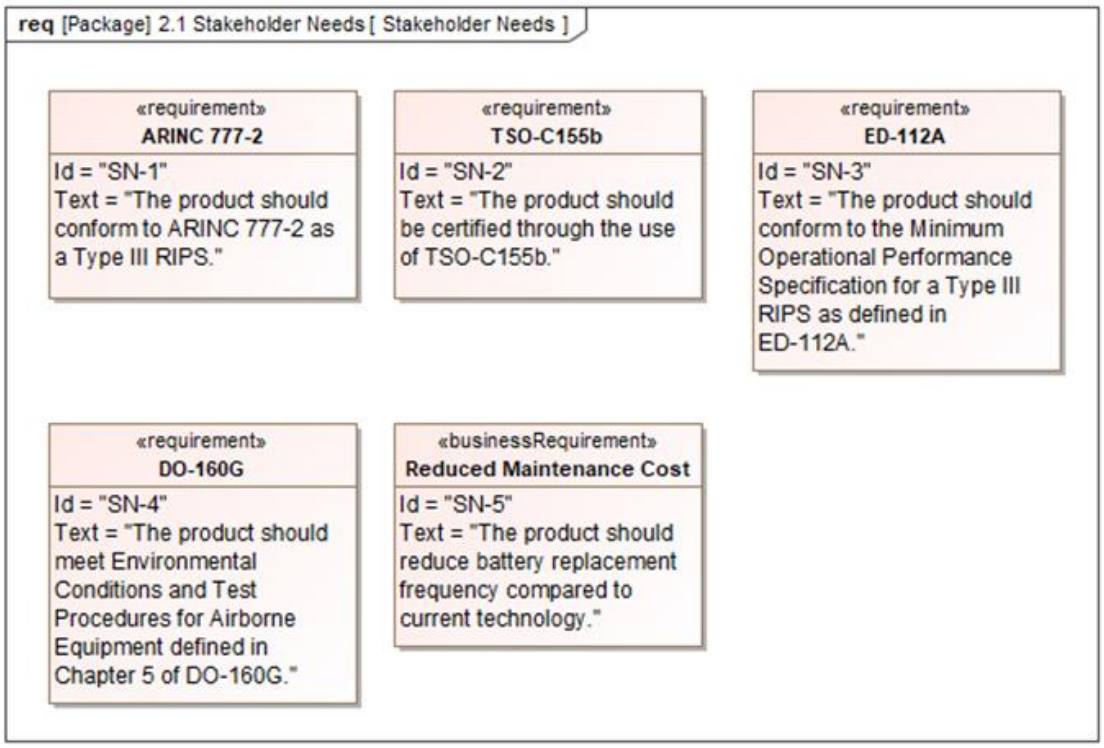



Figure 4. Stakeholder needs

From these stakeholder expectations, a set of RIPS System Requirements is constructed. For the model-based development process being followed, this involves a few steps. First, the desired system behaviors are collected in a set of use cases. Then, with the addition of system boundaries, the use cases are linked to systems and actors external to the product that may interact with the product in one or more contexts. These can be refined through behavioral diagrams to capture relevant parameters and their associated constraints. The use cases drive functional requirements. The system boundaries and interactions with the actors and external systems are used to define necessary interface requirements. Next, the relevant environmental and physical constraints are distilled from the standards. Finally, safety requirements are written corresponding to the hazards identified in the safety assessment.

The system level requirements detail the needs of the system but avoid over constraining with the inclusion of implementation decisions. For this reason, a neural network is not mentioned at this level. However, each type of requirement may have an impact on the neural network(s) being implemented, even though the neural network has not been specified. One or more functional requirements specifies a behavior that the neural network must contribute to. Interface requirements will specify inputs needed to feed the neural network or outputs that are driven by behaviors within the neural network. Physical and environmental constraints will impact the

training and test data that is collected and used in building and verifying the neural network. Then, the neural network will be scrutinized for its ability to impact safety requirements.

### 5.1.2 High-level software requirements development

Within a typical development process, a system architecture is constructed to decompose the system into a set of sub-systems and the items (software and complex electronic hardware) that make up those sub-systems. High Level Requirements (HLRs) are written for those items that detail the behaviors allocated to the item as well as what interactions that item has with other items. Within the RIPS system, only one sub-system (battery health monitor) contains a neural network, so high level software requirements were written for this component only. For this reason, not all system requirements will have a "satisfied by" link down to an HLR. Also, since the worst-case failure condition of the RIPS is Minor, this item would be assigned a Design Assurance Level of "D" and is only required to have HLRs. Therefore, Low Level Requirements (LLRs) are not planned to be developed. In constructing a set of HLRs, functional requirements were written for each of the system-level behaviors allocated to the battery health monitor, including behaviors necessary to drive any outputs. Next, interface requirements and their acceptable ranges were written for all inputs to the sub-system. None of these requirements specify the use of a neural network, and, since this is an implementation decision integral to the product and the performance of this program, derived requirements were written to specify the use of a neural network. As the neural network was being developed, it became clear that aspects of the neural network design, training, and testing were important to ensure that the implementation would meet requirements. Some of these aspects require additional process rigor when collecting data for training or test. An example of this might be the need for configuration management control around the data. Other aspects constitute technical requirements around the data. These have been added to the HLRs as additional derived requirements. For example, training data that does not cover the operational range would cause the neural network to extrapolate, which may not meet the system level design constraint requirements. Training data that contains gaps within the operational range that may reduce the accuracy of the neural network, and also cause the system to not meet these design constraints.

## 5.2 Natural language requirements

### 5.2.1 System requirements

The system requirements table contains columns for the essential elements that make up a requirement (Id, Name, and Text), as well as attributes that would typically be used to satisfy

objectives in ARP-4754A (SAE, 2010) and DO-178C (RTCA, Inc., 2011). Table 5, Table 6, and Table 7 show examples of the following columns:

- **Id** – Unique identifier used in requirement references.
- **Name** – Descriptive title for the requirement used in requirement references.
- **Text** – The natural language requirement.
- **Requirement Type** – The type of requirement being specified. NOTE: Development activities may differ depending on the requirement type.

Table 5. System requirements table (Part 1)

| ID | Name | Text | Requirement Type |
|---|---|---|---|
| RIPS-08 | Maintenance Discrete Behavior | The RIPS shall set the "Maintenance Required" discrete in the "ground" state when the RIPS has determined that the internal battery needs to be replaced. | Functional Requirement |

- **Mitigates Hazard** – If this requirement contributes to mitigating one or more hazards in the safety analysis, this will be populated with the hazard(s).
- **Satisfied By** – This indicates the trace relationship to the child(ren) high-level requirement(s).
- **Architecture Allocation** – This is the architectural element that the requirement governs.
- **Rationale** – Rationale can be included for a number of reasons, including clarifying information on the requirement, an explanation of why this requirement is derived, or clarification of other attributes.

Table 6. System requirements table (Part 2)

| Mitigates Hazard | Satisfied By | Architecture Allocation | Rationale |
|---|---|---|---|
| No | BHM-HLR-18 Cycle Count Input<br>BHM-HLR-01 Monitor Battery… | Recorder Independent Power Supply | [Example Text] |

- **Derived Requirement Indicator** – This indicates if the requirement is derived, as defined in DO-178C (RTCA, Inc., 2011). [Options: Requirement, Derived Requirement]
- **Source** – This is the source(s) of information from the stakeholder needs that was (were) used to create the requirement.

Table 7. System requirements table (Part 3)

| Derived Requirement Indicator | Source |
|---|---|
| Requirement | ARINC 777-2 Section 3.5.3 |

### 5.2.2 High-level software requirements

The high-level requirements table contains columns for the same essential elements that make up a requirement, as well as attributes to satisfy objectives in DO-178C. The requirements table is made up of the following columns with examples shown in Table 8, Table 9, and Table 10.

- **Id** – Unique identifier used in requirement references.
- **Name** – Descriptive title for the requirement used in requirement references.
- **Text** – The natural language requirement.

Table 8. High-level requirements table (Part 1)

| ID | Name | Text |
|---|---|---|
| BHM-HLR-01 | BHM-HLR-01 Monitor Battery | The Battery Health Monitor shall indicate battery maintenance is required when the State-of-Health is less than 70% with a tolerance of +/- 1%. |

- **Requirement Type** – The type of requirement being specified. NOTE: Development activities may differ depending on the requirement type.
- **Mitigates Hazard** – If this requirement contributes to mitigating one or more hazards in the safety analysis, this will be populated with the hazard(s).
- **Satisfies** – This indicates the trace relationship to the parent system requirement.
- **Architecture Allocation** – This is the architectural element that the requirement governs.

Table 9. High-level requirements table (Part 2)

| Requirement Type | Mitigates Hazard | Satisfies | Architecture Allocation |
|---|---|---|---|
| Functional Requirement | Loss of ability to provide backup power | RIPS-08 Maintenance Discrete Behavior | State-of-Health_Neural_Network_Software_Component |

- **Rationale** – Rationale can be included for a number of reasons, including clarifying information on the requirement, an explanation of why this requirement is derived, or clarification of other attributes.
- **Derived Requirement Indicator** – This indicates if the requirement is derived, as defined in DO-178C (RTCA, Inc., 2011). [Options: Requirement, Derived Requirement]

Table 10. High-level requirements table (Part 3)

| Rationale | Derived Requirement Indicator |
|---|---|
| [Example Text] | Requirement |

# 6 Artificial Neural Networks (ANNs) used in the RIPS

As indicated previously, two independent neural network models were trained as components of the BHM system - one for SoH prediction and the other for SoC prediction. The development of a neural network model requires a dataset that captures measurement of relevant battery parameters under conditions that largely mirror operating conditions that is expected to be encountered by the batteries in the actual scenario. This dataset is expected to contain, from a qualitative physics perspective, the critical battery parameters that contain measurements of the output of interest (y), namely SoH and SoC, as well as measurements of the inputs (X) that are expected to be related to those outputs. A data-driven model like a neural network captures a function $f_\theta()$, parameterized by t, such that: $y = f_\theta(X)$, i.e., it produces an empirically valid estimate of the measured output parameter, y, by making use of the measured input parameters, X, in the training dataset. If trained appropriately and if the training dataset sufficiently captures the expected operating conditions in which the model will be used, then we expect the neural network model to provide accurate and reliable estimates of the output parameters, when supplied with input measurements, during operation. The training dataset used for the model as well as details related to both the neural network models are described in the next sections.

## 6.1 Training data

To construct the Neural Network model, an existing dataset for commercial Type 1850 Lithium-ion batteries was used. More specifically, this data set was generated by a custom-built battery prognostics testbed at the Prognostics Center of Excellence (PCoE), NASA (B. Saha and K. Goebel, 2007; NASA, 2022). Four Li-ion batteries were run through three different operational profiles (charge, discharge, and Electrochemical Impedance Spectroscopy) at different temperatures. Discharges were carried out at different current load levels until the battery voltage fell to preset voltage thresholds. Some of these thresholds were lower than that recommended by the OEM (2.7 V) in order to induce deep discharge aging effects. Repeated charge and discharge cycles result in accelerated aging of the batteries. The experiments were stopped when the batteries reached the end-of-life (EOL) criteria of 30% fade in rated capacity (from 2 Ah to 1.4 Ah). In the experiment, multiple batteries repeatedly underwent charging and discharging cycles, as follows:

- *Charging*: Charge the battery with constant electric current 1.5A until the voltage reached 4.2V. Then, charge it with constant voltage until the charging current decreased to 20mA. Stop charging.
- *Discharging:* Discharge with 2A current until the voltage declined to around 2.7V.

The experiment was terminated when the capacity of the battery declined to about 70% of the capacity ratings.

Two independent neural network models were trained using this data, one for SoH prediction and the other for SoC prediction, using the dataset recorded during the discharging cycle. The following inputs were recorded for the discharging cycle which were available to be used for training the neural network model:

*x1. Voltage_measured: Battery terminal voltage (Volts)*
*x2. Current_measured: Battery output current (Amps)*
*x3. Temperature_measured: Battery temperature (degree C)*
*x4. Current_charge: Current measured at load (Amps)*
*x5. Voltage_charge: Voltage measured at load (Volts)*
*x6. Time: Time vector for the cycle (secs)*
*x7. Cycle: the charge-discharge cycle number (each cycle starts with a charging, followed by discharging cycle, as were described above)*
*x8. Capacity: Battery capacity (Ahr) for discharge till 2.7V*

Data from the discharging cycles collected in the above lab setting was considered as emulating actual discharging cycles seen during real-world operation, thereby qualifying for use in training the neural network. We describe the details of the two models next, including initial outcomes from the models.

## 6.2 Adaptation for the RIPS use case

The two output quantities being modeled using the neural networks, namely SoH and SoC, pertain to the states of long-term and short-term degradation of the battery. We provide definitions of the two output quantities and describe model details, training and validation outcomes, all based on the NASA dataset.

### 6.2.1 State of Health (SoH)

The State of Health (SoH) of a battery is a figure of merit, expressed in percentage, to capture the long-term degradation of a battery due to natural and irreversible physical and chemical

processes. Its purpose is to provide an indication of the performance which can be expected from the battery in its current condition or to provide an indication of how much of the useful lifetime of the battery has been consumed and how much remains before it must be replaced. This figure applies over the lifetime of a battery and qualitatively indicates how capable the battery is, at a given cycle during its lifetime, of providing performance relative to when it was brand new. Depending upon the application, multiple parameters can be used to generate this figure of merit. Most commonly, a comparison of the maximum capacity (ampere-hours) of a fully charged battery relative to its maximum capacity when it was brand new is used. By definition, when new, the battery's SoH is 100%, but over time as the battery ages, its SoH begins to reduce. It can be analogously compared to the odometer reading in an automobile, which indicates the number of miles travelled since the vehicle was new, being used as an indicator of the life of the vehicle.

#### *6.2.1.1 SoH Model*

From the variables in the dataset described in the previous section, the SoH model makes use of the following seven inputs in order to estimate SoH: *voltage_measured*, *current_measured*, *temperature_measured*, *current_charge*, *voltage_charge*, *time within cycle*, and *cycle*. This 7-dimensional input is mapped to an equivalent SoH estimated using the variable 'Capacity' in the dataset, as the ratio of the capacity value at the start of discharge cycle to the maximum capacity of the brand-new battery as specified by the manufacturer. This tuple of $<\{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}, \{y\}>$ was used for training the neural network model. In operation, the model takes current measurements of variables $x_1 - x_7$ and produces an estimate of the current SoH. The SoH neural network model is shown in Figure 5 below.

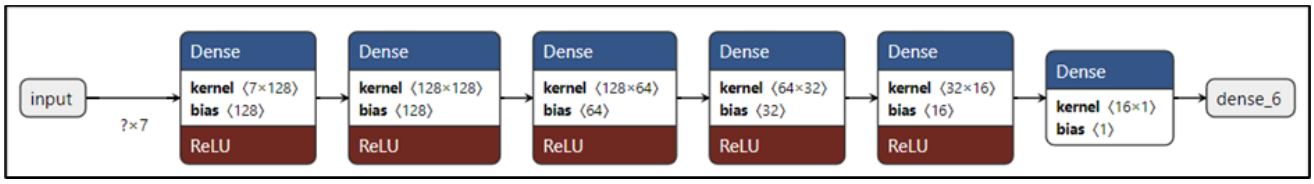



Figure 5. Architecture of the SoH neural network model

Figure 6 below shows model prediction performance in terms of estimating SoH – the blue curve is the ground truth, and the orange curve is the model prediction. Note that this model was trained and validated using three of the four batteries: 'B0005, 'B0006' and 'B0007' (three of the leftmost plots). The rightmost plot in the figure shows the model performance for battery 'B0018' that was not included in the training dataset.

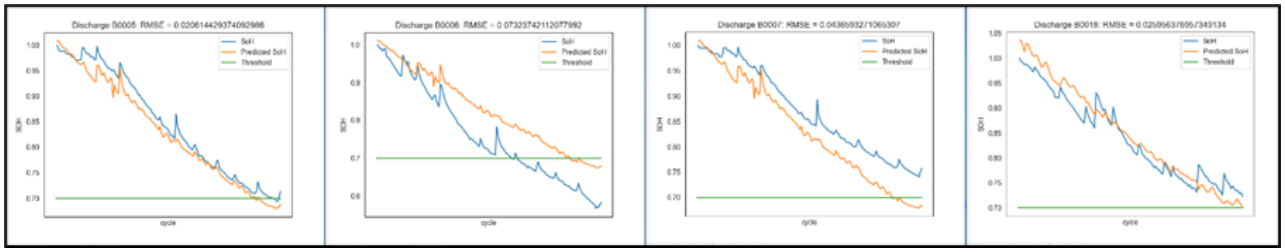


Figure 6. Prediction performance of the SoH model

*Three of the leftmost plots are for batteries used for model development; rightmost plot pertains to battery excluded from the training data.*

### 6.2.2 State of Charge

The State of Charge (SoC) of a battery is a figure of merit defined within a single operational cycle of a battery. Compared to SoH, the SoC provides a more short-term indication of the utility of the battery within a discharging cycle. It is defined as the ratio of its current capacity, Q(t) at time 't' within the cycle, to the maximal capacity ($Q_n$) of the battery, where the maximal capacity is the maximum amount of charge that was stored in the battery at the beginning of that cycle. In other words, if a fully charged battery had a certain amount of maximal energy available to be released during its use, the SoC at time 't' measures what fraction of that energy has been used at 't'. By definition, the SoC of a battery at the beginning of every discharge cycle is 100% and when fully discharged, it becomes 0%. To estimate instantaneous capacity, coulomb counting is used, which performs a direct integral of the current measurement of the battery over time to estimate energy released in that time.

#### *6.2.2.1 SoC Model*

From the variables in the dataset described in the previous section, the SoC model makes use of the following 5 inputs in order to estimate SoH: *voltage_measured, current_measured, temperature_measured, current_charge, and voltage_charge*. Additionally, the model looks at 'n' past values of each variable, each separated by 'k' timesteps, where 'n' and 'k' are parameters. For instance, n=4 and k=2 would imply that the previous 4 values of each of the input variables that are separated by 2 time-steps, are also taken as inputs to the model. This would make the input dimensionality of the model n * 5. For the present SoC model that was explored, the values were set to n=5 and k=2, making the model input 25-dimensional.

This 25-dimensional input is mapped to an equivalent SoC estimated using the time integral of the measured current (instantaneous capacity) and the variable 'capacity' in the dataset, as the ratio of the instantaneous capacity value to the capacity value at the start of discharge cycle. This

tuple of <{$x_i$,(t - n*k)}, {y(t)}> (where i ϵ (1,5), n ϵ (0,5), k ϵ (0,2)) was used for training the neural network model; training was done using data from only one of four the batteries ('B0005') in the dataset. In operation, the model takes current and past measurements of variables $x_1 - x_5$ for the battery and produces an estimate of the current SoH. The SoC neural network model is shown in Figure 7 below.

