DOT/FAA/TC- 26/20

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

Towards Comprehensive Safety Assurance of a DAL A AI/ML-based Runway Alignment System using Overarching Properties

July 2026

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).

Technical Report Documentation Page

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

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT/FAA/TC- 26/20		

4. Title and Subtitle	5. Report Date
Towards Comprehensive Safety Assurance of a DAL A AI/ML-based Runway Alignment System using Overarching Properties	July 2026
	6. Performing Organization Code: ANG-E271

7. Author(s)	8. Performing Organization Report No.
Saswata Paul[1], Daniel Prince[2], Naresh Iyer[3], Liang Tang[1], Michael Durling[1], Mike Meiners[2], Baoluo Meng[1], Sarat Chandra Varanasi[1], Nikita Visnevski[1]	DOT/FAA/TC- 26/20

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
1) GE Aerospace Research 2) GE Aerospace 3) GE Vernova Advanced Research	11. Contract or Grant No.

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Federal Aviation Administration, Office of Senior Technical Experts 12650 N. Featherwood Drive, Suite 230, Houston, TX 77034	Final
	14. Sponsoring Agency Code: AIR-20

16. Abstract

This report details an Overarching Properties (OPs)-based approach for assuring the safety of Artificial Intelligence/Machine Learning (AI/ML)-based digital aerospace systems. To rigorously evaluate and enhance this approach, an AI-Assisted Autonomous Runway Alignment (AARA) system is introduced as a motivating use case, allowing for the identification and mitigation of potential safety risks through premise-based arguments. The study demonstrates how multi-level safety assessments can be conducted for AI/ML systems to pinpoint failure conditions inherent to AI/ML's nature. It also provides examples of how requirements can be formulated to address risks posed by black-box AI/ML components with unpredictable or uncontrollable behaviors. Comprehensive discussions cover various aspects of the OPs-based approach, including foreseeable operating conditions, development and training activities, evidence generation for premises, hybrid certification, design assurance, necessary assumptions, and a plan for OPs compliance. While currently focused on Artificial Neural Networks (ANNs) developed using supervised learning, the arguments may be adaptable to other AI techniques. The findings establish a strong foundation for applying OPs to AI/ML assurance, acknowledging the need for further work to ensure robustness and practicality across diverse criticality and autonomy spectrums.

17. Key Words	18. Distribution Statement
Safety Assurance AI/ML Runway Alignment System Overarching Properties	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)	20. Security Classif. (of this page)	21. No. of Pages	19. Security Classif. (of this report)
Unclassified	Unclassified	56	Unclassified

Contents

Figures

Tables

Acronyms

Acronym	Definition
AARA	AI-Assisted Autonomous Runway Alignment
AI/ML	Artificial Intelligence/Machine Learning
ANN	Artificial Neural Network
DAL	Design Assurance Level
FAA	Federal Aviation Administration
FHA	Functional Hazard Assessment
FMS	Flight Management System
FDAL	Function Development Assurance Level
OPRAs	OPs-Related Arguments
RPS	Runway Perception Subsystem
SADL	Semantic Application Design Language

Executive summary

Artificial Intelligence/Machine Learning (AI/ML) techniques can be useful for autonomous or semi-autonomous operation of aircraft under diverse circumstances. Such safety-critical applications of AI/ML warrant adequate safety assurance before they are certified for operation. Moreover, because of the varying degrees of opacity and complexity in the design of non-trivial AI/ML-based aerospace applications, safety assurance must holistically consider the low-level behavioral and design elements of AI/ML-based components with respect to system-level hazards. However, existing certification standards and guidelines may not be adequate to assure systems using AI/ML. We present an *Overarching Properties* (OPs)-based approach for developing safety arguments that can be used to claim that an AI/ML-based component will be safe when used in the context of a more complex aircraft system. As a motivating use case for the study, we introduce an AI-Assisted Autonomous Runway Alignment (AARA) System that uses Artificial Neural Networks (ANNs) to help align the aircraft nose to the runway center line after touchdown. We use the AARA to identify potential safety concerns and mitigate them through appropriate premises and assumptions in the safety arguments. The results show promise in the applicability of our OPs-based approach for the holistic safety assurance of AI/ML-based complex aerospace systems.

1 Introduction

Rapid advancements in Artificial Intelligence/Machine Learning (AI/ML) technologies are demonstrating their potential in a wide spectrum of applications in the aerospace industry. Such techniques can be useful for autonomous or semi-autonomous operation of aircraft under diverse circumstances. However, aircraft operations are safety-critical, and any unmitigated failure during operation may potentially lead to catastrophic consequences. Therefore, adequate safety assurance of such AI/ML-based aerospace applications is necessary before they are certified to operate in civil airspaces.

AI/ML-based software systems are highly-complex with varying degrees of opacity in their design. Moreover, their usage as components of larger cyber-physical aerospace systems involves complex interactions with other components and the environment, making it difficult to predict and account for the different types of uncertainties that are involved in their operation. Therefore, analyzing the behavior of AI/ML models in isolation is not sufficient for safety-critical aerospace applications. For comprehensive safety assurance, it is necessary to develop clear safety arguments that can holistically connect the low-level behavioral and design elements of AI/ML-based components, such as robustness, accuracy, and the data used for training and testing, to the system-level hazards that are identified through safety assessments performed according to established guidelines, and to the stakeholder's needs.

These challenges were addressed previously in Saswata et al. (2023) and Paul et al. (2023). We had presented a set of *structured assurance arguments*, using the Overarching Properties (OPs) paradigm (Dassault Systemes, n.d.), that can be used to argue for the safe application of an AI/ML-based component in an aerospace system. The OPs are being considered by the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA) as a foundation for developing an alternate means of compliance (MoC) for non-traditional software, such as AI/ML, that cannot be assured using existing guidelines and standards. The OPs are comprised of three properties—*Intent*, *Correctness*, and *Innocuity*. A product is considered to be safe if it can be shown that it possesses all three OPs. Using a Design Assurance Level (DAL) D use case, we had demonstrated the viability of our approach by developing safety arguments which sufficiently considered all safety risks (Paul, et al., 2024). However, the scope of our arguments was limited to the minor failure consequence of the use case and its fairly simple operational space. For better evaluation of our OPs-Related Arguments (OPRAs) for AI, we must consider a more complex use case with more serious consequences of failure. Therefore, in this report, we introduce a hypothetical AI-Assisted Autonomous Runway Alignment (AARA) use case to evaluate (and adjust, as applicable) our OPRAs for AI. For the purpose of research,

we assume that the AARA is classified as a DAL A application so that it can facilitate rigorous analysis of our OPRAs for AI. Using the AARA, we improve the OPRAs for AI and present important observations that can help apply the OPs to other AI/ML use cases in the future.

2 New use case selection

2.1 Candidate use cases

For Phase 3 of the research, we considered the following candidate use cases:

1. vSLAM (Visual Simultaneous Location and Mapping) – A perception model used to analyze camera images in real time to determine the position of the aircraft.
2. Battery Health Monitor for Avionics Backup Power – A model that analyzes battery data from sensors to switch on-board avionics from primary to backup power if the primary battery is detected to fail.
3. Battery Health Monitor for Hybrid Electric Propulsion – A model that analyzes battery data from sensors to switch propulsion from an electric propulsion system to a fuel-based propulsion system if the electric propulsion system's battery is detected to fail.
4. Runway Alignment for Autonomous Aircraft Landing – A controller that uses the outputs from a perception model to align the aircraft to the runway center after touchdown during landing.