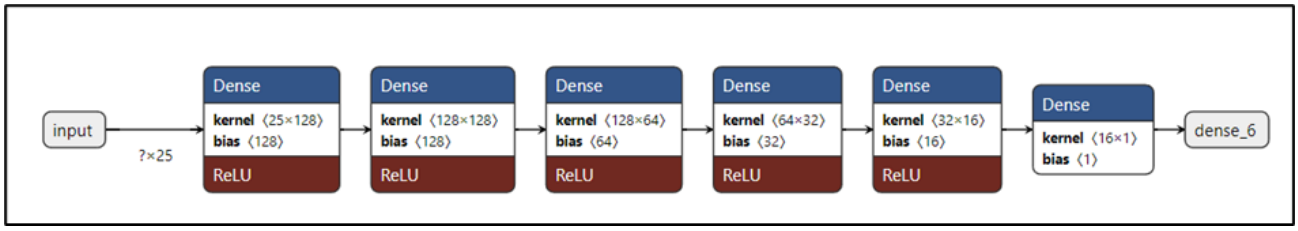



Figure 7. Architecture for SoC neural network model (with n=5, k=2)

Figure 8 shows the plot of multiple curves of SoC over time parameterized by cycles. In other words, 10 random cycles were chosen and the progression of SoC for each of those cycles over time was plotted.

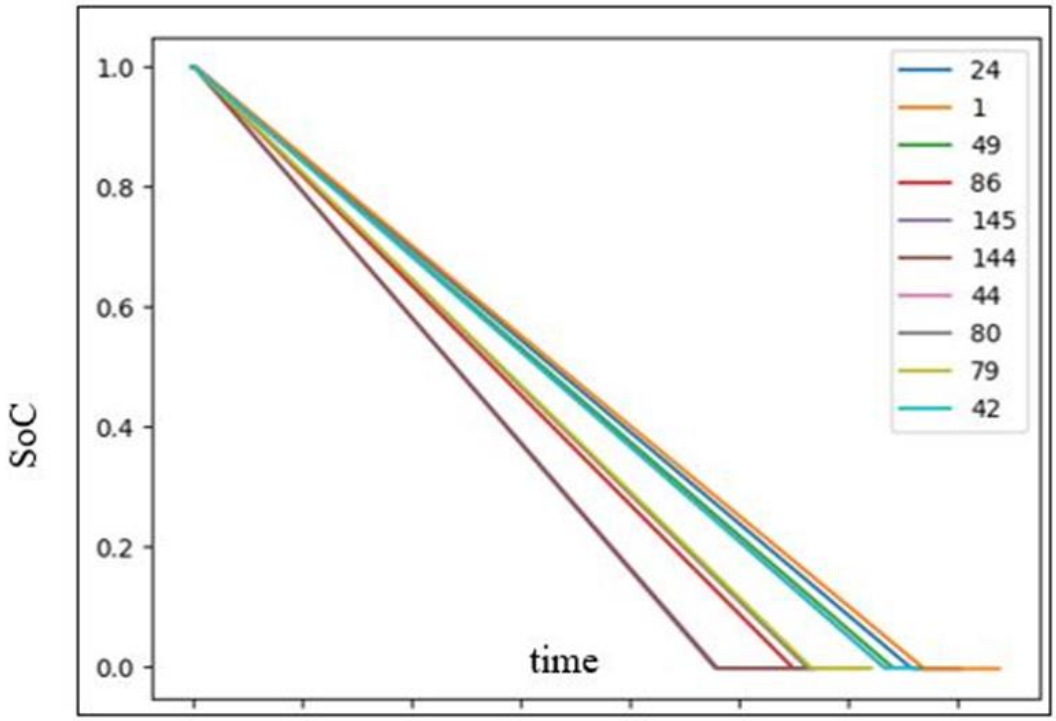



Figure 8. Showing SoC over time for diverse cycles of batteries

The smallest cycle number chosen was 1 (orange curve) and we can see it has the slowest rate of discharge (takes longer to get SoC to 0); in contrast, cycles 144 and 145 are the leftmost curves in the plot that seem to discharge relatively faster compared to cycle 1. Assuming the loads are similar, intuitively, this reduction in time to get to 0 as cycle count increases can also be thought of as the longer-term SoH figure of merit over the life (cycles) of the battery. Figure 9 below shows model prediction performance in terms of estimating instantaneous SoC – the blue curve is the ground truth, and the orange curve is the model prediction. Note that this model was trained and validate using battery 'B0005' (leftmost plot below). The remaining 3 plots in the figure show that the model performance for batteries not included in the training dataset.

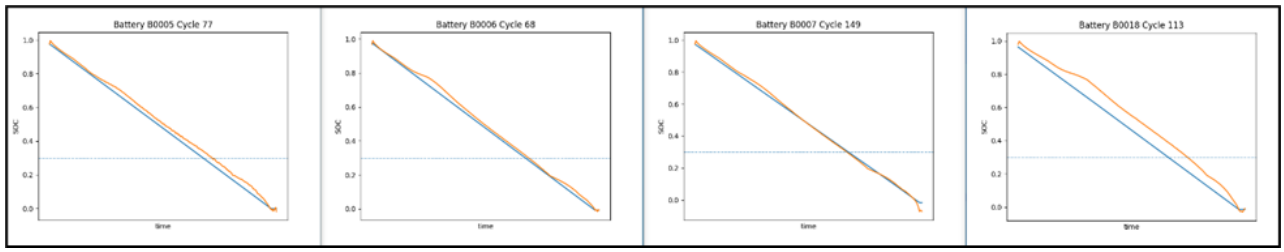



Figure 9. Prediction performance of SoC model

*Leftmost plot is for battery used for model development;*
*remaining 3 plots pertain to batteries excluded from the training data.*

# 7 Development and deployment workflow

The development workflow with elements of subsequent embedded target deployment of the battery health monitoring system based on SoC and SoH ANNs is an intricate, highly automated process relying on tight integration of the following aspects:

- Artificial Intelligence Development Tools: Training and preliminary validation of ANN-based components is conducted in TensorFlow (www.tensorflow.org) and resulting ANN implementations are generated in the Open Neural Network Exchange (ONNX) format (https://onnx.ai).
- System Architecture: Architecture and requirement specification development is performed in Cameo (www.3ds.com/products-services/catia/products) and this information is exported from Cameo for design, test and integration automation.

- Integration and Test Automation: ENSEMBLE Embedded Software integration Platform (ESiP) (Visnevski N. A., 2021; Visnevski, Hubscher-Younger, Rajhans, & Meng, 2020), developed by GE Research is used to process architecture specification from Cameo and automate integration of component implementation, test infrastructure, and embedded target deployment utilizing Simulink (www.mathworks.com/solutions/simulik.html), Deep Learning Blockset (www.mathworks.com/solutions/deep-learning.html), and Simulink Embedded Coder (www.mathworks.com/products/embedded-coder.html).
- Continuous Integration and Continuous Deployment (CI/CD): The workflow relies on Jenkins CI/CD automation server (www.jenkins.io).

The overall workflow is depicted in Figure 10. One of the key benefits of using Simulink-based implementation of the battery health monitoring system is the leverage of sophisticated embedded code generation capabilities of the Embedded Coder package. In fact, the deployment strategy chosen, relies heavily on the ability to automatically generate embedded C code with subsequent build and deployment of the target application on the embedded computing platform. To achieve this, we rely on Deep Learning Toolbox and its capability to bridge the gap between ANN implementations done in Python-based TensorFlow and embedded C code generation using ANN definitions in the ONNX format.

The structure of the Simulink model of the BHM is shown in Figure 11. The SoH and SoC ANNs are Deep Learning blockset components with direct imports of TensorFlow ANN models in ONNX format. Together with some signal pre- and post-conditioning and status evaluation logic, they form the core of the system implementation. ESiP processes this implementation shown in Figure 11 along with architecture specification from Cameo to produce an integrated system-level test harness for validation of the model. Test automation infrastructure in Simulink runs a series of test procedures to verify the implementation. At this point, if testing is successful, ESiP generates a deployment harness synthesizing the network communication layer for the deployable version of the battery health monitoring system. We use RTI Connext (RTI Connext DDS) middleware layer implementing Data Distribution Services (DDS) (Data Distribution Services Specification, n.d.) as middleware communication layer (Figure 12). Embedded Coder then generates C code and builds a fully encapsulated microservice-like deployment target application. Currently, we target ARM CPU architecture for deployment and have done preliminary tests on ARM Cortex-A on Raspberry Pi version 4.0 (www.raspberrypi.com).

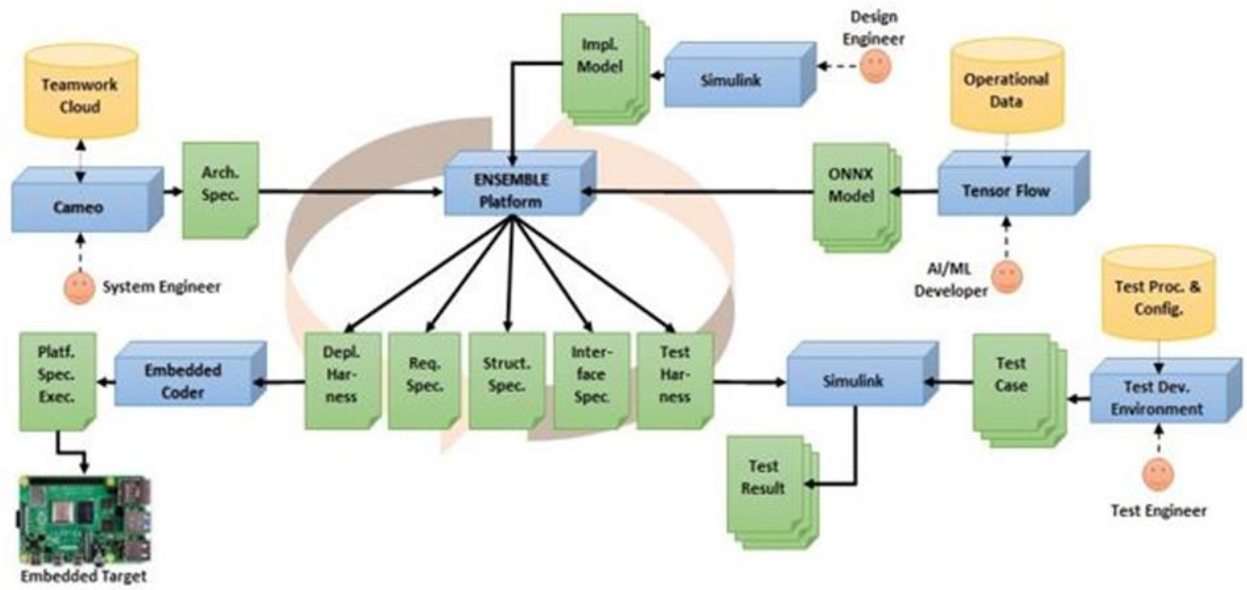



Figure 10. Development, testing, and deployment workflow

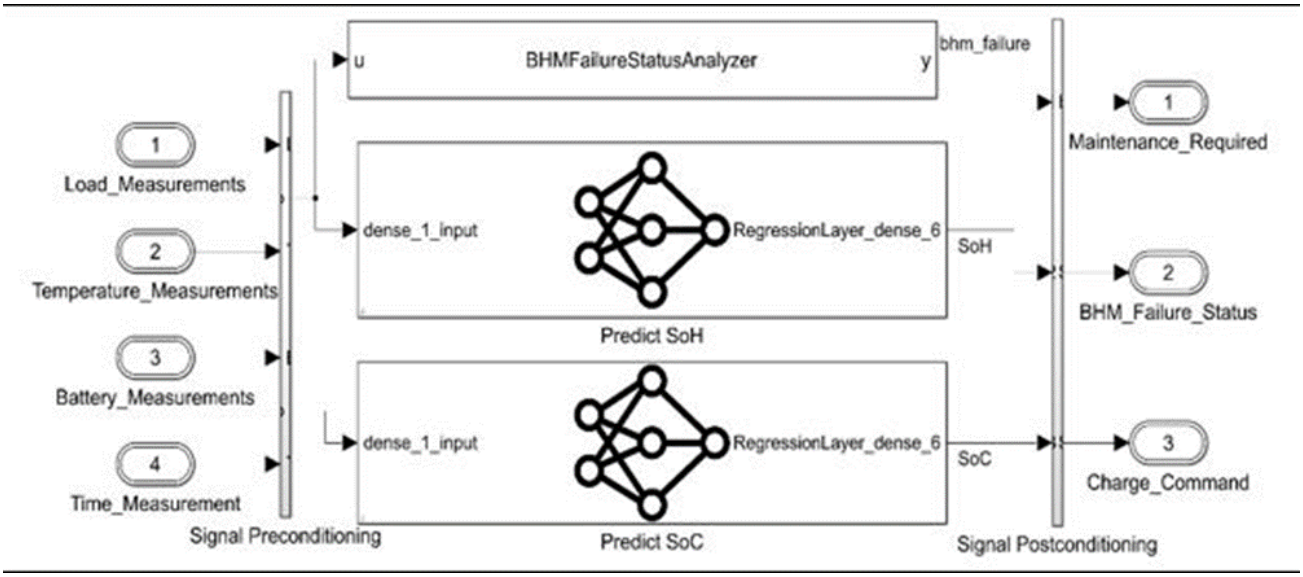



Figure 11. Core battery health monitoring system

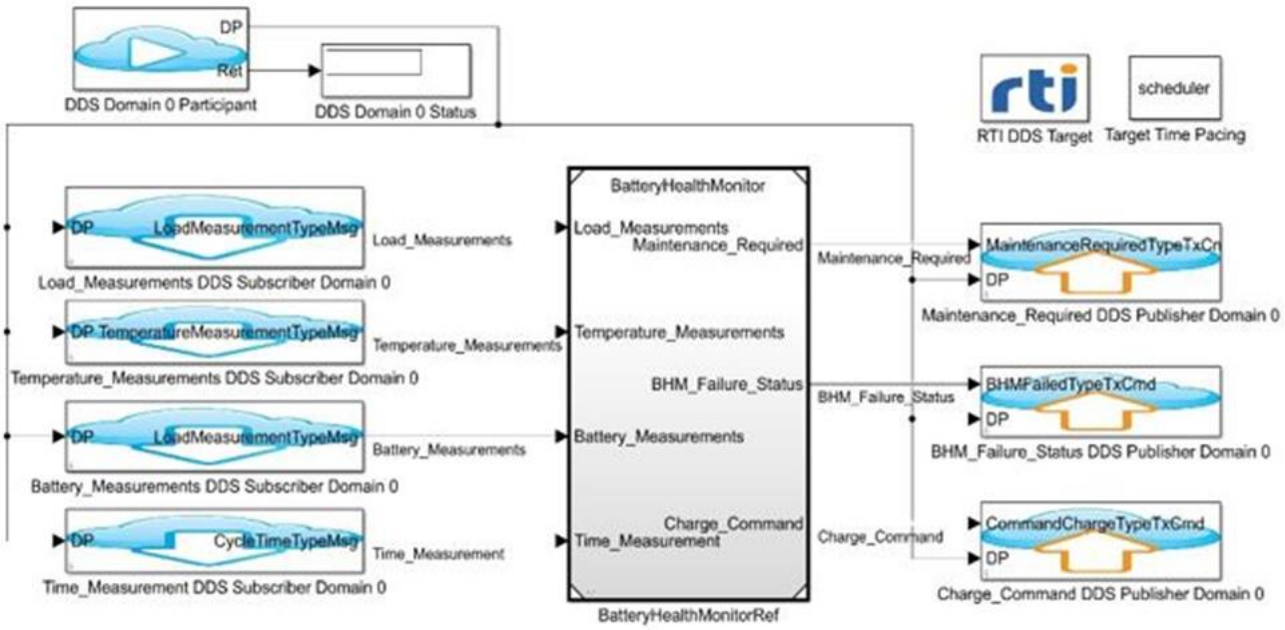



Figure 12. Integrated deployment harness

One of the critical advantages of this workflow is in its ability to comprehensively handle all aspects of development, integration, validation, and deployment of the system. It enables the path from ANN development in TensorFlow all the way down to embedded C code generation, eliminating some critical barriers and enabling possible paths to safety certification of implementations.

# 8 Overarching Properties (OPs) for ANN-based systems

## 8.1 Overarching Properties

The Overarching Properties have been described in (Holloway, 2019) as "a set of properties that are sufficient to establish the suitability of a product for installation on an aircraft". Three distinct property *statements* have been defined:

- *Intent*: The defined intended behavior is correct and complete with respect to the desired behavior,
- *Correctness*: The implementation is correct with respect to its defined intended behavior, under foreseeable operating conditions, and
- *Innocuity*: Any part of the implementation that is not required by the defined intended behavior has no unacceptable impact.

*Desired behavior* denotes the needs and constraints expressed by the stakeholders, *defined intended behavior* is the physical representation of the desired behavior (e.g., a set of requirements (Holloway, 2019)), *implementation* is the hardware or software element or combination of items for which approval is being sought, *foreseeable operating conditions* are the external and internal conditions in which the element to be evaluated is to be used, encompassing all known normal and abnormal conditions, and *unacceptable impact* is any impact that can lead to a direct or indirect undesirable effect on an aircraft or its components. To warrant approval for a hardware or software element, it is necessary to show with evidence that the element meets all three OPs. However, Holloway (2019) does not prescribe or imply any ordering between the three OPs.

Apart from the *statements* and *definitions* described above, the OP description comprises of a set of *requisites* that must exist to show the possession of OPs, a set of *assumptions* that must be stated for the demonstration of possession of OPs, and a set of *constraints* that restrict how demonstration of OP possession can be done (Holloway, 2019).

## 8.2 Overarching Properties for ANNs

We adopt a hybrid approach for certifying avionics software systems that contain ANN based sub-components. The hybrid approach, proposed in Daw et al. (2021) leverages traditional certification standards like ARP-4754A or DO-178C for parts of the systems that are supported by the standards, and OPs for parts that are not supported by the standards. In our adaptation of the hybrid approach, we consider an ANN-based sub-component (called a **NN-SOFTWARECOMPONENT**) as an entity that is certified using OPs. The schematics in Figure 13 depict our certification strategy in which only the ANN-based sub-components of the RIPS—the **SOH-NN-SW-COMP** for SoH prediction and the **SOC-NN-SW-COMP** for SoC prediction—are certified using OPs.