2.2 Use case evaluation matrix

We evaluated the different use cases based on a set of specific criteria to determine the most appropriate option for Phase 3. Each criterion was assigned a weight based on its relevance to the project objectives – 9 for high relevance, 5 for medium relevance, 1 for some relevance, and 0 for no relevance. The criteria are:

- Use for both EASA Level 2 and 3 of Autonomy (weight 9)
- Safety Criticality (DAL) (weight 1)
- Existing Requirements Maturity (weight 5)
- Existing Safety Analysis Maturity (weight 5)
- Existing Verification Evidence Maturity (weight 0)
- Existing AI Implementation (weight 9)

- Complexity of AI paradigm (weight 9)
- FAA Commercial Interest (weight 9)
- Publishability (weight 9)

To decide the score for each candidate use case, we designated a score to the possible values of each criterion. Then, each candidate use case was assigned an appropriate score for each criterion based on which value it satisfied. The scoring options and final scores for each candidate (calculated by multiplying the score of each candidate for each criterion and adding them) are shown in Figure 1.

Evaluation Criteria	UC#1	UC#2	UC#3	UC#4	9(High) 5(Med) 1(Low) and 0(NA)	UC#1 Score	UC#2 Score	UC#3 Score	UC#4 Score
	vSLAM	BHM For Avionics Backup Power	BHM for Electrified Propulsion	Autonomous Landing Runway Alignment	(Weights of the Different Criteria)	vSLAM	BHM For Avionics Backup Power	BHM for Electrified Propulsion	Autonomous Landing Runway Alignment
Use for both EASA Level 2 and 3 of Autonomy	0	0	9	9	9	0	0	81	81
Safety Criticality (DAL)	9	5	9	9	1	9	5	9	9
Existing Requirements Maturity	0	9	9	1	5	0	45	45	5
Existing Safety Analysis Maturity	1	1	1	1	5	5	5	5	5
Existing Verification Evidence Maturity	0	0	0	0	0	0	0	0	0
Existing AI Implementation	9	5	5	9	9	81	45	45	81
Complexity of AI paradigm	9	0	0	9	9	81	0	0	81
FAA Commercial Interest	0	9	9	9	9	0	81	81	81
Publishability	0	9	5	9	9	0	81	45	81
					Total =	176	262	311	424
Score Options (For the Different Use Cases)									
Use for EASA LoA: None (0) Only L1 (0) Only L2 (0) Only L3 (0) Both L2 and L3 (9)	**Safety Criticality:** No Safety Effect (0) Minor (1) Major (5) Hazardous (9) Catastrophic (9)	**Existing Requirements Maturity Levels:** No ConOps (0) ConOps (1) Systems Reqs (5) HLRs (9) LLRs (9)	**Existing safety Analysis Maturity Levels:** None (1) FHA (9) PSSA/SSA (9)	**Existing Verification Evidence Maturity Levels:** None (0) Some (5) Complete (9)	**Complexity of the AI model:** GOFAI (0) Simple NN (0) DNN (0) Perception (9) Transformer-based (9)	**Existing AI Implementation:** Nothing (0) Only data (5) Only model (0) Both data and model (9)	**FAA Commercial Interest:** Yes (9) No (0)	**Publishability:** Yes (9) Some restrictions (5) No (0)	

Figure 1: Candidate scores based on evaluation criteria

Based on the scores, we selected the Autonomous Runway Alignment use case for Phase 3 of the project. It stood out from the other use cases in terms of the complexity of the AI model, existing implementation, and publishability.

3 Artificial Intelligence-Assisted Autonomous Runway Alignment System

In the future, AI-based intelligent fight systems may be able to land aircraft in extraordinary circumstances. Such capabilities may be needed as a backup for circumstances arising due to pilot incapacitation, as in the case of Helios Airways Flight 522 (14th August 2005), or due to the need for enhanced pilot assistance. Autonomous landing involves several aspects, including final approach, final alignment to the runway, touchdown, and taxiing. One of the most important tasks after touchdown is to align the nose of the aircraft towards the center of the runway while

coming to a controlled stop. In Phase 3, our selected use case focused on autonomously aligning the aircraft to the center of the runway after touchdown (Figure 2).

Figure 2: Autonomous runway alignment using camera images

3.1 Concept of operations

The summary of the Concept of operations (ConOps) for the Phase 3 use case is as follows:

- There is a camera mounted on the aircraft that takes images of the runway.
- There is an AI model that can use the camera images to detect lateral distance and heading with respect to the center line of the runway.
- A Flight Management System (FMS) + Autopilot controls the aircraft based on the outputs of the AI model.

3.2 Level 2 and Level 3 autonomy scenarios

We assume two automation scenarios for the use case:

- EASA Level 2 (human can supersede)
    - The FMS + Autopilot can be overridden by a human
- EASA Level 3B (no human)
    - The FMS +Autopilot is certified for fully autonomous landing capabilities

4 Proposed AARA system architecture

We constructed a high-level model for the AARA in Cameo (Dassault Systemes, n.d.) to illustrate how the components interact with each other (Figure 3).

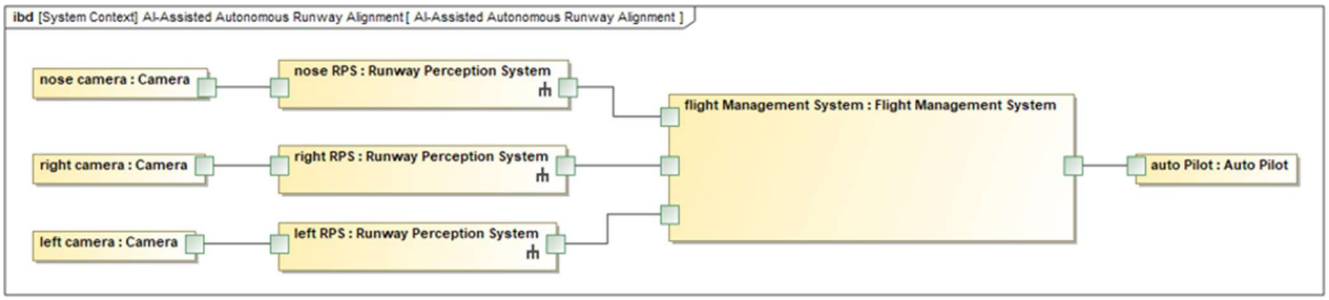

Figure 3: AARA system decomposition

The proposed architecture of the AARA consists of the following:

- Three cameras mounted at the nose, left wing, and right wing respectively.
- Three runway perception subsystems (RPSs) for redundancy, each connected to a different camera. Each RPS processes images from its respective camera as inputs to track the aircraft state via two parameters.
    - *Crosstrack position* (d): the lateral position of the aircraft relative to the runway's centerline.
    - *Heading angle* (θ): the angle made by the aircraft's longitudinal axis with the runway's centerline.

  Each RPS unit consists of a pre-trained Artificial Neural Network (ANN) and an input filter that filters any off-nominal inputs. The camera units are responsible for any pre-processing needed to make the data consumable by the RPS units. For every pre-processed nominal input, the ANN in each RPS computes a value of d and θ.
- A Flight Management System (FMS) module that takes the outputs of the three RPS units, makes a decision about the lateral correction parameters, and sends commands to the autopilot.
- An autopilot module that takes commands from the FMS and executes them.

There can be two levels of autonomy for the AARA—(1) there is a human to supervise and override the autopilot if needed, and (2) there is no human to supervise or override the autopilot (EASA, 2023). In both cases, a failure of the AARA can have catastrophic consequences. Therefore, we assume that it has a DAL A severity classification.

Next, to assess how the behaviors from the AI model would be impacted by external systems and how the AI model would impact other external systems, a high-level functional flow was

constructed, as shown in Figure 4. In this functional flow, the Runway Perception System is responsible for receiving image data from the camera and computing lateral correction data for the FMS to aggregate with other sensor information to produce lateral steering commands. While automated landing is not a current use case for FMSs, it is a potential future use case that allows an AI model to contribute to an aircraft level safety-critical function. In this use case, it is expected that the AI model would have catastrophic failure conditions and could be used in either EASA Level 2 or Level 3 autonomy scenarios.

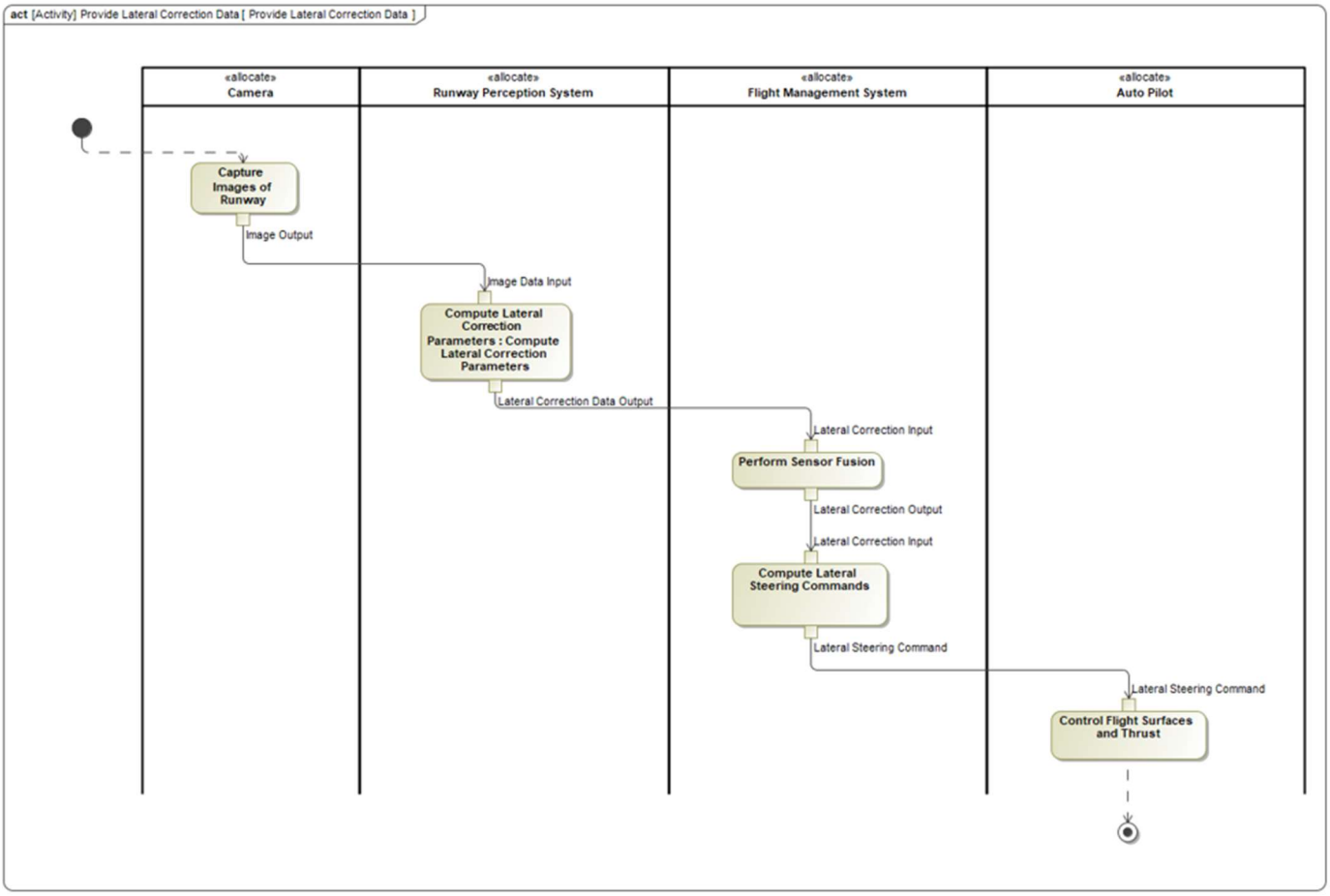

Figure 4: High-level functional flow for each RPS

5 Data and model

5.1 Data

The data for the perception-based model is publicly available from Stanford at https://purl.stanford.edu/zz143mb4347 (Katz, et al., 2021). It was collected using the X Plane 11

simulator platform using the simulated model of a Cessna 208B Grand Caravan aircraft moving along the simulated runway 04 of Grant County International Airport. The intent of the data collectors was to train a model that can output the following parameters that can then be used to determine necessary controls (by the FMS in our use case) to keep the aircraft nose aligned to the center of the runway[1]:

- The crosstrack position 'd' refers to the lateral position of the aircraft relative to the centerline of the runway.
- The heading angle 'θ' refers to the orientation angle of the aircraft relative to the centerline of the runway.

Images were taken at the rate of one frame per second from a simulated camera mounted on the right wing of the aircraft (Figure 5). The datasets were collected and segmented for ambient conditions – morning, afternoon, night, and overcast and the datasets were split into training, validation, and testing data for designing and evaluating models.

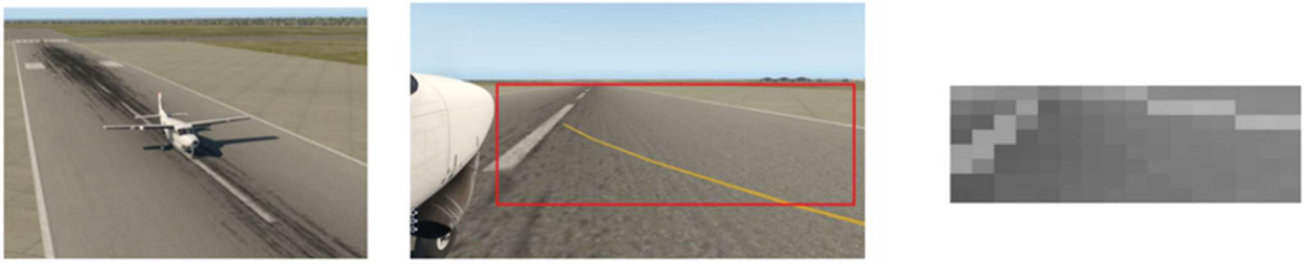

Figure 5: Sample of images pre-processed to simplify model

5.2 Model

The data was used to create a simple neural network model for predicting the crosstrack and heading of the aircraft with respect to the runway center line. The model creators wanted to evaluate the model using the Marabou tool (Katz, et al., 2019), so they processed the data in the following manner to keep the model simple (Julian, Lee, & Kochenderfer, 2020):

1. Crop out the sky and airplane nose
2. Resize image to 128 × 256 and convert to grayscale

[1] The runway center is defined as d = 0 m, and the runway heading angle i defined as θ = 0°.