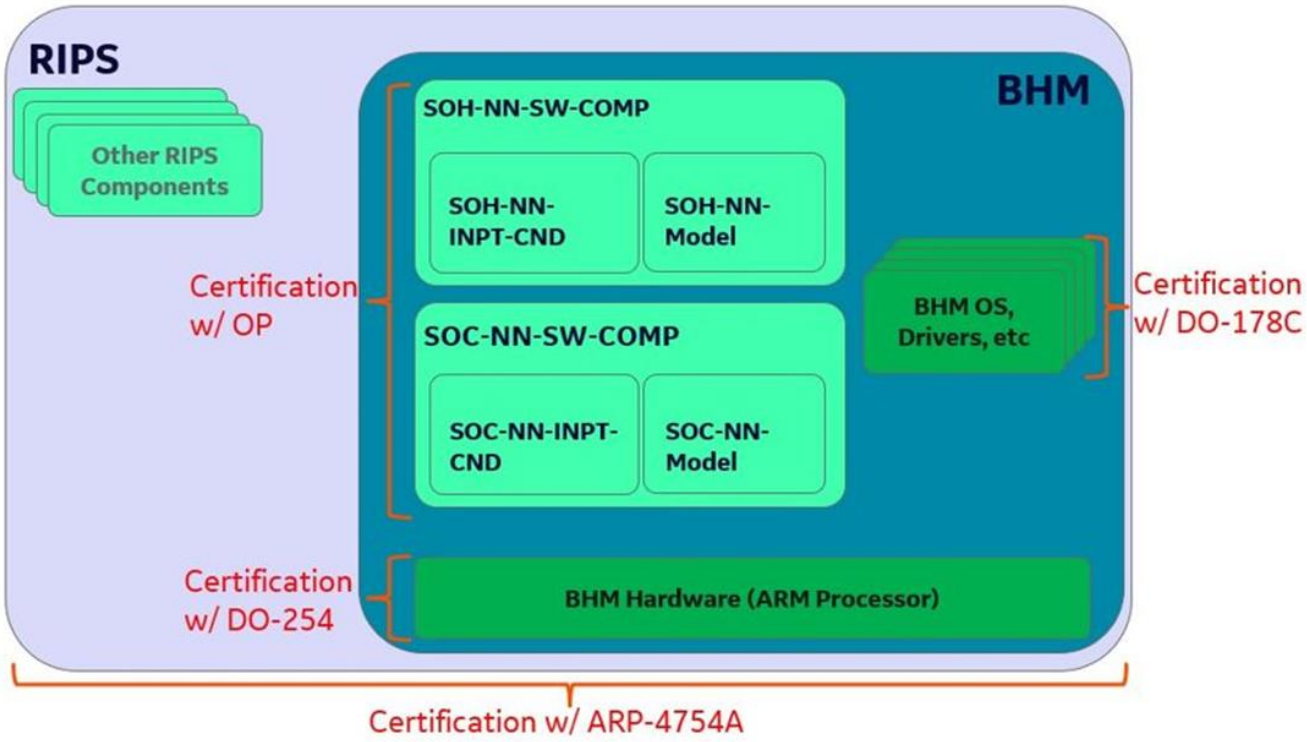



Figure 13. Hybrid certification strategy for RIPS system

A **NN-SOFTWARE-COMPONENT** is composed of an ANN **NN-MODEL** and an input conditioner **INPUT-C ONDITIONER** that can prevent off-nominal input values from being sent to the **NN-MODEL**. The defined intended behavior for the **NN-SOFTWARE-COMPONENT** is captured as a set of high-level requirements **NN-REQUIREMENTS**. We explicitly define an activity **MODEL-DEVELOPMENT-ACTIVITY** for model development which is involved with developing the **NN-MODEL** using a given dataset **MODEL-DEVELOPMENT-DATA** that can be split by the ML engineers into testing and training sets as needed. In addition, we define a logically separate activity **SOFTWARE-QUALIFICATION-ACTIVITY** for the atomic qualification/verification of the **NN-SOFTWARE-COMPONENT** using a given dataset **SOFTWARE- QUALIFICATION-DATA**. We also define a sub-component-level safety assessment **NN-SAFETY-ASSESSMENT** for the **NN-SOFTWARE-COMPONENT** aimed a identifying the set of aberrant behaviors **NN-FAILU RE-MODE** of the **NN-SOFTWARE-COMPONENT** that can directly or indirectly contribute to system-level hazards. A glossary of these terms is provided in Figure 14.

- **NN-SOFTWARE-COMPONENT**: A software component that comprises of a single **NN-MODEL** and an **INPUT-CONDITIONER**.
- **NN-MODEL**: The final pruned NN model developed by the ML engineers.
- **INPUT-CONDITIONER**: A software that is used to condition values before they are sent as inputs to a **NN-MODEL**. Input conditioning here can be filtering or other techniques (e.g., imputation, clipping, and runtime assurance) to address off-nominal values with respect to the **MODEL-DEVELOPMENT-DATA**.
- **NN-REQUIREMENTS**: Requirements developed for the **NN-SOFTWARE-COMPONENT** to concretely represent the desired behavior. These are the Defined Intended Behavior.
- **MODEL-DEVELOPMENT-ACTIVITY**: The activity of designing and developing the **NN-MODEL**.
- **MODEL-DEVELOPMENT-DATA**: The data available/provided to the ML engineers for the **MODEL-DEVELOPMENT-ACTI VITY**.
- **SOFTWARE-QUALIFICATION-ACTIVITY**: The activity of qualifying the **NN-SOFTWARE-COMPONENT** with respect to the **NN-REQUIREMENTS**.
- **SOFTWARE-QUALIFICATION-DATA**: The data used in the **SOFTWARE-QUALIFICATION-ACTIVITY**.
- **QUANTITY-OF-INTEREST**: The parameter analyzed by the **NN-MODEL**.
- **OFF-NOMINAL-INPUT-VALUE**: Any input vector meant for the **NN-MODEL** that is abnormal with respect to the **MO DEL-DEVELOPMENT-DATA**.
- **STAKEHOLDER-CONSTRAINTS**: Constraints that constitute the desired behavior of the **NN-SOFTWARE-COMPONENT**.
  - **DATA-DEPENDENT-CONSTRAINTS**: Constraints that are dependent on data. Eg: prediction performance characteristics like accuracy, robustness, stability, etc.
  - **DATA-INDEPENDENT-CONSTRAINTS**: Constraints that are independent of data. Eg: response time, size, etc.
- **NN-FAILURE-MODE**: A behavior of the **NN-SOFTWARE-COMPONENT** that can contribute to a system-level hazard identified in the system-level safety assessment.
- **NN-FAILURE-MODES**: The set of all identified instances of **NN-FAILURE-MODE** for a given **NN-SOFTWARE-COMPONENT**.
- **NN-SAFETY-ASSESSMENT**: The activity of identifying the **NN-FAILURE-MODES** for a **NN-SOFTWARE-COMPONENT** and their contributions to the system-level hazards.
- **UNACCEPTABLE-IMPACT**: Any impact of the **NN-SOFTWARE-COMPONENT** that can compromise the **NN-SAFETY-ASSE SSMENT**.

Figure 14. Glossary of definitions used in our OP arguments for ANN-based sub-components

Our primary goal was to develop a methodology to demonstrate OP possession for a **NN-SOF TWARE-COMPONENT**. In this endeavor, we worked with two objectives in mind:

1. unambiguously identifying the conditions under which Intent, Correctness, and Innocuity can be claimed for **NN-SOFTWARE-COMPONENT** by focusing on "*what* conditions are needed?" and not being limited by the complexity involved in meeting the conditions; and

2. developing generic arguments to show OP possession so that the arguments can be reused, either fully or partially, in the context of ANN-based systems other than our use case.

For this, we took inspiration from the Friendly Argument Notation (FAN) (Holloway, The Friendly Argument Notation (FAN), 2020) based argument structures presented in Daw et al. (2021) and Wasson & Holloway (2022). In the rest of this section, we will use the argument structure shown in Figure 15 which consists of—a *block* making a claim that the audience must accept as true (**Believing:**); a block that provides a reasoning behind why the audience should believe the claim (**Is Justified by applying:**); and a block of a set of *premises*[1] that can support the reasoning (**To these premises:**). An argument is a valid argument if all premises are shown to be true with evidence. Any **HIGHLIGHTED-WORD** in an argument has a well-defined context-specific definition provided in Figure 14.

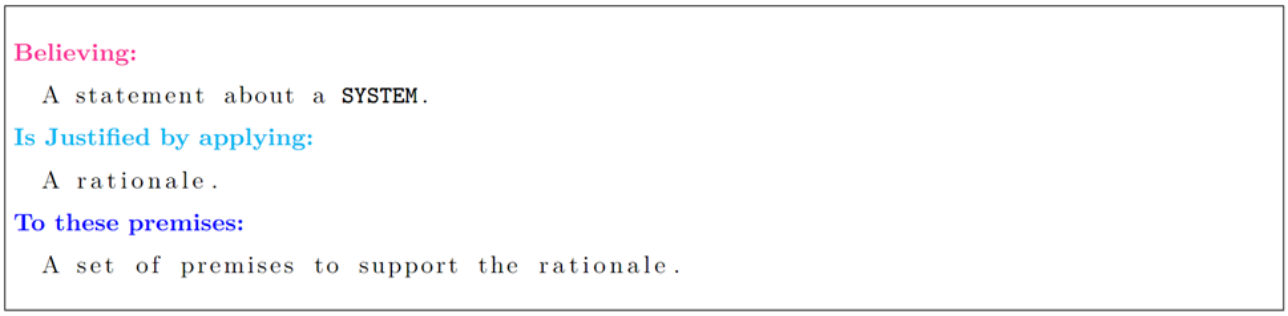



Figure 15. Example argument structure used in this paper

Figure 16 shows our argument structure for claiming that **NN-SOFTWARE-COMPONENT** possesses Intent. The highest level of the argument is a belief that **NN-SOFTWARE-COMPONENT** holds Intent and, in the justification, the statement of Intent has been customized to state that the **NN-REQUIREMENTS**, which constitute the defined intended behavior for the **NN-SOFTWARE-COMPONENT**, are correct and complete with respect to the desired behavior. To argue for possession of Intent, we believe that it is necessary to show the following —the defined intended behavior semantically captures the desired behavior, the defined intended behavior is verifiable, and the usage of the defined intended behavior will appropriately consider off-nominal and incorrect/adversarial input values. Therefore, we present four premises to support the Intent argument. Premise **A1** ensures that the **NN-REQUIREMENTS** correctly address all the stakeholder

[1] A premise is a statement that is either true or false.

constraints that are necessary for the desired behavior of the NN-SOFTWARE-COMPONENT and Premise **A2** ensures that there is a way to verify that the NN-REQUIREMENTS are satisfied. However, satisfactory performance of NN-SOFTWARE-COMPONENT against a qualification dataset does not necessarily guarantee that the software is safe. There could be malicious inputs in the model development dataset that could have been used for training the NN-MODEL that may not be detected during software qualification. For this reason, we introduce premise **A3** to ensure that the MODEL-DEVELOPMENT-DATA will be free from such inputs. Moreover, sparsity of the MODEL-DEVELOPMENT-DATA may cause the model to be error-prone for off-nominal inputs. To address that, we add premise **A4** to ensure that such inputs to the NN-MODEL will be addressed correctly.

**Believing:**

The NN-SOFTWARE-COMPONENT holds Intent.

**Is Justified by applying:**

The NN-REQUIREMENTS are correct and complete with respect to the Desired Behavior.

**To these premises:**

**A1**. The NN-REQUIREMENTS correctly address all DATA-DEPENDENT-CONSTRAINTS and DATA-INDEPENDENT-CONSTRAINTS desired by the stakeholders.

**A2**. The DATA-DEPENDENT-CONSTRAINTS in the NN-REQUIREMENTS are verifiable using the SOFTWARE-QUALIFICATION-DATA.

**A3**. The MODEL-DEVELOPMENT-DATA is an accurate representation of the QUANTITY-OF-INTEREST throughout the foreseeable operating conditions.

**A4**. The NN-SOFTWARE-COMPONENT addresses any off-nominal input value to the NN-MODEL.

Figure 16. Our argument that an ANN-based sub-component possesses Intent

The argument structure for claiming that a NN-SOFTWARE-COMPONENT possesses Correctness is presented in Figure 17. Similar to the Intent argument, the belief is that the NN-SOFTWARE-COMPONENT holds Correctness and the justification is that the NN-SOFTWARE-COMPONENT is correct with respect to the NN-REQUIREMENTS, under the SOFTWARE-QUALIFICATION-DATA (which represents the foreseeable operating conditions). The argument for Correctness is supported by intuitive premises that support the claim by ensuring that the ANN-based software component's implementation correctly satisfies the requirements. The first premise **B1** trivially ensures that NN-MODEL is developed using the intended development dataset which is analogous to the low-level requirements. The second premise **B2** ensures that the SOFTWARE-QUALIFICATION-DATA is an appropriate proxy for the foreseeable operating conditions. Premises **B2** and **B3** then ensure

that the NN-SOFTWARE-COMPONENT follows all constraints set in the requirements. Finally, premise **B4** ensures that the implementation correctly handles all off-nominal inputs.

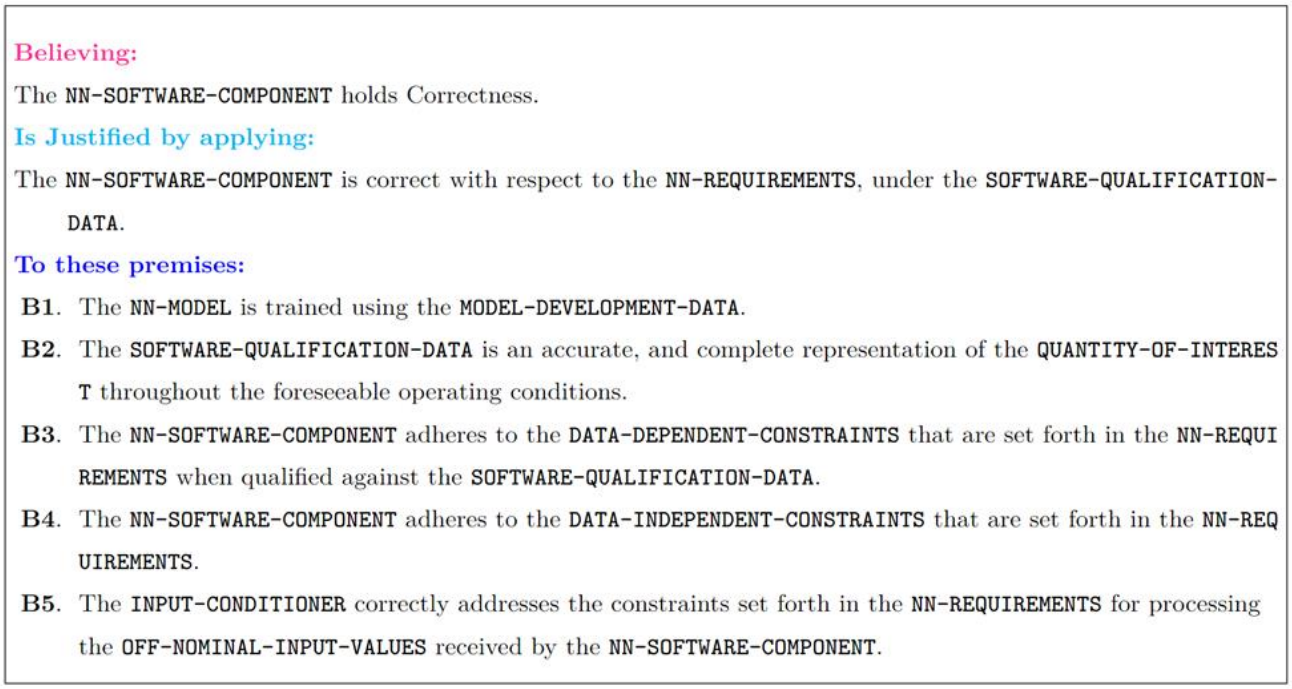



Figure 17. Our argument for an ANN-based sub-component possessing Correctness

Figure 18 shows our argument structure for claiming that a **NN-SOFTWARE-COMPONENT** possesses Innocuity. The justification for this argument is that any part of the **NN-SOFTWARE-COMPONENT** that is not required by the **NN-REQUIREMENTS** has no **UNACCEPTABLE-IMPACT**, where an **UNACCEPTABLE-IMPACT** is any impact that can violate the **NN-SAFETY-ASSESSMENT** and consequently, the system-level safety assessment. However, contrary to traditional software systems, ANNs are difficult to interpret, and it is usually challenging to identify the "parts" of an ANN, such as the predictors that are important for the property being modeled (Zhang, et al., 2018). For this reason, it is difficult to explicitly identify the "parts" of a **NN-SOFTWARE-COMPONENT** that are not directly warranted by the requirements. This creates a hurdle towards a direct argument for claiming the possession of Innocuity. Therefore, we have identified a set of premises to develop an argument such that, much like a *proof by contradiction* (Arkoudas & Musser, 2017) approach, if the premises hold, then it cannot be the case that the claim will not hold. Premise **C1** ensures that any potential **NN-FAILURE-MODE** that can lead to an **UNACCEPTABLE-IMPACT** will be identified in the **NN-SAFETY-ASSESSMENT** and premise **C2** states that the effect of all failure modes in the identified set **NN-FAILURE-MODES** will be mitigated by the **NN-REQUIREMENTS**. Therefore, if the **NN-R EQUIREMENTS** are satisfied by the implementation

(Correctness), then no unacceptable impact can occur and, by contradiction, there cannot be any "part" of the NN-SOFTWARE-COMPONENT that can have an unacceptable impact.

We can see that our premises for Innocuity ensure that *if* a NN-SOFTWARE-COMPONENT possesses Correctness and the premises **C1** and **C2** are true, then the NN-SOFTWARE-COMPONENT will also possess Innocuity. We believe that in the absence of a principled approach that can allow for parts of a NN-SOFTWARE-COMPONENT to be explicitly identified, this approach is an ad-hoc solution for ensuring that the NN-SOFTWARE-COMPONENT will possess Innocuity. Since the possession of all three properties is necessary for a product to warrant approval (Holloway, 2019), it is not strictly necessary to be able to show the possession of each property independently.

**Believing:**

The NN-SOFTWARE-COMPONENT holds Innocuity.

**Is Justified by applying:**

Any part of the NN-SOFTWARE-COMPONENT that is not required by the NN-REQUIREMENTS has no UNACCEPTABLE-IMPACT.

**To these premises:**

**C1**. Any potential NN-FAILURE-MODE is correctly identified in a NN-SAFETY-ASSESSMENT conducted for the NN-SOFTWARE-COMPONENT.