3. Down-sample image by splitting image into 128 16×16 boxes, averaging the 16 brightest pixels within each box, resulting in an 8 × 16 image
4. Bias all pixel values so that the average value is 0.5 (pixel values range from 0 to 1)

Multiple models were created and made available by the model creators. The models were created using the 8x16 images collected under various ambient conditions, including afternoon, morning, overcast, and night, by training neural networks for 200 epochs. For example, a dedicated model for predictions on inputs collected during 'afternoon' was built using only the afternoon training data. A similar, 5th model was built by joining data from all ambient conditions into a single training dataset, referred to as the 'ALL' model (Figure 6). The 'ALL' model is the one we consider for our OPs study. The architecture of the neural network model, as showing in Figure 6, is comprised of 4 layers with 128, 16, 8, and 8 neurons respectively in those layers, followed by the output layer comprisd of 2 outputs, the crosstrack position 'd' and the heading angle q. All models were built in Pytorch and the cumulative, average mean-squared loss of the 2 outputs was minimized as a scalar loss while training the network. Table 1 shows the number of training and testing image-samples that was used to train, validate, and test each of the 5 neural network models.

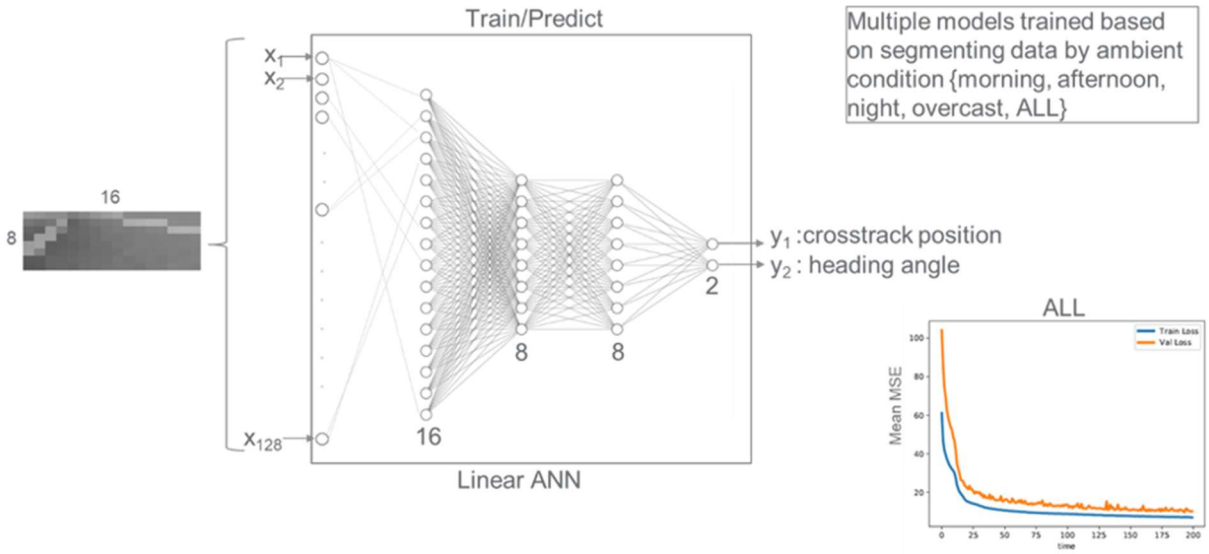

Figure 6: "ALL" model

Table 1: Training and testing data for different models

#SAMPLES v/s condition	MORNING	AFTERNOON	NIGHT	OVERCAST	ALL
TRAINING	44786	44879	45118	44544	179327

TESTING	6588	6595	6652	6561	26396

Table 2 shows the performance of each of the 5 models when applied to test data samples collected during different ambient conditions. As can be seen, models trained using data obtained from specific ambient conditions see a drop in performance when tested on other conditions, especially on samples collected during 'night.' The 'ALL' model that is trained using all data samples, on the other hand, suffers the least degradation in performance when tested on the night-samples. Additionally, the 'ALL' model also shows prediction performance for the other ambient conditions that are comparable to counterpart models trained and tested on the same ambient condition. This exemplifies the ability of the 'ALL' model to generalize across diverse ambient conditions, even if its performance for the 'night-samples' is an order worse.

Table 2: Performance of different models

Trained On	**Tested On**	**Cumulative MSE Loss**
Afternoon	Afternoon	3.2664886
Afternoon	Morning	3.912122
Afternoon	Overcast	3.2656322
Afternoon	Night	68.90312
Morning	Afternoon	4.743977
Morning	Morning	2.320454
Morning	Overcast	2.9370854
Morning	Night	103.39674
Night	Afternoon	71.07677
Night	Morning	69.61213
Night	Overcast	58.088726
Night	Night	21.586082
Overcast	Afternoon	3.8760629
Overcast	Morning	2.6205828
Overcast	Overcast	1.1719333
Overcast	Night	78.46656
ALL	Afternoon	3.7287166
ALL	Morning	3.2273734
ALL	Overcast	2.7260349
ALL	Night	15.74465

6 Hazard assessment for AARA use case

6.1 Purpose and scope of FHA

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

The selected use case is the design of an AARA system, where a new RPS has been added. The RPS makes use of a neural network to compute two outputs d and θ that are used downstream as lateral corrections parameters by the aircraft control systems (the FMS and the autopilot) to guide an aircraft to the correct path in alignment with a runway.

The FHA is intended to provide identification of RPS 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 driven from the failure condition classification (i.e., hazard classification), as well as the Function Development Assurance Level (FDAL) for the RPS 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).

6.2 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 alphabetic character represents the specific hazard being investigated (described below). The numeral is a sequential number quantifying the number of hazards of that specific type associated with that function Two hazard types are considered for each function being investigated, loss of function and undetected malfunction.

4. **Loss of function**: used to identify 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 system**: 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** (severity): 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** (severity) **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.)

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). 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. VFR: From point of VFR pattern entry or 1000 feet above the runway elevation.

Flight Phase Abbreviation	Flight Phase Name	Definition
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.

6.3 Guidance

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

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

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.

6.4 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, includes 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 RPS. 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.

6.5 Failure conditions of AARA and RPS

The safety assessment identified a hierarchy of failure conditions arising from both faulty and expected behavior of the AARA and its components. In total, we identified the following failure conditions at the different levels (Table 4, Table 5):

- AARA-Level
    - AARA-FC-01 Loss of Lateral Steering
    - AARA-FC-02 Incorrect Lateral Steering
- RPS-Level
    - RPS-FC-01 Loss Lateral Correction Data
    - RPS-FC-02 Hazardously Misleading Lateral Correction Parameter
- RPS ANN-Level
    - ANN-FC-1.1 ANN Robustness Failure
    - ANN-FC-1.2 ANN Out-of-Distribution

It should be noted that these failure conditions are representative of how there can be failure conditions at different levels and the list provided above is not exhaustive. There are other unidentified failure conditions that we have not specified as it is outside the scope of this project.