**C2**. The NN-REQUIREMENTS mitigate the effect of the identified NN-FAILURE-MODES.

Figure 18. Our argument for an ANN-based sub-component possessing Innocuity

In this subsection, we have identified premises that we believe are necessary and sufficient for arguing that a NN-SOFTWARE-COMPONENT will possess the OPs. We have not discussed the implications of the premises on the development and verification process and how appropriate evidence may be generated to support them. Next, we will use these arguments for one of the NN-SOFTWARE-COMPONENT instances in the RIPS system and present some DAL-appropriate strategies for generating evidences for the premises.

## 8.3 Instantiation of our OP arguments for the RIPS

Let SOC-NN-SW-COMP comprise of the model **SOC-NN-MODEL** and the **SOC-NN-INPT-CND** input conditioner, SOC-NN-REQS be the requirements on **SOC-NN-SW-COMP**, **SOC-NN-DEV-ACT** be the activity for developing SOC-NN-MODEL using **SOC-NN-DEV-DATA**, **SOC-NN-QUAL-ACT** be the activity for qualifying **SOC -NN-SW-COMP** using **SOC-NN-QUAL-DATA**, **RIPS-SOC-CONSTRAINTS** be the relevant constraints for the **SOCNN-SW-**

**COMP**, **SOC-NN-FMS** be the set of failure modes **SOC-NN-FM for SOC-NN-SW-COMP** identified in the component-level safety assessment **SOC-NN-SA**.

Believing:

The SOC-NN-SW-COMP holds Intent.

Is Justified by applying:

The SOC-NN-REQS are correct and complete with respect to the RIPS desired behavior.

To these premises:

**A1**. The SOC-NN-REQS correctly address all RIPS-SOC-CONSTRAINTS.

**A2**. The RIPS-SOC-CONSTRAINTS in the SOC-NN-REQS are verifiable using the SOC-NN-QUAL-DATA.

**A4**. The SOC-NN-SW-COMP addresses any off-nominal input value to the SOC-NN-MODEL.

Figure 19. Argument for RIPS SOC-NN-SW-COMP possessing Intent

Since the RIPS is a DAL D application, we instantiate the argument structures for Intent, Correctness, and Innocuity for the **SOC-NN-SW-COMP** and present DAL D appropriate strategies for each premise in the arguments. For the development of traditional DAL D software, it is sufficient to specify Intent through the development and validation of requirements, Correctness through the verification of those requirements on the implementation, and Innocuity through a safety assessment of the software component's contributions to system-level hazards. Given the DAL of the RIPS application, we only include the premises that we believed were appropriate. As we evaluated strategies to satisfy our proposed premises that were equivalent to this DAL D perspective, we limited those strategies for what could be performed treating the ANNs as black boxes and used existing standards as inspiration to remain grounded to what is traditionally done in the industry.

The Intent argument for **SOC-NN-SW-COMP** is given in Figure 19. Since for traditional DAL D software it is usually not necessary to review the quality of the low-level requirements, and the model development data is roughly analogous to low-level requirements for ANNs, we believe that premise **A3** is not applicable for our RIPS use case. Potential strategies for the applicable premises are:

- **A1**: A reviewer will review the **SOC-NN-REQS** to ensure that they address all the **RIPS-SOC-CONSTRAINTS**. This is roughly analogous to reviewing the high-level requirements for completeness and correctness in DO-178C Table A2 (Objectives 1 and 2) and Table A3 (Objectives 1, 2, and 6).

- **A2**: A reviewer will review the appropriate planning documentation that outlines the qualification plan for the RIPS. This is roughly analogous to reviewing the Software Verification Plan for DAL D in DO-178C Table A1 (Objective 1), Table A9, and Table A10.
- **A4**: A reviewer will review the RIPS requirements and design documents to ensure that a mechanism exists to address off-nominal input values to the **SOC-NN-SW-COMP**. This is roughly analogous to reviewing the design descriptions and the software requirements data for DAL D software in DO-178C Table A2 (Objectives 1, 2, and 3).

**Believing:**

The SOC-NN-SW-COMP holds Correctness.

**Is Justified by applying:**

The SOC-NN-SW-COMP is correct with respect to the SOC-NN-REQS, under the SOC-NN-QUAL-DATA.

**To these premises:**

**B2**. The SOC-NN-QUAL-DATA is an accurate and complete representation of the SoC throughout the foreseeable operating conditions.

**B3**. The SOC-NN-SW-COMP adheres to the Data-Dependent RIPS-SOC-CONSTRAINTS that are set forth in the SOC-NN-REQS when qualified against the SOC-NN-QUAL-DATA.

**B4**. The SOC-NN-SW-COMP adheres to the Data-Independent RIPS-SOC-CONSTRAINTS that are set forth in the SOC-NN-REQS.

**B5**. The SOC-NN-INPT-CND correctly addresses the constraints set forth in the SOC-NN-REQS for processing the Off-Nominal-Input-Values that are then sent to the SOC-NN-SW-COMP.

Figure 20. Argument for RIPS SOC-NN-SW-COMP possessing Correctness

The argument for **SOC-NN-SW-COMP** Correctness is shown in Figure 20. Here, premise **B1** is not applicable because traditionally, for DAL D, it is not necessary to verify the implementation's compliance to the low-level requirements. The strategies for the applicable premises are:

- **B2**: A reviewer will review that the **SOC-NN-QUAL-DATA** adequately represents the SoC under the foreseeable operating conditions of the RIPS. The qualification data is used to verify the software component and it is necessary to review it for any DAL.
- **B3** and **B4**: Software verification results against the **SOC-NN-REQS** (using the **SOC-NN-QUAL-D ATA** for data-dependent constraints) can be used. This is roughly analogous to the code

compliance and test coverage of high-level requirements for traditional DAL D software prescribed in DO-178C Table A6 (Objective 1) and Table A7 (Objective 3).

- **B5**: Robustness testing results using the SOC-NN-QUAL-DATA can be used. This is inspired by the recommendation in DO-178C Table A6 (Objective 2).

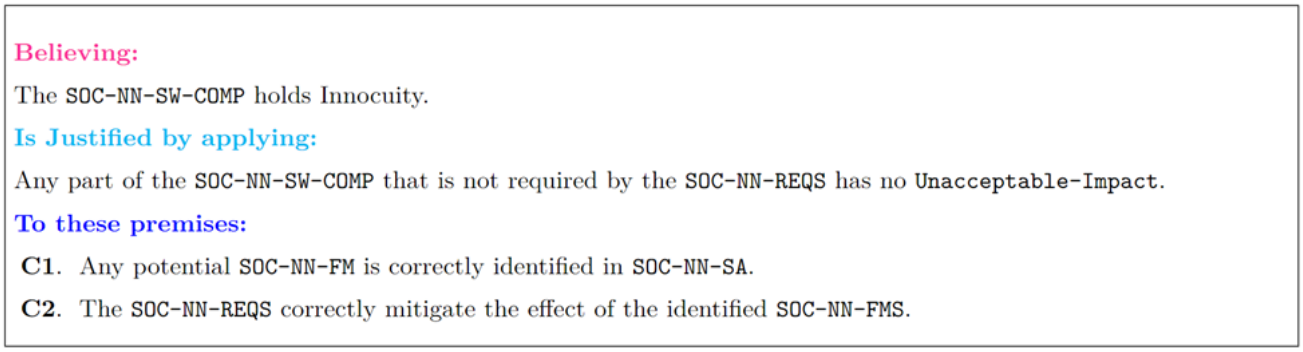



Figure 21. Argument for RIPS SOC-NN-SW-COMP possessing Innocuity

Finally, Figure 21 shows the Innocuity argument for SOC-NN-SW-COMP. In this case, all premises are applicable, and the strategies are as follows:

- **C1**: A reviewer will review the component-level safety assessment and confirm that that all component-level failure modes have been correctly identified. This is inspired by ARP-4754A Table A (Objectives 3.1, 3.3, and 3.6) which prescribes an initial system safety assessment and a post-development analysis of the implementation to identify the contribution of different components to system safety.
- **C2**: A reviewer will review that the SOC-NN-REQS mitigate the effects of all identified SOC-NN-FM. This is inspired by ARP-4754A Table A (Objective 3.6) which prescribes that high-level requirements must mitigate all system hazards (by mitigating component-level failure conditions contributing to system hazards).

# 9 Assurance cases

The text-based argument structures we presented earlier in this paper can be converted into other industry-accepted formats such as the Goal Structuring Notation (GSN) (Kelly & Weaver, 2004). In this section, we present some GSN fragments corresponding to our arguments using a colored augmentation of the GSN format presented in (Meng, Paul, Moitra, Siu, & Durling, 2021).

Figure 22, Figure 23, and Figure 24 show the GSN assurance case fragments corresponding to the Intent, Correctness, and Innocuity arguments presented earlier for the SOC-NN-SW-COMP. The top-level goals in the GSN fragments represent the top-level claims about OP possession in the arguments. These claims are supported by lower-level sub-goals which are the premises in the corresponding arguments. There are unique strategies for each of the top-level goals and the sub-goals that connect the hierarchy of goals in each argument structure. Red goals with diamonds under them in the figures are incomplete goals for which we have not generated evidence for in this work.

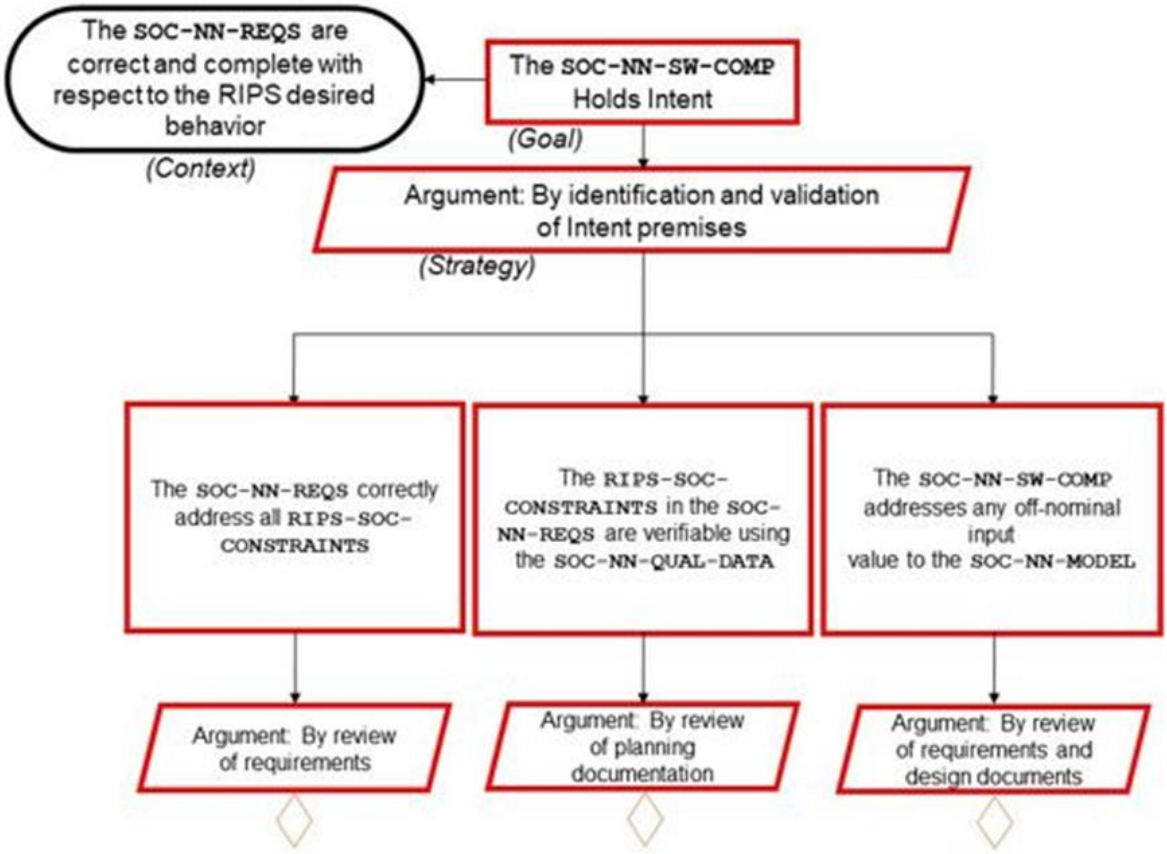



Figure 22. Assurance case fragment (SOC-NN-SW-COMP holds Intent)

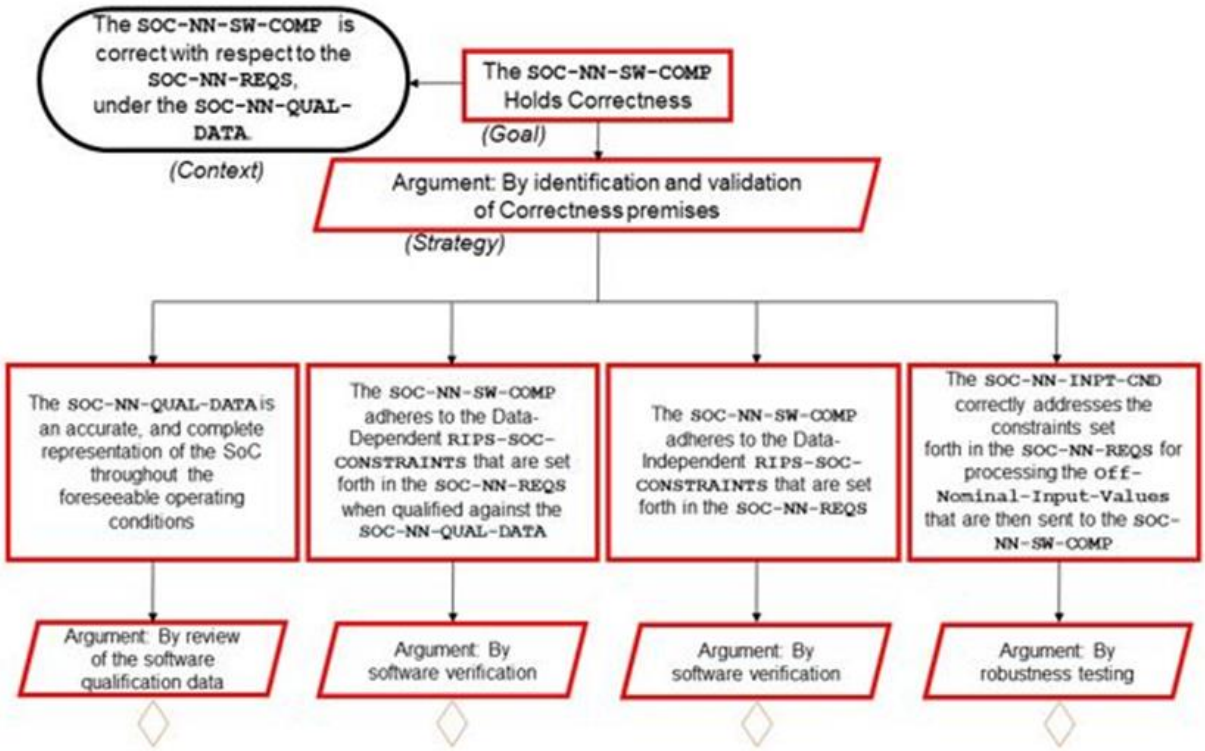



Figure 23. Assurance case fragment (SOC-NN-SW-COMP holds Correctness)

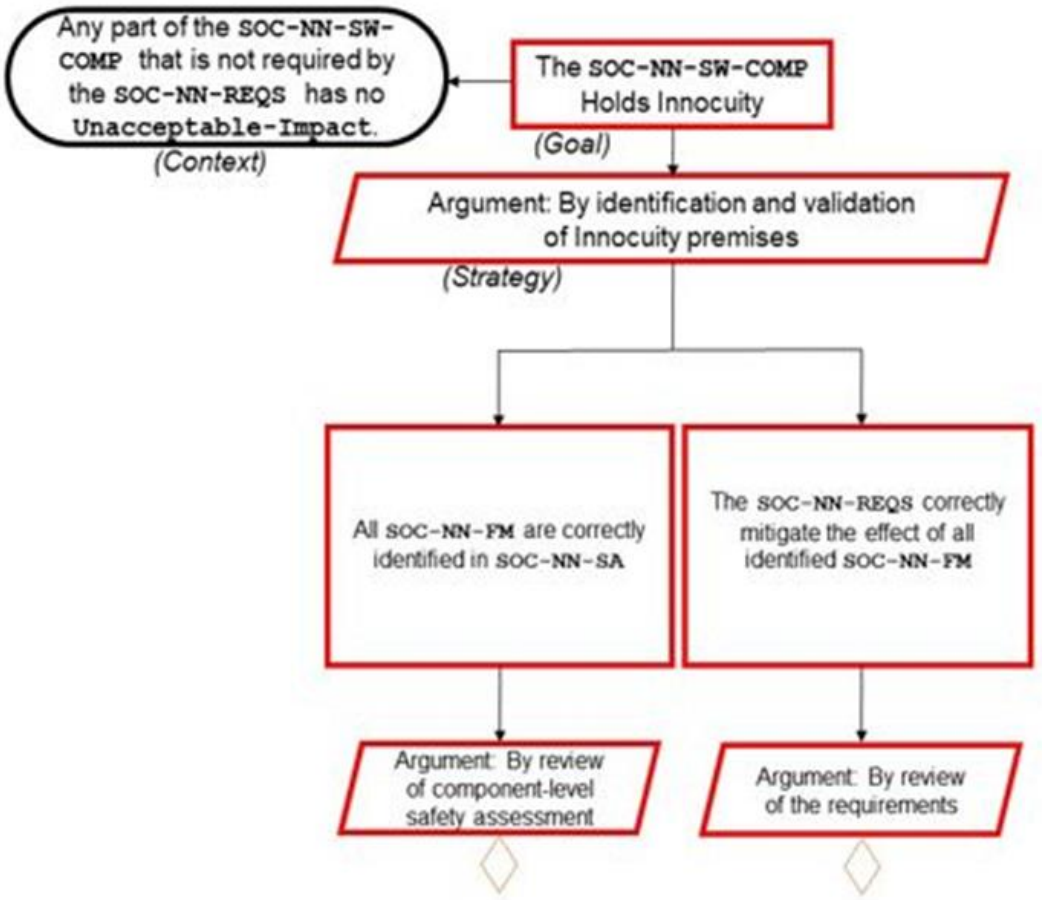



Figure 24. Assurance case fragment (SOC-NN-SW-COMP holds Innocuity)

# 10 Formalization of artifacts for curation

We have formalized the artifacts generated for this project based on a concept presented in Valapil et al. (2021) so that it can be ingested into our research-grade database called the Rapid Assurance Curation Kit (RACK) which is available as open-source (GE High Assurance GitHub Repository, 2022). RACK (Moitra, et al., 2022) is a novel data curation platform being developed under the DARPA Automated Rapid Certification of Software (ARCOS) program. We are developing RACK as a scalable way to curate diverse types of certification evidence such as test cases and test results, analytic results from formal methods tools, the structure of design artifacts, and relevant requirements. Behind the scenes, RACK is a triple store with a schema (which we often call a data model) optimized for curation of certification evidence. The foundation of our data model is the well-known entity-relationship (E-R) model. The E-R model is comprised of entity classes and relationship classes, where entity classes represent real-world concepts or objects, and relationship classes represent the connections between those concepts or

objects. In our use of that model, we restrict attributes to be associated only with entity instances. We allow the usual cardinalities on relationship instances.

RACK allows data models to be created using Semantic Application Design Language (SADL) (2022), which is a controlled-English grammar that expresses Web Ontology Language (OWL) ontologies plus rules. SADL expressivity is equivalent to OWL 1 plus qualified cardinality from OWL 2. SADL is also an integrated development environment (IDE), available as a set of Eclipse plugins or as a set of services providing a web browser-based interface for creating, testing, and maintaining semantic models. The grammar includes constructs for querying, testing, and modeling lifecycle management. SADL is open source and has been used in a wide variety of applications. The team has previously extended the SADL grammar to create the SADL Requirements Language (SRL), which enables capturing requirements in controlled English.

RACK consists of a core ontology consisting of the classes THING, ENTITY, COLLECTION, AGENT, and ACTIVITY, that can be extended with project-specific concepts and properties that can be used to formalize project-specific data and ingest it into RACK. To enable formalizing airborne systems, their development process, and their certification process, we have created classes called FILE, FUNCTION, HWCOMPONENT, INTERFACE, SWCOMPONENT, SYSTEM, ANALYSIS, HAZARD, REVIEW, REQUIREMENT, TEST, and OBJECTIVE that are sub-types of ENTITY.

## 10.1 SADL formalization of the FHA

To express functional hazard assessment for the RIPS system, we have designed a data model that is built upon RACK's core ontology. In the data model, we have introduced classes such as Severity, DesignAssuranceLevel, VerificationMethod, and Phase that can take predetermined enumerated values that constrain the possible instances to values that are appropriate as per the official standards as follows:

```
Severity (note "Severity Types taken from FAA Circular AC No: 23.1309-1E") is a type of THING,
      must be one of {Negligible (note "No safety effect: failure conditions that would not affect the operational capability of the airplane or increase crew workload"),
                      Minor (note "Failure conditions that would not significantly reduce airplane safety and involve crew actions that are within their capabilities."),
                      Major (note "Failure conditions that would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions to the extent that there would be a significant reduction in safety margins or functional capabilities."),
                      Hazardous (note "Failure conditions that would reduce the capability of the airplane or the ability of the crew to cope with adverse operating conditions"),
                      Catastrophic (note "Failure conditions that are expected to result in multiple fatalities of the occupants, or incapacitation or fatal injury to a flight crewmember normally with the loss of the airplane.")}.

 DesignAssuranceLevel (note "The minimum Design Assurance Level") is a type of THING
      must be one of {LevelA (note "Level A"),  // Cannot use 'A' since it is a keyword in SADL, so using LevelA instead
                      LevelB (note "Level B"),
                      LevelC (note "Level C"),
                      LevelD (note "Level D")}.

 VerificationMethod (note "The method used for verifying a component") is a type of ACTIVITY
      must be one of {FHA (note "Functional Hazard Assessment"),
                      FTA (note "Fault Tree Analysis"),
                      DD (note "Dependency Diagram"),
                      MA (note "Markov Analysis"),
                      FMEA (note "Failure Modes and Effects Analysis"),
                      FMES (note "Failure Modes and Effects Summary"),
                      ZSA (note "Zonal Safety Analysis"),
                      CMA (note "Common Mode Analysis"),
                      PRA (note "Particular Risk Analysis")}.

 Phase (note "The flight phase") is a type of THING
```

```
      must be one of {ALL (note "All phases"),
                          STD (note "Standing"),
                          PBT (note "Pushback/Towing"),
                          TXI (note "Taxi"),
                          TOF (note "TakeOff"),
                          ICL (note "Initial climb"),
                          ENR (note "En-route (Cruise)"),
                          APR (note "Approach"),
                          LDG (note "Landing")}.
```

## 10.2 SADL formalization of the RIPS requirements

The RACK core ontology provides a REQUIREMENT class that has some basic properties that can be used to encode the information present in our RIPS requirements as follows:

```
REQUIREMENT
      (note "Captures (both high- and low-level) properties of a
process or artifact that are to be assessed")
      is a type of ENTITY.

      governs (note "ENTITY(s) that are the subject of the
requirement") describes REQUIREMENT with values of type ENTITY.
      governs is a type of wasImpactedBy.

      satisfies (note "Parent ENTITY(s) (e.g. REQUIREMENT) that this
REQUIREMENT is derived from") describes REQUIREMENT with values of
type ENTITY.
      satisfies is a type of wasImpactedBy.

      Rq:mitigates (note "ENTITY(s) (e.g. HAZARD) that is being
mitigated by this REQUIREMENT") describes REQUIREMENT with values of
type ENTITY.
      Rq:mitigates is a type of wasImpactedBy.

      wasGeneratedBy of REQUIREMENT only has values of type
REQUIREMENT_DEVELOPMENT.
```

To express our project-specific properties like Requirement Type, Architecture Allocation, Source, etc., we introduce new properties to the REQUIREMENT class of RACK Core as follows:

```
architectureAllocation describes REQUIREMENT with values of type THING.

Rq:derivedRequirementIndicator (note "Use True for derived requirement and false otherwise") describes REQUIREMENT with values of type boolean.

rationale describes REQUIREMENT with values of type string.

requirementType describes REQUIREMENT with a single value of type RequirementType.

correctness describes REQUIREMENT with values of type Correctness.

correctnessFailComments describes REQUIREMENT with values of type string.

completetness describes REQUIREMENT with values of type Completeness.

completenessFailComments describes REQUIREMENT with values of type string.

verificationRationale describes REQUIREMENT with a single value of type string.

Rq:verificationMethod describes REQUIREMENT with a single value of type VerificationMethod.
```

To separate the different levels of requirements in our systems design, we create three sub-classes of REQUIREMENT class to encode three different levels – System, High Level, and Low Level as follows:

```
SystemLevelRequirement (note "System Level Requirement") is a type of
REQUIREMENT
      described by rd:satisfiedBy with values of type string
      described by rd:source with values of type string.

SoftwareHighLevelRequirement (note "Software High Level Requirement")
is a type of REQUIREMENT.
SoftwareLowLevelRequirement (note "Software Low Level Requirement") is
a type of REQUIREMENT.
```

To allow users to instantiate some properties with values from a pre-fixed set, we can also create enumerated classes like the one shown below for Requirement Type as follows:

```
RequirementType (note "The type of requirements") is a type of THING
      must be one of {functionalRequirement,
                              designConstraint,
                              interfaceRequirement,
                              safetyRequirement,
                              performanceRequirement,
                              usabilityRequirement,
                              physicalRequirement
                              }.
```

All SADL files have been provided in Appendix C for reference.

## 10.3 Traceability of artifacts in RACK

Using RACK's query interface, it is possible to see the ontological traceability between the artifacts ingested into RACK.

**Requirements to requirements**

Shown below in Figure 25 is the RACK query (and partial result) that shows the traceability between requirements using the "satisfies" property of the REQUIREMENT class.

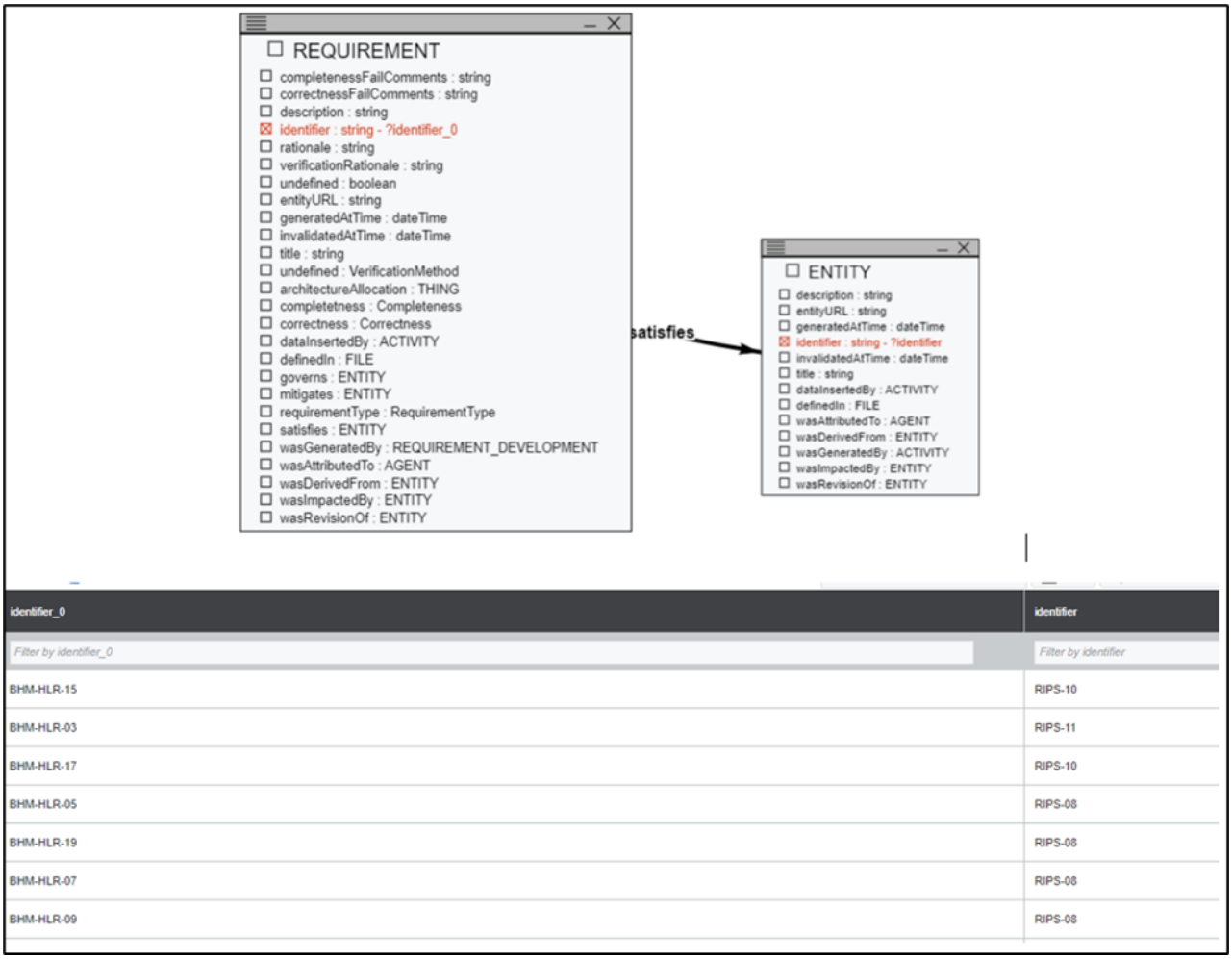
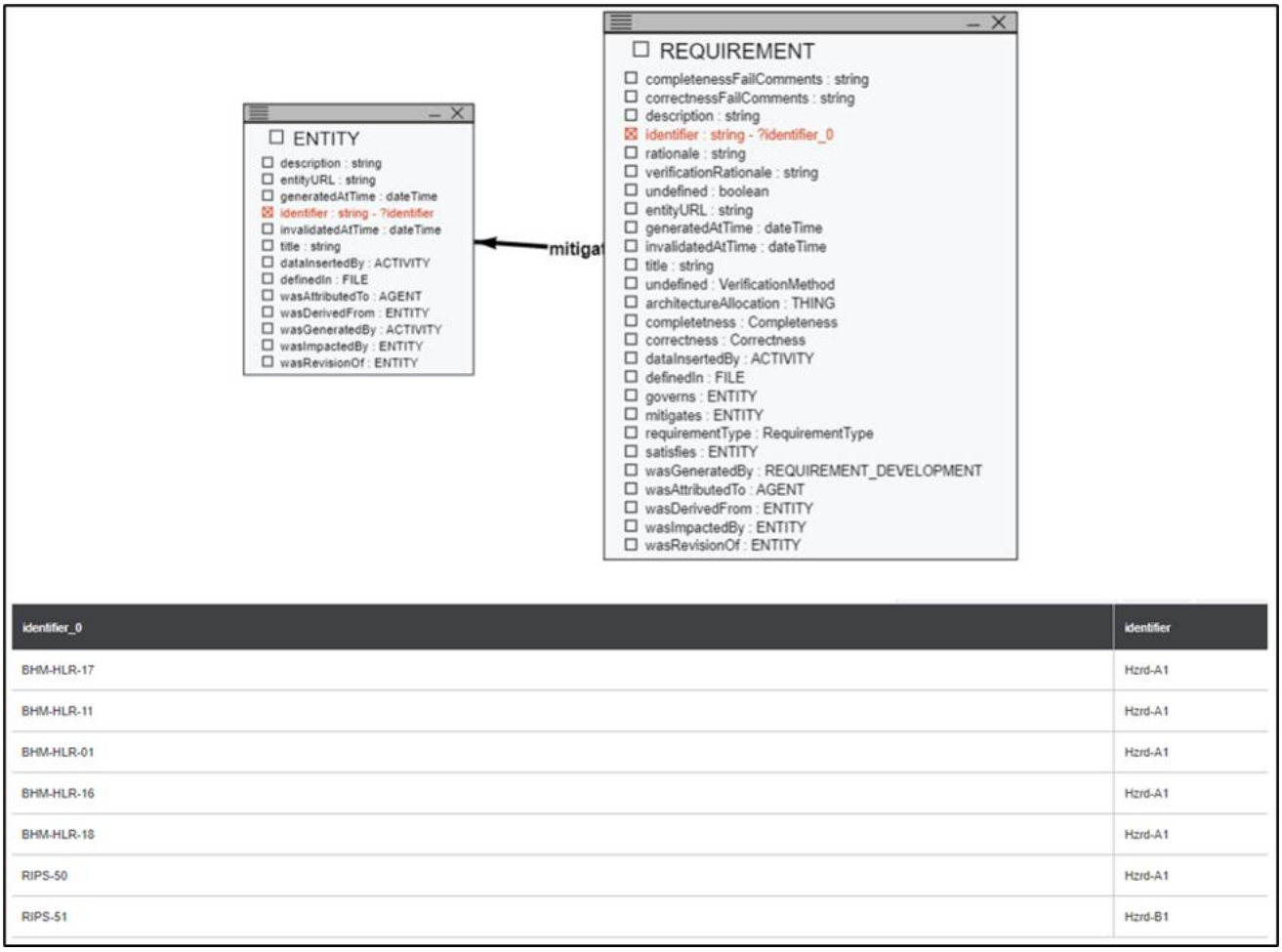



Figure 25. RACK query and results connecting HLRs to system requirements

**Requirements to hazards**

Shown below in Figure 26 is the RACK query (and partial result) that shows the traceability between requirements and hazards using the "mitigates" property of the REQUIREMENT class.




| identifier_0 | identifier |
|---|---|
| BHM-HLR-17 | Hzrd-A1 |
| BHM-HLR-11 | Hzrd-A1 |
| BHM-HLR-01 | Hzrd-A1 |
| BHM-HLR-16 | Hzrd-A1 |
| BHM-HLR-18 | Hzrd-A1 |
| RIPS-50 | Hzrd-A1 |
| RIPS-51 | Hzrd-B1 |

Figure 26. RACK query and results connecting requirements to hazards

In the future, we plan to develop an automated toolchain such that once the requirements, OP arguments, and evidence supporting the premises are ingested into RACK, GSN assurance case fragments can be automatically generated.

# 11 Reflection

The AI model development community has devised strategies and rules that, implicitly or explicitly, address multiple issues that are relevant and related to assurance of AI models, more specifically, machine learning models. Since the underlying framework of logical inference that ML models try to emulate is induction, the outputs produced by these models are not categorically defensible; any individual prediction, in the absence of further context, is equally likely to be incorrect or grossly inaccurate. Moreover, this induction is performed directly from data using non-linear, empirical models like ANNs, with extremely large parametric spaces. This

complexity leads to models that are largely black-box in nature - namely it is hard to characterize how the model performs its inference or estimation. A predominant approach to address this risk has involved devising methods that can additionally quantify prediction uncertainty, even if the uncertainty metrics used are also empirical and do not have clean semantics (e.g., in classification problems, the softmax output produced by the ANN is used as a proxy for prediction probabilities, even if they are known to not be indicative of the true degree of accuracy of prediction, or confidence). More recent approaches explicitly recognize and estimate two different categories of uncertainty, namely aleatoric versus epistemic, to address the independent causes associated with each category, thus deconstructing uncertainty arising from stochasticity in the data generation process from the uncertainty caused by lack of necessary information in the training data (i.e., ignorance). Model opaqueness has also been addressed as an issue of interpretability or transparency of the model, and methods have been created that can link a model's prediction to salient portions of the raw input, thus implicitly ascribing cause or saliency to the inference produced (e.g., classification of the picture as a "cat" is augmented with highlighting regions of the picture the capture the "whiskers" and "ears"). Multiple initiatives like XAI (Gunning & Aha, 2019) and Trustworthy AI (Thiebes, Lins, & Sunyaev, 2021) have pursued the goal of reducing the opacity of black-box models like ANNs and led to the design of techniques to do the same.

Assurance is also related to model performance, and the expected performance of a data driven model like ANN is a function of the model configuration, its training procedure and parameters, and the data used to train the model—poor choices on any of the three can result in an inferior model, making its predictions unreliable. Many strategies and mechanisms (hyperparameter optimization, dropout, early stopping) have been devised to guide modelers in the choice of model configuration and training parameters, such that for a given training dataset, it will lead to a model with good expected performance. Model performance is also affected by the training data used to estimate its parameters - the distribution of this data informs the competence of the model to make predictions in various regions of the input data space - one common issue arises when the model is making predictions in regions which are outside the envelope of the training data (i.e., regions of extrapolation). It is well understood that ANNs are unreliable when making predictions in the region of extrapolation, and care is often taken to make sure that the model is not making predictions in this region; from an assurance perspective, the goal would be to ensure that ANNs are not exposed to inputs that are in the region of extrapolation for the model. It is important to make sure that the training data sufficiently covers, without gaps, regions of the input space as characterized by the operating conditions of the model, in order for the model predictions to be uniformly reliable across the entire input space. Given that the input space is

often continuous, and the training data can only have a finite number of discrete samples, all data-driven models inherently perform interpolation, whose quality in a region of the input is dictated by the density of training samples in that region. Often, data-driven models do not explore this issue since they do not have control over the data generation process; it is common for AI models to be built starting with data that is available at the outset.