Table 4: Failures at AARA and RPS levels

#	Id	Name	Documentation	Function	Effect On System	Flight Phase	Severity	Severity Justification	Min Required DAL	Verification Method
1	AARA -FC-01	Loss of Lateral Steering	Loss of Lateral Steering	Provide Lateral Steering	Loss of ability for the aircraft to perform safe landing	Landing	Catastrophic	Loss of ability for the aircraft to perform safe landing	A	PSSA/SSA
2	AARA -FC-02	Incorrect Lateral Steering	Incorrect Lateral Steering	Provide Lateral Steering	Loss of ability for the aircraft to perform safe landing	Landing	Catastrophic	Loss of ability for the aircraft to perform safe landing	A	PSSA/SSA
3	RPS-FC-01	Loss Lateral Correction Data	Loss of Lateral Correction Data from all redundant RPS/Camera systems.	Compute Lateral Correction Parameters(co ntext Runway Perception Sub-system) Loss of Lateral Steering	Flight Management System operating during landing phase may fail to determine lateral path alignment to the runway, resulting in an unsafe landing.	Landing	Catastrophic	Loss of Lateral Correction Data from all redundant RPS/Camera systems may result in loss of ability for the aircraft to perform safe landing.	A	PSSA/SSA
4	RPS-FC-02	Hazardous ly Misleadin g Lateral Correction Parameter	Hazardously Misleading Lateral Correction from the combined set of RPS/Camera systems	Compute Lateral Correction Parameters(co ntext Runway Perception Sub-system) Incorrect Lateral Steering	Flight Management System operating during landing phase may determine an incorrect lateral path relative to the runway, resulting in an unsafe landing.	Landing	Catastrophic	The determination of hazardously misleading lateral correction from the combined set of RPS/Camera systems may result in the loss of ability to perform safe landing.	A	PSSA/SSA

Table 5: Failures at RPS ANNlevel

1	ANN-FC-1.1	ANN Robustness Failure	Common environmental noise/perturbations not included in training data Scenario: An image recognition system incorrectly classifies images that are subtly distorted (e.g., through noise/tilt/stretching)	Robustness failure	ANN produces incorrect lateral correction parameters
2	ANN-FC-1.2	ANN Out-of-Distribution	Changes in the nature of input data over time (e.g. sensor updates/calibration, processor/unit changes).	Out-of-distribution	Input filter filters out out-of-distribution input data and ANN fails to produce an output

7 Requirements for AARA use case

Figure 7 shows the requirement hierarchy for the AARA system.

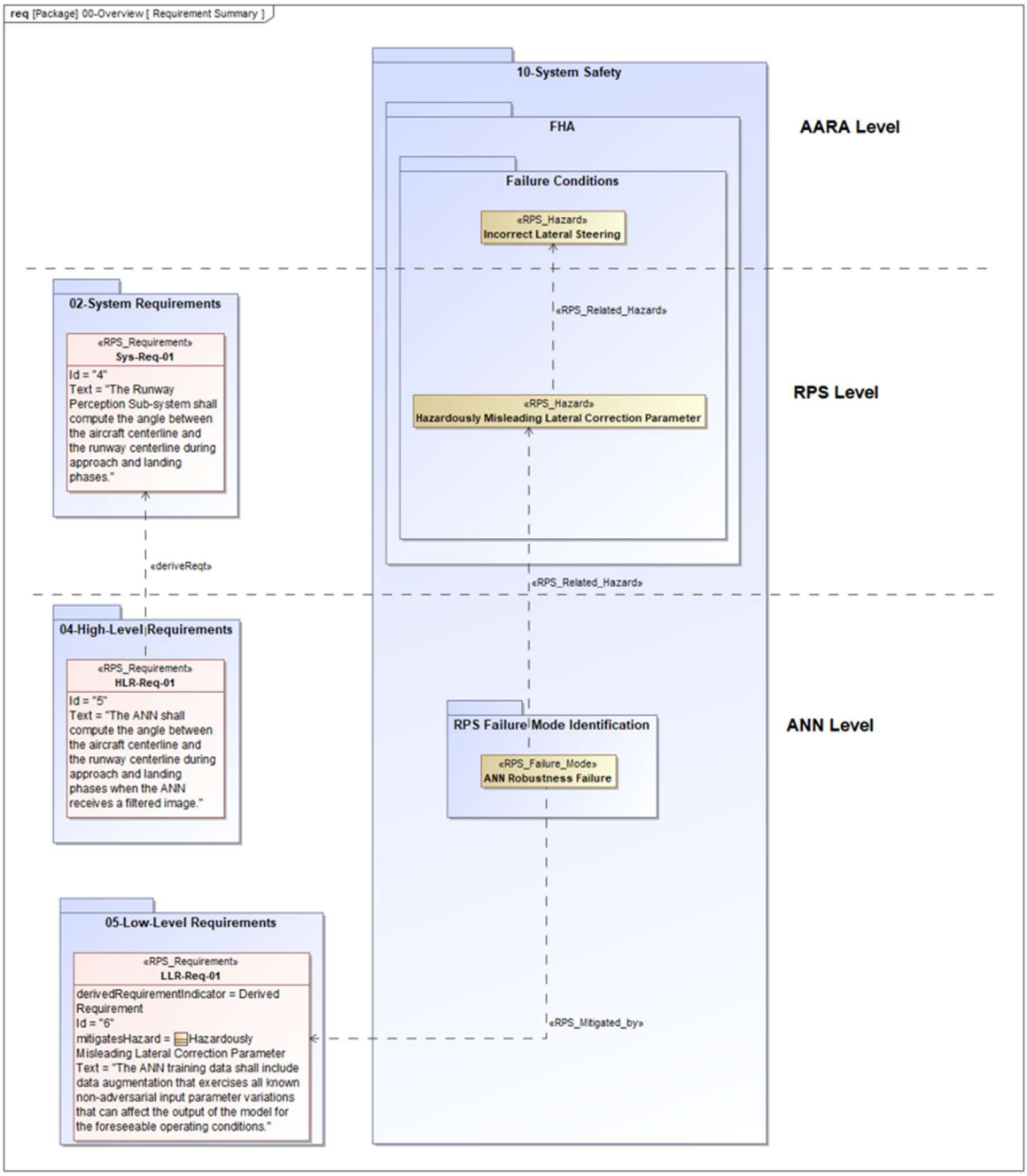

Figure 7: Requirement hierarchy for AARA system

The requirements identified at different levels are:

- System-Level (Table 6)

  Sys-Req-01 The RPS shall compute the angle between the aircraft centerline and the runway centerline during approach and landing phases.

- High-Level (Table 7)

  HLR-Req-01 The ANN shall compute the angle between the aircraft centerline and the runway centerline during approach and landing phases when the ANN receives a filtered image.

- Low-Level (Table 8)

  LLR-Req-01 The ANN training data shall include data augmentation that exercises all known non-adversarial input parameter variations that can affect the output of the model for the foreseeable operating conditions.

In addition, for claiming the possession of Innocuity for the RPS, we added the following requirement on the FMS, which is a downstream component to the RPS (Table 9):

- Downstream (Table 9)

  FMS-Sys-01 The FMS shall announce a lateral position determination failure, when lateral correction parameters from all three lateral correction inputs disagree for 10 consecutive cycles.

Table 6: System requirements

#	**Name**	**Text**
1	4 Sys-Req-01	The RPS shall compute the angle between the aircraft centerline and the runway centerline during approach and landing phases.