The goal of all such strategies is to help design a model that generalizes and shows good prediction performance when applied to data samples that were not seen by the model (this is a balancing act as is characterized by the bias-variance tradeoff) and involves finding the right balance between overfitting or underfitting the model to the training data. When available training data is found to be lacking in terms of volume, techniques have been devised to generate synthetic examples (e.g., data augmentations, generative models like Generative Adversarial Networks (GANs)) in order to increase data volume; while this often addresses the data sufficiency issue to help address the bias-variance tradeoff, it introduces the new risk of these synthetic samples being different from the actual distribution of input data, thereby causing the model to learn the underlying concept incorrectly.

Another property of AI models like ANN that is relevant to the assurance question is the property of robustness[2], whereby minute perturbations to the inputs can cause the model outputs to change drastically - this can lead to a model whose performance then becomes extremely sensitive to properties of the ambient conditions in which the model is deployed. A special case of this robustness question is adversarial robustness, which involves algorithmically generating perturbations to input in order to drastically change model output. A large body of approaches have been developed for tackling adversarial robustness and guarding models from falling prey to malicious attacks. While adversarial robustness is about attacking a model that has already been trained, Trojans are data samples that are corrupted in a way to influence model performance during training, making them vulnerable to manipulation during operation. Fairness metrics have also dominated in the AI community to measure ethics of using AI models for making social decisions (e.g., job offers, loan evaluation). Such metrics are meant to characterize and measure the ability of AI models to be fair and unbiased with respect to demographics representing minorities in society (e.g., gender, race), which are naturally under-represented in the training data, thereby biasing the model to incompletely characterize outcomes for such classes. Bias can also be inherently present in the training data on account of existing social

[2] Robustness in the context of a NN-MODEL implies that the model will be resilient to minute perturbations to the inputs while in the context of a NN-SOFTWARE-COMPONENT, it implies that the component will respond correctly to abnormal (off-nominal) inputs.

practices that are themselves biased; training a model on this data without correcting for such bias will lead to the model producing outcomes that reflect the same bias in its predictions.

The value alignment problem identified recently in AI presents the broadest perspective on the assurance question for AI models. Most data-driven models, when built under supervision (training data also contains outputs), make use of an appropriate metric (e.g., mean square error) that they try to optimize, towards the larger goal of estimating the model parameters. This metric or objective is often a quantity whose optimization can be heuristically linked to what can be considered as a good model. However, such metrics are only proxies of the true intent for which the model is being developed, and thus it is imaginable that we can run into cases where the correspondence between the metric being optimized by the model and the true intent cease to align, thereby making the model deviate from the true purpose for which it was developed. Fundamentally, this results from our inability to define a concrete metric or objective that exactly aligns with the true intent of the model. This is a new subfield in AI and its expression of the assurance problem, especially *intent* is amply clear. Even if this area is nascent and not a lot of approaches have been developed towards addressing this issue, we expect it will result in a body of work that will intersect most closely to the work being done by the assurance community itself.

In the light of the challenges described above, we have tried to be as thorough as possible in designing the premises for our OP arguments to ensure that they can correctly support the claims. However, it should be noted that other alternative combinations of premises for each argument may also be possible. Moreover, since our primary goal was to exhaustively identify sufficient premises, we have been conservative in some cases (e.g., it may be impossible to guarantee premise B2 if the operating conditions are defined by continuous parameters).

Additionally, when working through the phases of development for a system relying on AI/ML, the use of OP may alter the development approach. For traditional development of DAL D software, HLRs are written and validated followed by review for their applicability to the safety assessment. When working through OP arguments, the premises supporting Innocuity place a greater emphasis on the identification of failure modes specific to the software component. These failure modes may drive the need for additional mitigations in the design. In the requirements set, this results in a larger set of derived requirements. When this was applied to the AI/ML software components in the RIPS system, we focused on assuring failure modes in the ANNs would not contribute to incorrect decisions made by the software component. Mitigations around each ANN were needed to assure that decisions were never left to the ANN for input vectors that

would result in low prediction performance, such as input combinations outside the bounds of the training data.

The hybrid certification strategy adopted is not only beneficial when strategically applying OP to sub-components developed with new or novel methodologies that do not align well with existing traditional certification standards, but also when integrating these subcomponents with previously developed software and COTS software as done in this work. The hybrid certification strategy must be identified in the planning documents (e.g., Plan for Software Aspects of Certification) and the certification approach for each sub-system clearly identified. Care must be taken to ensure that the design and or development assurance of the integration of sub-components is covered (e.g., software/software integration) by an adequate certification standard. This will likely be achievable by the traditional certification standards.

The OP approach to certification of the SoC and SoH sub-components is a promising alternative to attempting to certify AI/ML under DO-178C. The OP framework provides the opportunity to streamline the activities required to adequately show compliance of software developed with novel development methodologies such as ANNs. While the use case selected was only DAL D, reusing the generic argument-based OP approach and extending it to higher DALs seems feasible. One area of improvement would be to remove the reliance on DO-178C objective applicability referenced by this OP approach. Nevertheless, given the paucity of prior work on this topic, we believe that our work provides a suitable foundation for developing a well-accepted OP-based certification approach for AI/ML-based avionics systems in the future. Additionally, there is close alignment with the principles of Trustworthy AI, such as robustness, safety, and fairness, that may be useful for establishing “trust” in AI-based systems.

# 12 Related work

Daw et al. (2023) have used OPs (Holloway, 2019) and Overarching Properties Related Arguments (OPRA) (Wasson & Holloway, 2022) at the system level by performing a case study of an auxiliary power unit in an aircraft very similar to our RIPS system. They use the FAN notation to capture OPs and the system models are captured in AADL linked to MATLAB simulation toolchains. They observe that OPs capture with enough precision a single critical component of a safety-critical system and that OPs need a hybrid and multi-disciplinary approach to be used along with existing certification guidelines and related standards (Daw & Beecher, 2023). They have also similarly applied a hybrid approach to OPs on a surrogate UAV model with a collision avoidance system (Daw, Beecher, Holloway, & Graydon, 2021). They note that use of AI/ML components is acceptable to model “macro-behaviors” of safety-critical

software but do not develop a corresponding OP framework as outlined in this paper. Durling et al. (2021) have aligned Adaptive Stress Testing (AST) of safety-critical airborne software with OPs while also aligning AST with DO-178Cs objectives such as Robustness Testing, Exhaustive Testing & Product Service History for airworthiness.

However, they note a gap in linking OPs to DAL-specific evidence that is prescribed by DO-178C objectives. In comparison, we demonstrate OP possession for the DAL D RIPS system. Graydon et al. (2021) have retrospectively demonstrated OP possession of a DAL C system named SAFEGUARD, intended for providing geo-fencing capabilities for unmanned aerial vehicles. They have performed a detailed case study providing a plausible approach to establish OP possession of an already certified system. They reuse already established DO-178C qualification packs for some of SAFEGUARD's sub-components such as its operating system, to argue for innocuity. Their OP possession argument highlights the inter-related nature of OP arguments with existing DO-178C evidence and objectives (Graydon & Cronin, 2021). Blood et al. (2023) identify several AI failure modes that can occur in systems with AI function and advocate the use of traditional reliability and hazard analysis techniques as a starting point to manage their outcomes iteratively during their design, development, and operation.

Our paper identifies AI failure modes like their work, such as *data pipeline failures, robustness failure of NN model & validation of inputs to NN model (automated naivete)* and refines them as premises into the OP framework. The EASA concept paper for use of AI/ML in aviation (Torens, Durak, & Dauer, 2022) provides guidelines and a regulatory framework prescribing different sets of assessment objects at each stage of AI/ML deployment from training of the AI/ML models to their testing and use in real systems. They develop the guidelines for Urban Air Mobility (UAM) while considering increasing levels of autonomy of AI/ML functions. Further, EASA and Collins Aerospace (2023) have provided an extensive survey about the use of Formal Methods for ML along with a methodology for FM-based ML Assurance while aligning with EASA concept paper objectives. They have applied their methodology on an on-ground ML-based Remaining Useful Life (RUL) estimator that aids flight preparation (EASA and Collins Aerospace, 2023). The SAE G34 meeting (Brat G. , 2021) compares the different tools supported by NASA ranging from requirements capture to the development of assurance cases of safety-critical systems with respect to the AI/ML guidelines provided in the EASA concept paper. Aerospace Industry and academic partners have developed a roadmap and vision 2045 for the verification & validation of autonomous systems (Brat, et al., 2023) extensively along several technical areas ranging from Machine Learning V&V techniques, Model-Based Systems Engineering (MBSE) for AI/ML and certification of AI/ML systems. Kaakai et al. (Kaakai, et al., 2022) have developed a machine learning product development lifecycle that also includes

EASA guidelines for ML that will eventually become an aeronautical standard, namely, AS6983. Aerospace Vehicle Systems Institute (AVSI) (2020) have produced a report, "AFE 87 - Machine Learning", that contains recommendations for development of assurance guidelines and certification methodologies for AI/ML in safety-critical systems. Usman et al. (2022) have developed coverage metrics for Deep Neural Networks (DNNs) but observed that their coverage metrics are not sensitive to the functional diversity of DNNs. Sun et al. (2022) have used NN verification frameworks such as Marabou framework (Katz, et al., 2019) to verify NNs against model poisoning attacks during the training phase and have applied their approach to the MNIST digit recognition and the German Traffic Sign Recognition benchmarks which are classified using small NN models. Irfan et al. (2020) performed formal verification of neural networks for small, unmanned aircraft collision avoidance using the Marabou framework. Finally, EASA's Artificial Intelligence Roadmap 2.0 (with a minor section partly written by ChatGPT/Open AI GPT-3 and edited by human experts) (EASA, 2023) outline the applicability of human-aiding AI with several avionics applications from air-traffic management to aircraft production & maintenance while also providing the challenges involved developing assurance and establishing *trustworthiness in AI*.

# 13 Conclusion

In this document, we presented a set of premise-based argument structures that can be used for arguing for the OP possession of neural network-based sub-components and proposed potential strategies to generate supporting evidence for an example DAL D system. To our knowledge, this work is the first attempt at laying a generic foundation for using OPs to certify AI/ML-based systems. Although our approach presents a good initial foundation for using OPs for the certification of AI/ML-based systems, more work is needed to for its practical realization. One potential direction of future work would be to analyze the practical implications of our premises and investigate ways to generate supporting evidence. Similarly, work is needed to study the effectiveness of our premises and detect any potential inconsistencies or vulnerabilities in our argument structures. In subsequent phases of this project, we plan to continue investigating our proposed approach with respect to DAL A, B, and C systems.

# 14 References


Irfan, Ahmed, Kyle D. Julian, Haoze Wu, Clark Barrett, Mykel J. Kochenderfer, Baoluo Meng, and James Lopez. (2019). Towards verification of neural networks for small unmanned aircraft collision avoidance. *AIAA/IEEE 39th Digital Avionics Systems Conference (DASC).*

*ACAS Xu DNN Repository*. (2017). Retrieved 2023, from https://github.com/guykatzz/ReluplexCav2017/tree/master/nnet

Aerospace Vehicles Systems Institute. (2020). *AFE87 - Machine Learning*. Retrieved from www.avsi.aero/wp-content/uploads/2020/06/AFE-87-Final-Report.pdf

ARINC. (2010). *Recorder Independent Power Supply (RIPS).* ARINC 777-2, Aeronautical Radio Incorporated.

Arkoudas, K., & Musser, D. (2017). *Fundamental proof methods in computer science: a computer-based approach.* MIT Press.

B. Saha and K. Goebel. (2007). "Battery Data Set", NASA Prognostics Data Repository. *NASA Ames Research Center.* Moffet Field, California.

Blood, J. C., Herbert, N. W., & Wayne, M. R. (2023). Reliability assurance for AI systems. *2023 Annual Reliability and Maintainability Symposium (RAMS).*

Brat, G. (2021). Are we ready for the first EASA guidance on the use of ML in aviation. *SAE G34 Meeting.*

Brat, G., Yu, H., Atkins, E., Sharma, P., Cofer, D., Durling, M., . . . Garg, K. (2023). *Autonomy verification & validation roadmap and vision 2045.* Tech. Rep.

*Cameo Enterprise Architecture*. (n.d.). Retrieved from www.3ds.com/products-services/catia/products/ no-magic/cameo-enterprise-architecture/

Chen, C.; Seff, A.; Kornhauser, A.; Xiao, J. (2015). "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. *IEEE International Conference on Computer Vision (ICCV),.*

Cheng Wang, Tongtong Ji, Feng Mao, Zhenpo Wang, Zhiheng L. (2021). Prognostics and Health Management System for Electric Vehicles with a Hierarchy Fusion Framework: Concepts, Architectures, and Methods. *Advances in Civil Engineering, vol. 2021, Article ID 6685900.*

D. Cofer et al. (2020). Run-Time Assurance for Learning-Based Aircraft Taxiing. *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC).* San Antonio, Texas.

*Data Distribution Services Specification*. (n.d.). Retrieved from www.omg.org/omg-dds-portal/

Daw, Z., & Beecher, S. (2023). Assuring safety in a flexible aerospace certification—lessons learned on applying OPs at the system level—. *2023 IEEE International Systems Conference (SysCon).*

Daw, Z., Beecher, S., Holloway, M., & Graydon, M. (2021). Overarching properties as means of compliance: An industrial case study. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference.*

Dean A. Pomerleau. (1992). Progress in Neural Network-based Vision for Autonomous Robot Driving. *Proceedings of the Intelligent Vehicles `92 Symposium.*

*Deep Learning Blockset*. (n.d.). Retrieved from www.mathworks.com/solutions/deep-learning.html

(2011). *DO-178C Software Considerations in Airborne Systems and Equipment Certification.* RTCA.

DO-254 Design Assurance Guidance for Airborne Electronic Hardware. (2000).

Durling, M., Herencia-Zapana, H., Meng, B., Meiners, M., Hochwarth, J., Visser, N., . . . Valapil, V. (2021). Certification considerations for adaptive stress testing of airborne software. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC).*

EASA. (2023). *EASA Artificial Intelligence Roadmap 2.0*. Retrieved from www.easa.europa.eu/en/downloads/137919/en

EASA and Collins Aerospace. (2023). *Formal methods use for learning assurance (formula).* Tech. Rep. .

EUROCAE. (2013). *ED-112A, Minimum Operational Performance Specification for Crash Protected Airborne Recording Systems.* Malakoff, France: EUROCAE.

FAA. (2015). *Technical Standard Order: Recorder Independent Power Supply*. Federal Aviation Administration (FAA), TSO-C155b.

FAA. (2021). *Special Conditions: magniX USA, Inc., magni350 and magni650 Model Engines; Electric Engine Airworthiness Standards.* Federal Aviation Administration.

GE. (2021, October 1). *GE Press Release*. Retrieved from https://www.geaerospace.com/press-release/other-news-information/ge-aviation-selected-nasa-hybrid-electric-technology

*GE High Assurance GitHub Repository*. (2022, 12 22). Retrieved from Rapid Assurance Curation Kit: https://github.com/ge-high-assurance/RACK

General Elelctric Aviation Systems LLC (US). (2022). *Standard Model for the Recorder Independent Power Supply, RIPS.mdzip.*

Goebel, K., Saha, B. (2015). Prognostics Applied to Electric Propulsion UAV. *Handbook of Unmanned Aerial Vehicles. Springer, Dordrecht.*

Graydon, M. S., & Cronin, J. D. (2021). *Retrospectively documenting satisfaction of the overarching properties: An exploratory prototype.* Tech. Rep. .

Gunning, D., & Aha, D. (2019). DARPA's explainable artificial intelligence (XAI) program. *AI magazine, 40*(2), 44-58.

Hogge, E. F., Bole, B. M., Vazquez, S. L., & Celaya, J. (2015). Verification of a Remaining Flying Time Prediction System for Small Electric Aircraft. *Annual Conference of the Prognostics and Health Management.*

Holloway, C. M. (2019). *Understanding the Overarching Properties.* Tech. Rep.

Holloway, C. M. (2020). The Friendly Argument Notation (FAN).

Irfan, A., Julian, K. D., Wu, H., Barrett, C., Kochenderfer, M. J., Meng, B., & Lopez, J. (2020). Towards verification of neural networks for small unmanned aircraft collision avoidance. *2010 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC).*

*Jenkins CI/CD* . (n.d.). Retrieved from www.jenkins.io

Julian, Kyle D., Mykel J. Kochenderfer, and Michael P. Owen. (2019). Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics 42, no. 3.*

Kaakai, F., Dmitriev, K., Baskaya, E., Bezzecchi, E., Bharadwaj, R., Brown , B., . . . Travers, C. (2022). Toward a machine learning development lifecycle for product certification and approval in aviation. *SAW International Journal of Aerospace, 15*(01-15-02-0009), 127-143.

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., . . . Zeljić, A. (2019). The Marabou framework for verification and analysis of deep neural networks. *Computer Aided Verification: 31st International Conference, CAV 2019.* New York City, NY, USA.

Keller, K., Kevin Swearingen, J J Sheahan, Michelle D. Bailey, Jonathan Mark Dunsdon, Katarzyna Przytuła and Brett Jordan. (2006). Aircraft electrical power systems prognostics and health management. *IEEE Aerospace Conference.*

Kelly, T., & Weaver, R. (2004). The Goal Structuring Notation–A Safety Argument Notation. *Proceedings of the dependable systems and networks 2004 workshop on assurance cases.*

Kulkarni, C. S., Corbetta, M., & Robinson, E. (2020). Enhancing Fault Isolation for Health Monitoring of Electric Aircraft Propulsion by Embedding Failure Mode and Effect Analysis into Bayesian Networks. *Annual Conference of the PHM Society.*

Kulkarni, C., Hogge, E., Quach, C., & Goebel, K. (2007). *HIRF Battery Data Set.* (NASA Ames Research Center, Moffett Field, CA) Retrieved from https://www.nasa.gov/intelligent-systems-division

Litjens, G., Kooi, T., Babak Ehteshami, B., Setio, A., Ciompi, F., Ghafoorian, M., . . . Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. *Medical image analysis.*

Meng, B., Paul, S., Moitra, A., Siu, K., & Durling, M. (2021). Automating the assembly of security assurance case fragments. *Computer Safety, Reliability, and Security.*

Moitra, A., Cuddihy, P., Siu, K., Meng, B., Interrante, J., Archer, D., . . . Russell, D. (2022). A Semantic Reference Model for Capturing System Development and Evaluation. *16th IEEE International Conference on Semantic Computing,*. CA, USA.