Table 7: High-level requirements

#	**Name**	**Text**
1	5 HLR-Req-01	The ANN shall compute the angle between the aircraft centerline and the runway centerline during approach and landing phases when the ANN receives a filtered image.

Table 8: Low-level requirements

#	Name	Text	Derived Requirement Indicator	Mitigates Hazard	Allocated To
1	6 LLR-Req-01	The ANN training data shall include data augmentation that exercises all known non-adversarial input parameter variations that can affect the output of the model for the foreseeable operating conditions.	Derived Requirement	Hazardously Misleading Lateral Correction Parameter	ANN

Table 9: Downstream requirement on FMS

#	Name	Text
1	7 FMS-Sys-01	The FMS shall announce a lateral position determination failure, when lateral correction parameters from all three lateral correction inputs disagree for 10 consecutive cycles.

8 Formalizations of requirements using SADL

We encoded the requirements using SADL (Figure 8). The entire set of SADL encoding of the FHA, system design, and other artifacts can be found in Appendix A. These can be used to ingest the data in the RACK tool for the synthesis of assurance cases when there is evidence available to support the OPRAs (Paul, et al., 2024).

```
Sys-Req-01 is a SystemLevelRequirement
  with identifier "Sys-Req-01"
  with description "The Runway Perception Sub-system shall compute the angle
between the aircraft centerline and the runway centerline during approach and
landing phases."
  with requirementType functionalRequirement
  with architectureAllocation RPS.


 HLR-Req-01 is a SoftwareHighLevelRequirement
  with identifier "HLR-Req-01"
  with description "The ANN shall compute the angle between the aircraft
centerline and the runway centerline during approach and landing phases when the
ANN receives a filtered image."
  with requirementType functionalRequirement
  with architectureAllocation RPS.



 LLR-Req-01 is a SoftwareLowLevelRequirement
  with identifier "LLR-Req-01"
  with description "The ANN training data shall include data augmentation that
exercises all known non-adversarial input parameter variations that can affect
the output of the model for the foreseeable operating conditions."
  with requirementType functionalRequirement
  with architectureAllocation RPS-ANN
  with rd:derivedRequirement "true".


 FMS-Sys-01 is a SystemLevelRequirement
  with identifier "FMS-Sys-01"
  with description "The FMS shall announce a lateral position determination
failure, when lateral correction parameters from all three lateral correction
inputs disagree for 10 consecutive cycles."
  with requirementType functionalRequirement
  with architectureAllocation FMS.
```

Figure 8: Requirements formalized in SADL

9 Updated OPRAs in Phase 3

9.1 Background

The three Overarching Properties are defined in Holloway (2019):

- **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.
- **Innocuity:** Any part of the implementation that is not required by the defined intended behavior has no unacceptable impact.

Our OPs-based approach involves two aspects—(1) a set of *structured premise-based arguments (OPRAs for AI)* that can be used to claim that an ANN-based product possesses each of the OPs; and (2) *a set of terminologies* (Figure 9) that attempt to normalize cross-domain verbiage and reduce ambiguity. We presented an initial set of arguments and terminologies in Saswata et al. (2023) and Paul et al. (2023). In light of the assurance challenges presented by the complex AARA use case, we have made some significant adjustments to the original arguments and terminologies to address the challenges.

The OPRAs are written in the Friendly Argument Notation (FAN) (Holloway M. C., 2020). Each argument consists of a main claim that an ANN-based software component (called a **NN-SW-COMPONENT**) possesses one of the OPs. The claim is justified by a set of premises, each of which must be independently verified and supported with appropriate evidence for the argument to be complete.

- **NN-MODEL** - A trained and pruned ANN model developed using supervized learning.
- **NN-MODEL-IMPLEMENTATION** - An implementation of the **NN-MODEL** deployed at the target platform (or an equivalent proxy of the target platform).
- **INPUT-FILTER** - A filter that can detect and filter any off-nominal value with respect to a given dataset.
- **INPUT-FILTER-IMPLEMENTATION** - An implementation of the **INPUT-FILTER** deployed at the target platform (or an equivalent proxy of the target platform).
- **NN-SW-COMPONENT** - A software component that consists of an **NN-MODEL** and an **INPUT-FILTER**.
- **NN-SW-COMPONENT-IMPLEMENTATION** - An implementation of an **NN-SW-COMPONENT** deployed at the target platform (or an equivalent proxy of the target platform). Consists of the **NN-MODEL-IMPLEMENTATION** and the **INPUT-FILTER-IMPLEMENTATION**.
- **DOWNSTREAM-COMPONENT** - A component that uses the outputs of a **NN-SW-COMPONENT**.
- **UPSTREAM-COMPONENT** - A component that sends inputs to a **NN-SW-COMPONENT**.
- **AI-SYSTEM** - A cyber-physical system that a **NN-SW-COMPONENT**, an **UPSTREAM-COMPONENT**, and a **DOWNSTREAM-COMPONENT** are parts of.
- **SOFTWARE-DEVELOPMENT-ACTIVITY** - The activity of developing a **NN-SW-COMPONENT** and its **NN-SW-COMPONENT-IMPLEMENTATION**.
- **SOFTWARE-DEVELOPMENT-DATA** - A dataset that is used in the **SOFTWARE-DEVELOPMENT-ACTIVITY**.
- **SOFTWARE-QUALIFICATION-ACTIVITY** - The activity of qualifying a **NN-SW-COMPONENT** and its **NN-SW-COMPONENT-IMPLEMENTATION** to verify that they possess Correctness.
- **SOFTWARE-QUALIFICATION-DATA** - A dataset that is used in the **SOFTWARE-QUALIFICATION-ACTIVITY** to qualify an **NN-MODEL** and an **INPUT-FILTER** with respect to the foreseeable operating conditions.
  - **NOMINAL-SOFTWARE-QUALIFICATION-DATA** - A subset of the **SOFTWARE-QUALIFICATION-DATA** that is nominal with respect to the foreseeable operating conditions (used to qualify the **NN-MODEL** and the **INPUT-FILTER**).
  - **OFF-NOMINAL-SOFTWARE-QUALIFICATION-DATA** - A subset of the **SOFTWARE-QUALIFICATION-DATA** that is off-nominal with respect to the foreseeable operating conditions (used to qualify the **INPUT-FILTER**).
- **QUANTITY-OF-INTEREST** - The quantity of interest that must be output by the **NN-SW-COMPONENT**.
- **OFF-NOMINAL-INPUT** - Any input to the **NN-SW-COMPONENT** that is off-nominal with respect to the **SOFTWARE-DEVELOPMENT-DATA**.
- **STAKEHOLDER-CONSTRAINTS** - Constraints on the desired behavior of the **NN-SW-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: resource limits, response time, scalablility, etc.
- **UNACCEPTABLE-IMPACT** - Any event or occurence that violates the safety of the **AI-SYSTEM**.
- **NN-SW-COMPONENT-FAILURE-CONDITION** - Any behavior of the **NN-SW-COMPONENT** or its components (the **NN-MODEL** and the **INPUT-FILTER**) that can directly or indirectly lead to an **UNACCEPTABLE-IMPACT**.
- **NN-SW-COMPONENT-SAFETY-ASSESSMENT** - A safety assessment performed for the **NN-SW-COMPONENT**.
- **NN-SW-COMPONENT-REQUIREMENTS** - A set of requirements written on the **NN-SW-COMPONENT**.