NASA. (2022, 12 22). *NASA Prognostics Center of Excellence Data Set Repository*. Retrieved from https://www.nasa.gov/content/prognostics-center-of-excellence-data-set-repository

NASA. (n.d.). *NASA Battery Dataset.* Retrieved from https://phm-datasets.s3.amazonaws.com/NASA/5.+Battery+Data+Set.zip

*Open Neural Network Exchange*. (n.d.). Retrieved from onnx.ai

Palanisamy, Rajendra prasath & Kulkarni, Chetan & Corbetta, Matteo & Banerjee, Portia. (2022). Fault detection and Performance Monitoring of Propellers in Electric UAV. *10.1109/AERO53065.2022.9843261.*

Paul, S., Prince, D., Iyer, N., Durling, M., Visnevski, N., Meng, B., . . . Meiners, a. M. (2023). Towards the Certification of Neural Networks using Overarching Properties: An Avionics Case Study. *AIAA/IEEE Digital Avionics Systems Conference (DASC).* Barcelona.

*Raspberry Pi v. 4.0.* (n.d.). Retrieved from www.raspberrypi.com

Ravaioli, U. J., Cunningham, J., McCarroll, J., Gangal, V., Dunlap, K., & Hobbs, K. L. (2022). Safe reinforcement learning benchmark environments for aerospace control systems. *IEEE Aerospace Conference (AERO).*

RTCA, Inc. (2010). *DO-160G, Environmental Conditions and Test Procedures for Airborne Equipment.* Washington, DC, USA: RTCA, Inc.

RTCA, Inc. (2011). *DO-178C, Software Considerations in Airborne Systems and Equipment Certification.* Washington, DC, USA: RTCA, Inc.

*RTI Connext DDS*. (n.d.). Retrieved from www.rti.com/products/

SAE. (1996). *ARP4761 Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* SAE International.

SAE. (2010). *ARP 4754A: Guidelines for Development of Civil Aircraft and Systems.* SAE S-18. SAE International.

SafeRL. (n.d.). *SafeRL Benchmark Repository*. Retrieved 09 30, 2022, from https://github.com/act3-ace/SafeRL

Sanabria, P. (2019). *BatteryDatasetImplementation*. Retrieved from https://github.com/psanabriaUC

*Semantic Application Design Language*. (2022, 12 22). Retrieved from SADL GitHub: https://github.com/SemanticApplicationDesignLanguage/sadl

*Simulink*. (n.d.). Retrieved from www.mathworks.com/solutions/simulik.html

*Simulink Embedded Coder*. (n.d.). Retrieved from www.mathworks.com/products/embedded-coder.html

Sripad, S., Bills, A., & Viswanathan, V. (2021). A review of safety considerations for batteries in aircraft with electric propulsion. *MRS Bulletin.*

Sun, Y., Usman, M., Gopinath, D., & Păsăreanu, C. S. (2022). VPN: Verification of Poisoning in Neural Networks. *Software Verification and Formal Methods for ML-Enabled Autonomous Systems: 5th International Workshop, FoMLAS 2022, and 15th International Workshop, NSV 2022.* Haifa, Israel.

*TensorFlow*. (n.d.). Retrieved from www.tensorflow.org

Thiebes, S., Lins, S., & Sunyaev, A. (2021). Trustworthy artificial intelligence. *Electronic Markets, 31*, 447-464.

Torens, C., Durak, U., & Dauer, J. C. (2022). Guidelines and regulatory framework for machine learning in aviation. *AIAA Scitech 2022 Forum.*

Usman, M., Sun, Y., Gopinath, D., Dange, R., Manolache, L., & P, C. S. (2022). An overview of structural coverage metrics for testing neural networks. *International Journal on Software Tools for Technology Transfer*, 1-13.

Valapil, V. T., Herencia-Zapana, H., Durling, , M., Armstrong, K., Paul, S., Borgyos, S., . . . Premerlani, W. (2021). Towards Formalization of a Data Model for Operational Risk Assessment. *IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, (p. 10). San Antonio, Texas.

Visnevski, N. A. (2021). A novel, model-based, specification-driven embedded software integration platform. *IEEE Aerospace Conference (50100).*

Visnevski, N., Hubscher-Younger, T., Rajhans, A., & Meng, B. (2020). Automatic synthesis of information flow driven execution managers for embedded software applications. *AIAA/IEEE 39th Digital Avionics Systems Conference (DASC).*

Wasson, K. S., & Holloway, C. M. (2022). An Introduction to Constructing and Assessing Overarching Properties Related arguments (OPRAs): Version 1.0.

Zhang, Z., Beck, M., Winkler, D., Huang, B., Sibanda, W., & Goyal, H. (2018). Opening the black box of neural networks: methods for interpreting neural network models in clinical applications. *Annals of translational medicine, 6*(11).

# A Comprehensive list of RIPS requirements

Table A- 1. RIPS system requirements table

| Id | Name | Text |
|---|---|---|
| RIPS-01 | Backup Power Time | When aircraft power to the recorder drops below 18 Vdc, the RIPS shall make DC power available for 10 minutes. |
| RIPS-02 | Backup Time Tolerance | The tolerance on the time of 10 minutes output shall be ±1 minute. |
| RIPS-03 | Recharge Timing | From the time aircraft power greater than 22 Vdc is available until the RIPS is capable of providing the full 10 minutes of power shall be no more than 15 minutes. |
| RIPS-04 | Battery Replacement | The RIPS shall have a replaceable battery |
| RIPS-05 | Maintenance Discrete | The RIPS shall have a “Maintenance Required” Standard Discrete Output. |
| RIPS-06 | RIPS Active Discrete | The RIPS shall have a “RIPS Active” Standard Discrete Output. |
| RIPS-07 | No Fault Discrete | The RIPS shall have a “No Fault” Standard Discrete Output. |
| RIPS-08 | Maintenance Discrete Behavior | The RIPS shall set the "Maintenance Required" discrete in the “ground” state when the RIPS has determined that the internal battery needs to be replaced. |
| RIPS-09 | RIPS Active Discrete Behavior | The RIPS shall set the "RIPS Active" discrete in the “ground” state when the RIPS is supplying power to the recorder. |
| RIPS-10 | No Fault Discrete Behavior | The RIPS shall set the "RIPS Active" discrete in the “ground” state when the RIPS has determined that it is able to supply back-up power to the recorder for the duration specified in RIPS-01 and that it has detected no internal faults or external wiring faults. |
| RIPS-11 | Operating Temperature Range | The RIPS shall operate from -15 degC to 55 degC. |
| RIPS-12 | RIPS Form Factor | The RIPS shall form factor shall match the dimensions defined in ARINC 777-2 Attachment 5 Figure 5-1. |
| RIPS-13 | Weight | The RIPS shall weigh no more than 5 pounds. |
| RIPS-14 | Connector Type | The RIPS shall use D38999/20JC35P connectors. |

| Id | Name | Text |
|---|---|---|
| RIPS-15 | Battery Recharge Initiation | When aircraft power is available, the RIPS shall initiate a recharge of the internal battery when estimated remaining charge is less than 30%. |
| RIPS-16 | Battery Recharge Completion | After a battery recharge has been initiated and while aircraft power is available, the RIPS shall continue recharge the internal battery until the estimated remaining charge is above 99%. |
| RIPS-17 | Minimum Operating Pressure | The RIPS shall operate down to a minimum operating pressure of 57.18 kPa. |
| RIPS-18 | Operating Temperature Variation | The RIPS shall operate in an environment with a maximum temperature rate of change of 2 degC per minute. |
| RIPS-50 | Loss of Backup Power | Loss of ability for the RIPS to provide backup power to the Flight Data Recorder shall be considered a MINOR failure condition. |
| RIPS-51 | Inadvertent Backup Power | Backup power provided by the RIPS to the Flight Data Recorder shall be considered a MINOR failure condition. |

Table A- 2. Battery Health Monitor HLRs

| Id | Name | Text |
|---|---|---|
| BHM-HLR-01 | Monitor Battery | The Battery Health Monitor shall indicate battery maintenance is required when the State-of-Health is less than 70% with a tolerance of +/- 1%. |
| BHM-HLR-02 | Temperature Input | The Battery Health Monitor shall receive temperature as an input. |
| BHM-HLR-03 | Temperature Range | The Battery Health Monitor shall estimate state-of-charge for batteries operating from -15 degC to 55 degC. |
| BHM-HLR-04 | Battery Terminal Voltage Input | The Battery Health Monitor shall receive battery terminal voltage measurements as an input. |
| BHM-HLR-05 | Battery Terminal Voltage Range | The Battery Health Monitor battery terminal voltage measurement input shall allow for voltage measurements from 0 Vdc to positive 40 Vdc. |

| Id | Name | Text |
|---|---|---|
| BHM-HLR-06 | Battery Output Current Input | The Battery Health Monitor shall receive battery output current measurements as an input. |
| BHM-HLR-07 | Battery Output Current Range | The Battery Health Monitor battery output current measurement input shall allow for current ranges from 0 A to 3 A. |
| BHM-HLR-08 | Time Input | The Battery Health Monitor shall receive operating time since last charge as an input. |
| BHM-HLR-09 | Time Range | The Battery Health Monitor time measurement input shall allow for time measurements from 0 seconds up to 3.6e+7 seconds. |
| BHM-HLR-10 | Maintenance Required Output | The Battery Health Monitor shall provide Maintenance Required state as an output. |
| BHM-HLR-11 | Charge Command Behavior | The Battery Health Monitor shall set the charge command output to the active state when the State-of-Charge is less than or equal to 30% with a tolerance of +/- 3%. |
| BHM-HLR-12 | Charge Command | The Battery Health Monitor shall provide a charge command as an output. |
| BHM-HLR-13 | State-of-Health Neural Network | The Battery Health Monitor shall implement a neural network to compute the State-of-Health. |
| BHM-HLR-14 | State-of-Charge Neural Network | The Battery Health Monitor shall implement a neural network to compute the State-of-Charge. |
| BHM-HLR-15 | Battery Terminal Voltage Out of Range | The Battery Health Monitor shall indicate a failure if a voltage measurement is received outside of the range defined in BHM-HLR-05. |
| BHM-HLR-16 | Temperature Out of Range | The Battery Health Monitor shall indicate a failure if a temperature measurement is received outside of the range defined in BHM-HLR-03. |
| BHM-HLR-17 | Battery Output Current Out of Range | The Battery Health Monitor shall indicate a failure if a current measurement is received outside of the range defined in BHM-HLR-07. |
| BHM-HLR-18 | Cycle Count Input | The Battery Health Monitor shall receive a count of the charge-discharge cycles for the current battery. |
| BHM-HLR-19 | Cycle Count Input Range | The Battery Health Monitor cycle count input shall allow for values from 0 to 200 cycles. |
| BHM-HLR-20 | Load Voltage Input | The Battery Health Monitor shall receive voltage measurements at the load as an input. |

| Id | Name | Text |
|---|---|---|
| BHM-HLR-21 | Load Voltage Input Range | The Battery Health Monitor load voltage measurement input shall allow for voltage measurements from 0 Vdc to positive 40 Vdc. |
| BHM-HLR-22 | Load Current Input | The Battery Health Monitor shall receive the current measurements at the load as an input. |
| BHM-HLR-23 | Load Current Input Range | The Battery Health Monitor load current measurement input shall allow for current ranges from 0 A to 3 A. |
| BHM-HLR-24 | Charge Command Complete | The Battery Health Monitor shall set the charge command output to the inactive state when the State-of-Charge is greater than or equal to 99% +/- 1%. |
| BHM-HLR-25 | State-of-Health Training Range | The Battery Health Monitor State-of-Health Neural Network shall be trained with a training data set that contains State-of-Health values from 0% to 100%. |
| BHM-HLR-26 | State-of-Health Training Gap Coverage | The Battery Health Monitor State-of-Health Neural Network shall be trained with a training data set that does not contain a gap in the State-of-Health values greater than 0.5% for the range defined in BHM-HLR-25. |
| BHM-HLR-27 | State-of-Charge Training Range | The Battery Health Monitor State-of-Charge Neural Network shall be trained with a training data set that contains State-of-Charge values from 0% to 100%. |
| BHM-HLR-28 | State-of-Charge Training Gap Coverage | The Battery Health Monitor State-of-Charge Neural Network shall be trained with a training data set that does not contain a gap in the State-of-Charge values greater than 0.5% for the range defined in BHM-HLR-27. |
| BHM-HLR-29 | Battery Voltage Runtime Monitor | The Battery Health Monitor shall indicate battery maintenance is required if the measured battery voltage is less than 17 Vdc. |

# B RIPS FHA Worksheet

The FHA worksheet for the RIPS is included in Table B- 1.

Table B- 1. RIPS FHA Summary

| Id | Function | Failure Condition | Flight Phase | Effect On System | Severity | Min Required DAL | Severity Justification | Verification Method | Affected Systems | Documentation |
|---|---|---|---|---|---|---|---|---|---|---|
| **A1** | Provide backup power to the Flight Data Recorder | Loss of ability to provide backup power | All | Unable to record aircraft state and performance parameters | Minor | D | TSO-C155b section 3.b(2) classifies this as a minor failure condition | FHA | Flight Data Recording System | Power from the RIPS is only provided to the Flight Data Recorder and will only impact the elements of the Flight Data Recording System. |
| **B1** | Provide backup power to the Flight Data Recorder | Backup power provided when not required | All | Erroneously provided power may result in the inability to record aircraft state and performance parameters | Minor | D | TSO-C155b section 3.b(1) classifies this as a minor failure condition | FHA | Flight Data Recording System | Power from the RIPS is only provided to the Flight Data Recorder and will only impact the elements of the Flight Data Recording System. |

# C Complete SADL formalization of requirements and hazards

## FHA Ontology

```
//-------------------------------------------------------------------
-------------------------
 //-- Additional classes required
 //-------------------------------------------------------------------
-------------------------
 Severity (note "Severity Types taken from FAA Circular AC No:
23.1309-1E") is a type of THING,
      must be one of {Negligible (note "No safety effect: failure
conditions that would not affect the operational capability of the
airplane or increase crew workload"),
                      Minor (note "Failure conditions that would
not significantly reduce airplane safety and involve crew actions that
are within their capabilities."),
                      Major (note "Failure conditions that would
reduce the capability of the airplane or the ability of the crew to
cope with adverse operating conditions to the extent that there would
be a significant reduction in safety margins or functional
capabilities."),
                      Hazardous (note "Failure conditions that
would reduce the capability of the airplane or the ability of the crew
to cope with adverse operating conditions"),
                      Catastrophic (note "Failure conditions that
are expected to result in multiple fatalities of the occupants, or
incapacitation or fatal injury to a flight crewmember normally with
the loss of the airplane.")}.

 DesignAssuranceLevel (note "The minimum Design Assurance Level") is a
type of THING
      must be one of {LevelA (note "Level A"),  // Cannot use 'A' since
it is a keyword in SADL, so using LevelA instead
                      LevelB (note "Level B"),
                      LevelC (note "Level C"),
                      LevelD (note "Level D")}.

 VerificationMethod (note "The method used for verifying a component")
is a type of ACTIVITY
      must be one of {FHA (note "Functional Hazard Assessment"),
                      FTA (note "Fault Tree Analysis"),
                      DD (note "Dependency Diagram"),
```

```
                              MA (note "Markov Analysis"),
                              FMEA (note "Failure Modes and Effects
Analysis"),
                              FMES (note "Failure Modes and Effects
Summary"),
                              ZSA (note "Zonal Safety Analysis"),
                              CMA (note "Common Mode Analysis"),
                              PRA (note "Particular Risk Analysis")}.

 Phase (note "The flight phase") is a type of THING
      must be one of {ALL (note "All phases"),
                              STD (note "Standing"),
                              PBT (note "Pushback/Towing"),
                              TXI (note "Taxi"),
                              TOF (note "TakeOff"),
                              ICL (note "Initial climb"),
                              ENR (note "En-route (Cruise)"),
                              APR (note "Approach"),
                              LDG (note "Landing")}.


 //-------------------------------------------------------------------------
-------------------------
 //-- Additional properties required
 //-------------------------------------------------------------------------
-------------------------

 //-- For HAZARD
 eventPhase (note "The event phase") describes HAZARD with values of
type Phase.
 severityClassification (note "The severity classification of the
hazard") describes HAZARD with values of type Severity. // Required
because the original "severity" property of HAZARD is float [0,1]
 minimumRequiredDal (note "The minimal DAL required for such a
hazard") describes HAZARD with values of type DesignAssuranceLevel.
 classificationJustification (note "Justification of severity
classification") describes HAZARD with values of type string.
 verificationMethod (note "The verification method used to verify that
the hazard has been mitigated") describes HAZARD with values of type
VerificationMethod.
 affects (note "The system affected by the hazard") describes HAZARD
with values of type SYSTEM.
```

## FHA Instance data

```
//-- Functions

 Backup-power (note "Provide backup power to the Flight Data
Recorder") is a FUNCTION
      with identifier "Backup-power"
      with description "Provide backup power to the Flight Data
Recorder".


 //-- The system and components involved, and their functions

 Flight-data-recorder (note "The flight data recorder") is a SYSTEM
      with identifier "Flight-data-recorder".


 Recorder-independent-power-supply (note "Recorder independent power
supply") is a SYSTEM
      with identifier "Recorder-independent-power-supply"
      with function Backup-power.


//-- Hazards

Hzrd-A1 is a HAZARD
      with identifier "Hzrd-A1"
      with description "Loss of ability to provide backup power"
      with eventPhase ALL
      with H:effect "Unable to record aircraft state and performance
parameters"
      with severityClassification Minor
      with minimumRequiredDal LevelD
      with classificationJustification "TSO-C155b section 3.b(2)
classifies this as a minor failure condition"
      with verificationMethod FHA
      with H:source Recorder-independent-power-supply
      with affects Flight-data-recorder.


 Hzrd-B1 is a HAZARD
      with identifier "Hzrd-B1"
      with description "Backup power provided when not required"  with
eventPhase ALL
      with H:effect "Erroneously provided power may result in the
inability to record aircraft state and performance parameters"
```

```
      with severityClassification Minor
      with minimumRequiredDal LevelD
      with classificationJustification "TSO-C155b section 3.b(1)
classifies this as a minor failure condition"
      with verificationMethod FHA
      with H:source Recorder-independent-power-supply
      with affects Flight-data-recorder.
```