NN-SW-COMPONENT-DOWNSTREAM-REQUIREMENTS - A set of requirements written on the **DOWNSTREAM-COMPONENT**.

Figure 9: Glossary of terminologies used in our arguments

We assume a simplified architecture for a generic NN-SW-COMPONENT (Figure 10) that consists of an input filter and a pretrained ANN model, which is trained using supervised learning prior to deployment. The role of the input filter is to prevent any off-nominal input (with respect to the training data) from being sent to the model, thereby preventing the model from being operated outside of the scope of its training.

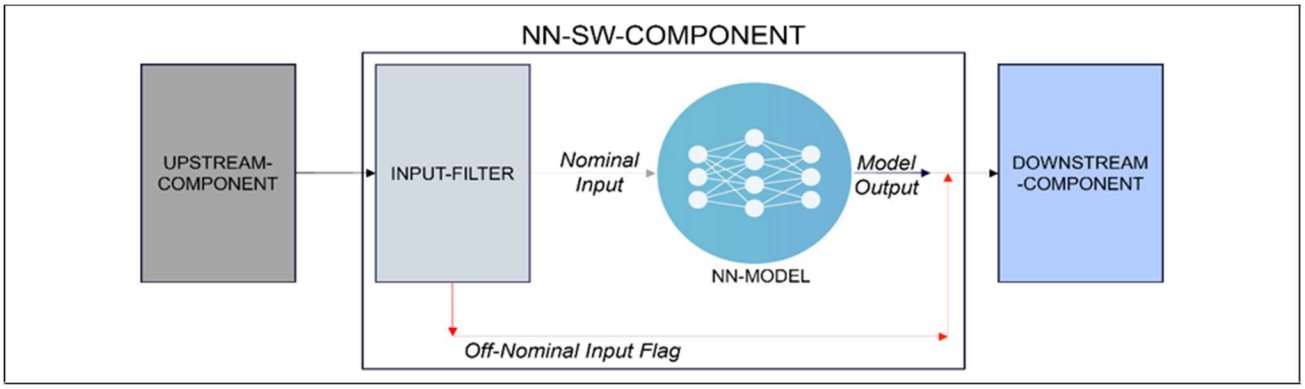

Figure 10: Generic architecture of ANN-based software component in our approach

The NN-SW-COMPONENT is incorporated into a more complex AI-SYSTEM via an UPSTREAM-COMPONENT and a DOWNSTREAM-COMPONENT, which respectively provide pre-processed inputs and receive outputs from the NN-SW-COMPONENT. There are two logically independent activities—the SOFTWARE-DEVELOPMENT -ACTIVITY involves any kind of design, optimization, training, testing, and validation needed to develop the NN-SW-COMPONENT; the SOFTWARE-QUALIFICATION-ACTIVITY verifies the final NN-SW-COMPONENT for certification. The two activities use *logically independent* sets of data. Note that the scope of our OPRAs is limited to the safety assurance of only the NN-SW-COMPONENT in the context of its usage as a part of the AI-SYSTEM.

9.2 The OPs-Related Arguments

In our OPs-based approach, the defined intended behavior of the NN-SW-COMPONENT is represented by the requirements on the NN-SW-COMPONENT. An UNACCEPTABLE-IMPACT is anything that can violate the safety assessment of the AI-SYSTEM. A NN-SW-COMPONENT-SAFETY-ASSESSMENT is performed for the NN-SW-COMPONENT (Figure 11) to identify all possible failure conditions of the NN-SW-COMPONENT, where a failure condition for the NN-SW-COMPONENT is defined as any behavior or failure of the NN-SW-COM PONENT that can contribute to an UNACCEPTABLE-IMPACT. This can result from an unintended behavior of the INPUT-FILTER or the NN-MODEL. In certain cases, even an "acceptable behavior" of the NN-MODEL can lead to unintended consequences (e.g., if the NN-MODEL

produces an incorrect output for a previously unseen nominal input at runtime). Such behavior of the NN-MODEL is also classified as a failure condition of the NN-SW-COMPONENT.

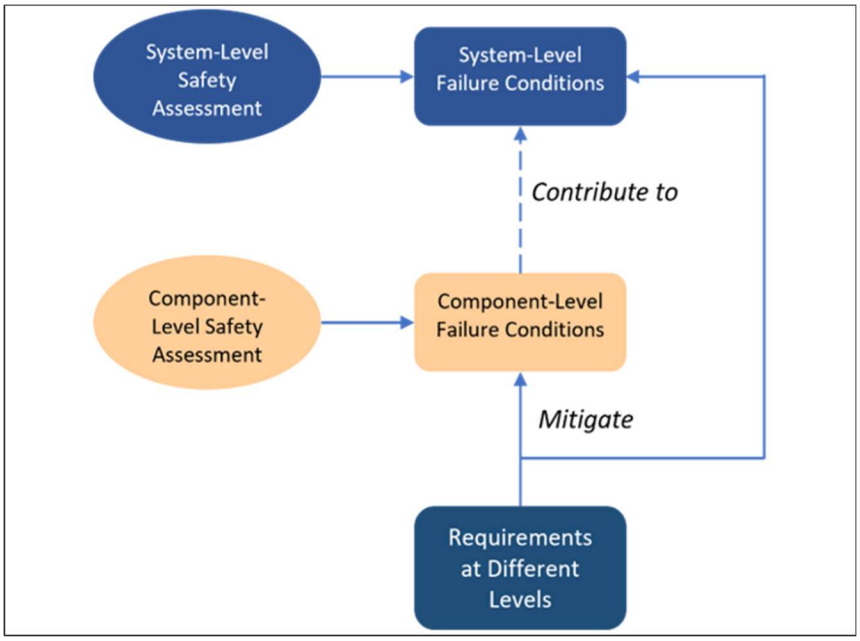

Figure 11: Safety assessment at different levels

The Intent argument (Figure 12) states that the defined intended behavior of the NN-SW-COMPONENT correctly and completely captures the desired behavior of the stakeholders. The premises for this argument impose necessary conditions on the requirements, the development data, and the software component. Essentially, the Intent claim ensures that the requirements are correct, complete, and verifiable and that the model is not used outside of the scope of the desired behavior expressed by the stakeholders. Furthermore, since for AI/ML the low-level requirements are analogous to the algorithm learned from the training data (FAA, 2024), the Intent argument also ensures that the development data will be apropos to the foreseeable

operational conditions. This constraint also ensures that the development data is devoid of any malicious or incorrect samples.