## Requirements Ontology

```
//-- Different types of requirements
 SystemLevelRequirement (note "System Level Requirement") is a type of
REQUIREMENT
      described by rd:satisfiedBy with values of type REQUIREMENT
      described by rd:source with values of type string. // Should we
enumerate this in the future?

 SoftwareHighLevelRequirement (note "Software High Level Requirement")
is a type of REQUIREMENT.
 SoftwareLowLevelRequirement (note "Software Low Level Requirement")
is a type of REQUIREMENT.



 //-- Other classes needed to express system design
 Correctness (note "Correctness of a REQUIREMENt (taken from Cameo
Class Properties)") is a type of THING.

 Completeness (note "Completeness of a REQUIREMENt (taken from Cameo
Class Properties)") is a type of THING.

 RequirementType (note "The type of requirements") is a type of THING
      must be one of {functionalRequirement,
                      designConstraint,
                      interfaceRequirement,
                      safetyRequirement,
                      performanceRequirement,
                      usabilityRequirement,
                      physicalRequirement
                      }.

 CertificationReference (note "Certification References used in a
System Design") is a type of THING
```

```
      must be one of {TSO-C124c (note "For Flight Data Recorder
Equipment"),
                              TSO-C155b (note "For Recorder Independent
Power Supply"),
                              ARINC-777-2 (note "For Recorder Independent
Power Supply"),
                              ED-112A (note "For Minimum Operational
Performance Specification For Crash Protected Airborne Recorder
Systems"),
                              DO-160G (note "For Environmental Conditions
and Test Procedures for Airborne Equipment"),
                              SAE-ARP-4761 (note "For Guidelines and
Methods for Conducting the Safety Assessment Process on Civil Airborne
System and Equipment"),
                              RTCA-DO178C (note "For Software
Considerations in Airborne Systems and Equipment Certification")
                              }.


 Parameter (note "The input or output to a SYSTEM") is a type of THING
      must be one of {MaintainanceRequired,
                              MeasuredTemperature,
                              LoadMeasurements,
                              MeasuredCurrent,
                              MeasuredVoltage,
                              ChargeControllerComm,
                              BHMComm,
                              SwitchCommand,
                              BatteryPower,
                              ReportFault,
                              BackupActive,
                              BateryPower,
                              SwitchCommand,
                              SupplyPowerToRecorder,
                              ReceivePowerFromAircraft
                              }.

 parValue describes Parameter with a single value of type Value.


 Value (note "The values that can be taken by parameters") is a type
of THING
      described by probability with a single value of type string,
      described by magnitude with a single value of type string.
```

```
 //-- Additional properties on RACK core classes
 architectureAllocation describes REQUIREMENT with values of type
THING.
 Rq:derivedRequirementIndicator (note "Use True for derived
requirement and false otherwise") describes REQUIREMENT with values of
type boolean.
 rationale describes REQUIREMENT with values of type string.
 requirementType describes REQUIREMENT with a single value of type
RequirementType.
 correctness describes REQUIREMENT with values of type Correctness.
 correctnessFailComments describes REQUIREMENT with values of type
string.
 completetness describes REQUIREMENT with values of type Completeness.
 completenessFailComments describes REQUIREMENT with values of type
string.
 verificationRationale describes REQUIREMENT with a single value of
type string.
 Rq:verificationMethod describes REQUIREMENT with a single value of
type VerificationMethod.


 input describes SYSTEM with values of type Parameter.
 output describes SYSTEM with values of type Parameter.
```

## Requirements Instance data

```
// -- RIPS System Requirements

RIPS-01 is a SystemLevelRequirement
  with identifier "RIPS-01"
  with title "RIPS-01 Backup Power Time"
  with description "When aircraft power to the recorder drops below 18
Vdc, the RIPS shall make DC power available for 10 minutes."
  with requirementType functionalRequirement
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.1".



RIPS-02 is a SystemLevelRequirement
```

```
  with identifier "RIPS-02"
  with title "RIPS-02 Backup Time Tolerance"
  with description "The tolerance on the time of 10 minutes output
shall be +/- 1 minute."
  with requirementType designConstraint
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.2.3".

RIPS-03 is a SystemLevelRequirement
  with identifier "RIPS-03"
  with title "RIPS-03 Recharge Timing"
  with description "From the time aircraft power greater than 22 Vdc
is available until the RIPS is capable of providing the full 10
minutes of power shall be no more than 15 minutes."
  with requirementType performanceRequirement
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.2.2".

RIPS-04 is a SystemLevelRequirement
  with identifier "RIPS-04"
  with title "RIPS-02 Backup Time Tolerance"
  with description "The RIPS shall have a replaceable battery"
  with requirementType usabilityRequirement
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false.

RIPS-05 is a SystemLevelRequirement
  with identifier "RIPS-05"
  with title "RIPS-05 Maintenance Discrete"
  with description "The RIPS shall have a 'Maintenance Required'
Standard Discrete Output."
  with requirementType interfaceRequirement
  with rd:satisfiedBy BHM-HLR-10
  with rationale "Discrete interface required by ARINC 777-2."
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.5.3".

RIPS-06 is a SystemLevelRequirement
  with title "RIPS-06 RIPS Active Discrete"
  with identifier "RIPS-06"
  with description "The RIPS shall have a 'RIPS Active' Standard
Discrete Output."
  with requirementType interfaceRequirement
```

```
  with rationale "Discrete interface required by ARINC 777-2."
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.5.1".


RIPS-07 is a SystemLevelRequirement
  with title "RIPS-07 No Fault Discrete"
  with identifier "RIPS-07"
  with description "The RIPS shall have a 'No Fault' Standard Discrete
Output."
  with requirementType interfaceRequirement
  with rationale "Discrete interface required by ARINC 777-2."
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.5.2".

RIPS-08 is a SystemLevelRequirement
  with identifier "RIPS-08"
  with title "RIPS-08 Maintenance Discrete Behavior"
  with description "The RIPS shall set the 'Maintenance Required'
discrete in the 'ground' state when the RIPS has determined that the
internal battery needs to be replaced."
  with requirementType functionalRequirement
  with rd:satisfiedBy BHM-HLR-18
  with rd:satisfiedBy BHM-HLR-01
  with rd:satisfiedBy BHM-HLR-02
  with rd:satisfiedBy BHM-HLR-04
  with rd:satisfiedBy BHM-HLR-20
  with rd:satisfiedBy BHM-HLR-22
  with rd:satisfiedBy BHM-HLR-23
  with rd:satisfiedBy BHM-HLR-05
  with rd:satisfiedBy BHM-HLR-06
  with rd:satisfiedBy BHM-HLR-07
  with rd:satisfiedBy BHM-HLR-08
  with rd:satisfiedBy BHM-HLR-09
  with rd:satisfiedBy BHM-HLR-19
  with rd:satisfiedBy BHM-HLR-21
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.5.3".

RIPS-09 is a SystemLevelRequirement
  with identifier "RIPS-09"
  with title "RIPS-09 RIPS Active Discrete Behavior"
```

```
  with description "The RIPS shall set the 'RIPS Active' discrete in
the 'ground' state when the RIPS is supplying power to the recorder."
  with requirementType functionalRequirement
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.5.1".

RIPS-10 is a SystemLevelRequirement
  with identifier "RIPS-10"
  with title "RIPS-10 No Fault Discrete Behavior"
  with description "The RIPS shall set the 'RIPS Active' discrete in
the 'ground' state when the RIPS has determined that it is able to
supply back-up power to the recorder for the duration specified in
RIPS-01 and that it has detected no internal faults or external wiring
faults."
  with requirementType functionalRequirement
  with rd:satisfiedBy BHM-HLR-15
  with rd:satisfiedBy BHM-HLR-16
  with rd:satisfiedBy BHM-HLR-17
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 3.5.2".

RIPS-11 is a SystemLevelRequirement
  with identifier "RIPS-11"
  with title "RIPS-11 Operating Temperature Range"
  with description "The RIPS shall operate from -15 degC to 55 degC."
  with requirementType designConstraint
  with rd:satisfiedBy BHM-HLR-03
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "DO-160G Section 4.5.2 and 4.5.4".

RIPS-12 is a SystemLevelRequirement
  with identifier "RIPS-12"
  with title "RIPS-12 RIPS Form Factor"
  with description "The RIPS shall form factor shall match the
dimensions defined in ARINC 777-2 Attachment 5 Figure 5-1."
  with requirementType physicalRequirement
  with architectureAllocation Recorder-independent-power-supply
  with rationale "Following the form factor defined in ARINC 777-2
allows the RIPS system to be installed in all aircraft locations
compatible with this standard."
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 2.2".
```

```
RIPS-13 is a SystemLevelRequirement
  with identifier "RIPS-13"
  with title "RIPS-13 Weight"
  with description "The RIPS shall weigh no more than 5 pounds."
  with requirementType designConstraint
  with architectureAllocation Recorder-independent-power-supply
  with rationale "Max weight specified in ARINC 777-2."
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 2.7".

RIPS-14 is a SystemLevelRequirement
  with identifier "RIPS-14"
  with title "RIPS-14 Connector Type"
  with description "The RIPS shall use D38999/20JC35P connectors."
  with requirementType physicalRequirement
  with architectureAllocation Recorder-independent-power-supply
  with rationale "Use the standard connectors suggested in ARINC 777-
2."
  with Rq:derivedRequirementIndicator false
  with rd:source "ARINC 777-2 Section 2.2".


RIPS-15 is a SystemLevelRequirement
  with identifier "RIPS-15"
  with title "RIPS-15 Battery Recharge Initiation"
  with description "When aircraft power is available, the RIPS shall
initiate a recharge of the internal battery when estimated remaining
charge is less than 35%."
  with requirementType designConstraint
  with rd:satisfiedBy BHM-HLR-12
  with rd:satisfiedBy BHM-HLR-11
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false.


RIPS-16 is a SystemLevelRequirement
  with identifier "RIPS-16"
  with title "RIPS-16 Battery Recharge Completion"
  with description "After a battery recharge has been initiated and
while aircraft power is available, the RIPS shall continue recharge
the internal battery until the estimated remaining charge is above
99%."
  with requirementType designConstraint
  with rd:satisfiedBy BHM-HLR-24
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false.
```

```
RIPS-17 is a SystemLevelRequirement
  with identifier "RIPS-17"
  with title "RIPS-17 Minimum Operating Pressure"
  with description "The RIPS shall operate down to a minimum operating
pressure of 57.18 kPa."
  with requirementType designConstraint
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "DO-160G Section 4".

 RIPS-18 is a SystemLevelRequirement
  with identifier "RIPS-18"
  with title "RIPS-18 Operating Temperature Variation"
  with description "The RIPS shall operate in an environment with a
maximum temperature rate of change of 2 degC per minute."
  with requirementType designConstraint
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "DO-160G Section 4".

 RIPS-50 is a SystemLevelRequirement
  with identifier "RIPS-50"
  with title "RIPS-50 Loss of Backup Power"
  with description "Loss of ability for the RIPS to provide backup
power to the Flight Data Recorder shall be considered a MINOR failure
condition."
  with requirementType safetyRequirement
  with Rq:mitigates Hzrd-A1
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "TSO C155b Section 3.b(2)".


 RIPS-51 is a SystemLevelRequirement
  with identifier "RIPS-51"
  with title "RIPS-51 Inadvertant Backup Power"
  with description "Backup power provided by the RIPS to the Flight
Data Recorder shall be considered a MINOR failure condition."
  with requirementType safetyRequirement
  with Rq:mitigates Hzrd-B1
  with architectureAllocation Recorder-independent-power-supply
  with Rq:derivedRequirementIndicator false
  with rd:source "TSO C155b Section 3.b(2)".

 // BHM High Level Requirements
```

```
 BHM-HLR-01 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-01"
      with title "BHM-HLR-01 Monitor Battery"
      with description "The Battery Health Monitor shall indicate
battery maintenance is required when the State-of-Health is less than
70% with a tolerance of +/- 1%."
      with requirementType functionalRequirement
      with Rq:mitigates Hzrd-A1
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-02 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-02"
      with title "BHM-HLR-02 Temperature Input"
      with description "The Battery Health Monitor shall receive
temperature as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-03 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-03"
      with title "BHM-HLR-03 Temperature Range"
      with description "The Battery Health Monitor shall estimate
state-of-charge for batteries operating from -15 degC to 55 degC."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-11.

BHM-HLR-04 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-04"
      with title "BHM-HLR-04 Battery Terminal Voltage Input"
      with description "The Battery Health Monitor shall receive
battery terminal voltage measurements as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-05 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-05"
      with title "BHM-HLR-05 Battery Terminal Voltage Range"
```

```
      with description "The Battery Health Monitor battery terminal
voltage measurement input shall allow for voltage measurements from 0
Vdc to positive 40 Vdc."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-06 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-06"
      with title "BHM-HLR-06 Battery Output Current Input"
      with description "The Battery Health Monitor shall receive
battery output current measurements as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-07 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-07"
      with title "BHM-HLR-07 Battery Output Current Range"
      with description "The Battery Health Monitor battery output
current measurement input shall allow for current ranges from 0 A to 3
A."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-08 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-08"
      with title "BHM-HLR-08 Time Input"
      with description "The Battery Health Monitor shall receive
operating time since last charge as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-09 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-09"
      with title "BHM-HLR-09 Time Range"
      with description "The Battery Health Monitor time measurement
input shall allow for time measurements from 0 seconds up to 3.6e+7
seconds."
      with requirementType designConstraint
```

```
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-10 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-10"
      with title "BHM-HLR-10 Maintenance Required Output"
      with description "The Battery Health Monitor shall provide
Maintenance Required state as an output."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-05.

BHM-HLR-11 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-11"
      with title "BHM-HLR-11 Charge Command Behavior"
      with description "The Battery Health Monitor shall set the charge
command output to the active state when the State-of-Charge is less
than or equal to 30% with a tolerance of +/- 3%."
      with requirementType functionalRequirement
      with Rq:mitigates Hzrd-A1
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-15.

BHM-HLR-12 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-12"
      with title "BHM-HLR-12 Charge Command"
      with description "The Battery Health Monitor shall provide a
charge command as an output."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-15.

BHM-HLR-13 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-13"
      with title "BHM-HLR-13 State-of-Health Neural Network"
      with description "The Battery Health Monitor shall implement a
neural network to compute the State-of-Health."
      with requirementType designConstraint
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-01."
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator true.
```

```
BHM-HLR-14 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-14"
      with title "BHM-HLR-14 State-of-Charge Neural Network"
      with description "The Battery Health Monitor shall implement a
neural network to compute the State-of-Charge."
      with requirementType designConstraint
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-01."
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator true.

BHM-HLR-15 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-15"
      with title "BHM-HLR-15 Battery Terminal Voltage Out of Range"
      with description "The Battery Health Monitor shall indicate a
failure if a voltage measurement is received outside of the range
defined in BHM-HLR-05."
      with requirementType functionalRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-10.

BHM-HLR-16 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-16"
      with title "BHM-HLR-16 Temperature Out of Range"
      with description "The Battery Health Monitor shall indicate a
failure if a temperature measurement is received outside of the range
defined in BHM-HLR-03."
      with requirementType functionalRequirement
      with Rq:mitigates Hzrd-A1
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-10.

BHM-HLR-17 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-17"
      with title "BHM-HLR-17 Battery Output Current Out of Range"
      with description "The Battery Health Monitor shall indicate a
failure if a current measurement is received outside of the range
defined in BHM-HLR-07."
      with requirementType functionalRequirement
      with Rq:mitigates Hzrd-A1
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-10.
```

```
BHM-HLR-18 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-18"
      with title "BHM-HLR-18 Cycle Count Input"
      with description "The Battery Health Monitor shall receive a
count of the charge-discharge cycles for the current battery."
      with requirementType interfaceRequirement
      with Rq:mitigates Hzrd-A1
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-19 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-19"
      with title "BHM-HLR-19 Cycle Count Input Range"
      with description "The Battery Health Monitor cycle count input
shall allow for values from 0 to 200 cycles."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-20 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-20"
      with title "BHM-HLR-20 Load Voltage Input"
      with description "The Battery Health Monitor shall receive
voltage measurements at the load as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-20 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-20"
      with title "BHM-HLR-20 Load Voltage Input"
      with description "The Battery Health Monitor shall receive
voltage measurements at the load as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-21 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-21"
      with title "BHM-HLR-21 Load Voltage Input Range"
```

```
      with description "The Battery Health Monitor load voltage
measurement input shall allow for voltage measurements from 0 Vdc to
positive 40 Vdc."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-22 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-22"
      with title "BHM-HLR-22 Load Current Input"
      with description "The Battery Health Monitor shall receive the
current measurements at the load as an input."
      with requirementType interfaceRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-23 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-23"
      with title "BHM-HLR-23 Load Current Input Range"
      with description "The Battery Health Monitor load current
measurement input shall allow for current ranges from 0 A to 3 A."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-08.

BHM-HLR-24 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-24"
      with title "BHM-HLR-24 Charge Command Complete"
      with description "The Battery Health Monitor shall set the charge
command output to the inactive state when the State-of-Charge is
greater than or equal to 99% +/- 1%."
      with requirementType functionalRequirement
      with architectureAllocation Battery-Health-Monitor
      with Rq:derivedRequirementIndicator false
      with Rq:satisfies RIPS-16.

BHM-HLR-25 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-25"
      with title "State-of-Health Training Range"
      with description "The Battery Health Monitor State-of-Health
Neural Network shall be trained with a training data set that contains
State-of-Health values from 0% to 100%."
      with requirementType designConstraint
```

```
      with architectureAllocation Battery-Health-Monitor
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-01."
      with Rq:derivedRequirementIndicator true.

BHM-HLR-26 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-26"
      with title "State-of-Health Training Gap Coverage"
      with description "The Battery Health Monitor State-of-Health
Neural Network shall be trained with a training data set that contains
State-of-Health values from 0% to 100%."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-01."
      with Rq:derivedRequirementIndicator true.

BHM-HLR-27 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-27"
      with title "State-of-Charge Training Range"
      with description "The Battery Health Monitor State-of-Charge
Neural Network shall be trained with a training data set that contains
State-of-Charge values from 0% to 100%."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-11."
      with Rq:derivedRequirementIndicator true.

BHM-HLR-28 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-28"
      with title "State-of-Charge Training Gap Coverage"
      with description "The Battery Health Monitor State-of-Charge
Neural Network shall be trained with a training data set that does not
contain a gap in the State-of-Charge values greater than 0.5% for the
range defined in BHM-HLR-27."
      with requirementType designConstraint
      with architectureAllocation Battery-Health-Monitor
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-11."
      with Rq:derivedRequirementIndicator true.


BHM-HLR-29 is a SoftwareHighLevelRequirement
      with identifier "BHM-HLR-29"
      with title "Battery Voltage Runtime Monitor"
```

```
      with description "The Battery Health Monitor shall indicate
battery maintenance is required if the measured battery voltage is
less than 17 Vdc."
      with requirementType functionalRequirement
      with Rq:mitigates Hzrd-A1
      with architectureAllocation Battery-Health-Monitor
      with rationale "This requirement was derived from implementation
decisions made to satisfy BHM-HLR-01."
      with Rq:derivedRequirementIndicator true
      with Rq:satisfies RIPS-08.
```