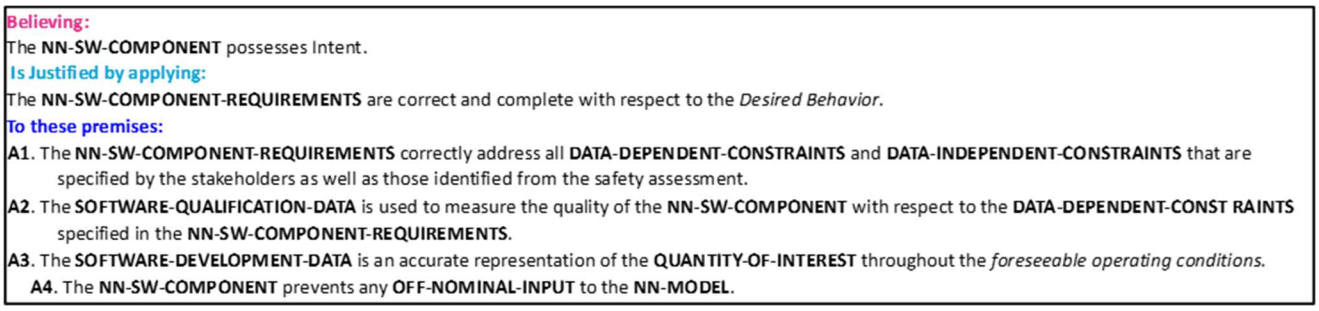

Figure 12: Intent Argument

The Correctness argument (Figure 13) is mainly concerned with showing that the NN-SW-COMPONENT has been designed and verified appropriately to satisfy all the constraints as set forth in the requirements. The Correctness premises impose necessary conditions on the training process, the data used for qualification, and the verification of the NN-SW-COMPONENT and its implementation using the qualification data. Simply put, the Correctness argument claims that all requirements have been adhered to. When designing system-specific Correctness arguments, the premises allow engineers to connect local properties of the NN-MODEL, such as robustness, accuracy, and stability, to the system-level safety assessment via the requirements (Paul, et al., 2024).

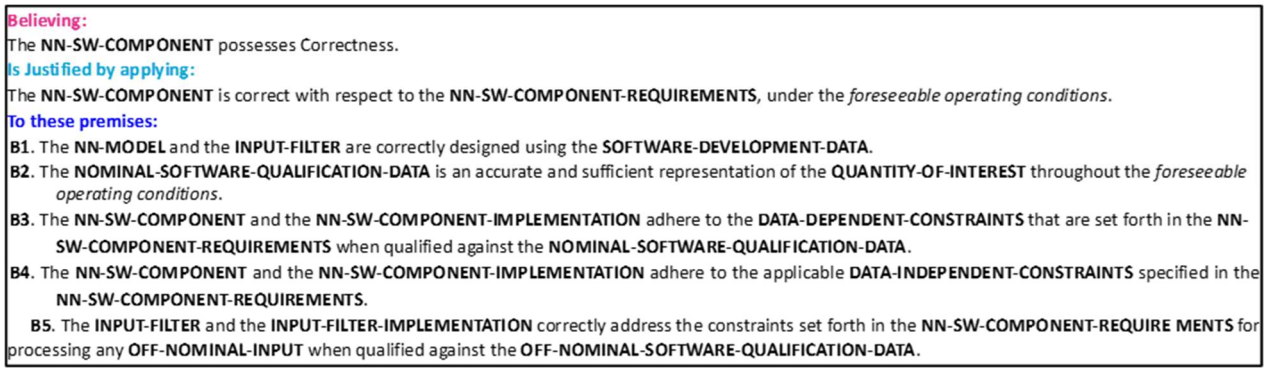

Figure 13: Correctness Argument

The property labeled Innocuity is complicated to address for a NN-SW-COMPONENT because it requires that any part of the component that is not warranted by the requirements shall have no unacceptable impact. Owing to the "black-box" nature of ANNs, it is not always possible to identify "parts of an ANN" and trace them back to requirements (FAA, 2024). Therefore, to keep the argument generically applicable, we took a conservative approach for the Innocuity premises. The Innocuity argument (Figure 14) relies on any potential failure condition of the NN-SW-

COMPONENT to be identified in the NN-SW-COMPONENT-SAFETY-ASSESSMENT[2] and then mitigated using requirements. However, for most practical purposes, the foreseeable operating conditions for a NN-SW-COMPONENT are expected to be unbounded. Therefore, there is always a chance that the NN-MODEL may encounter a nominal input at runtime that it has not been trained on and produce an incorrect[3]output. Such behavior of the NN-MODEL cannot be controlled or mitigated by NN-SW-COMPONENT-REQUIREMENTS and may need to be detected and mitigated in the DOWNSTREAM-COMPONENT. Therefore, we have imposed a constraint that any identified NN-SW-COMPONENT-FAILURE-CONDITION is mitigated either by the NN-SW-COMPONENT-REQUIREMENTS or the NN-SW-COMPONENT-DOWNSTREAM -REQUIREMENTS. Consequently, by contradiction, there cannot be any part of the NN-SW-COMPONENT that can lead to an UNACCEPTABLE-IMPACT.

Believing:
The **NN-SW-COMPONENT** possesses Innocuity.
Is Justified by applying:
Any part of the **NN-SW-COMPONENT** that is not required by the **NN-SW-COMPONENT-REQUIREMENTS** has no **UNACCEPTABLE-IMPACT**.
To these premises:
C1. Any potential **NN-SW-COMPONENT-FAILURE-CONDITION** is correctly identified in a **NN-SW-COMPONENT-SAFETY-ASSESSMENT** conducted for the **NN-SW-COMPONENT**.
C2. Any identified **NN-SW-COMPONENT-FAILURE-CONDITION** is mitigated by the **NN-SW-COMPONENT-REQUIREMENTS** or the **NN-SW-COMPONENT-DOWNSTREAM-REQUIREMENTS**.

Figure 14: Innocuity Argument

Every premise in the OPRAs presented above is a generic constraint that must be supported by evidence to claim that the arguments hold. Our approach does not prescribe any particular manner, tool, or procedure for demonstrating that a premise holds.

10 Context for OPs-related arguments

An argument is an attempt to convince others to believe a claim using reasoning (Holloway M. C., 2020). Reasoning occurs within a specific *context*, which essentially acts as a localized theory or model of the world where the reasoning is conducted (Giunchiglia, 1993). Without a well-defined context, the reasoning behind an argument cannot be justified. Therefore, the premise-based OPRAs for AI presented earlier need some additional context for them to be valid. This is supported by the OPs framework, which, in addition to the three OPs, includes *definitions, requisites, assumptions,* and *constraints* (Holloway C. M., 2019) that form the fundamental

[2] However, for non-deterministic systems operating in real-world environments, it is hard to mathematically prove that all possible failure conditions have been identified.

[3] Without more context or information beyond the training data, it is usually impossible to determine the correctness of the output produced for a previously unseen input at runtime, making it difficult to mitigate such issues only via the **NN-SW-COMPONENT-REQUIREMENTS**.

context for the OPs. These provisions in the original OPs framework enable engineers to lay down the necessary context when creating their product-specific OPs arguments.

Some generic context applicable to all ANN-based software components using our OPRAs are:

- **Hybrid certification** (Figure 15): Our approach assumes that when ANN-based software components are used as a part of a hybrid aerospace system that consists of other software and hardware components, the OPs will be used for the certification of the ANN-based software components. The traditional software and hardware components can either be certified using existing standards (e.g., DO-178 (RTCA), DO-254 (RTCA), and ARP-4754 (SAE, 2010)) or by using the OPs. This aligns with the FAA's guideline for using existing standards where appropriate and possible (FAA, 2024).

- **Assumptions on the** UPSTREAM-COMPONENT: If any pre-processing is needed for the sensor data to allow it to be well-formed for the NN-MODEL, then such pre-processing must be performed in the UPSTREAM-COMPONENT to ensure that inputs are treated the same way during the development activity, the qualification activity, and at runtime. The verification of the pre-processing technique and its implementation falls outside the scope of verification of the N N-SW-COMPONENT using the OPs. Instead, we assume that the pre-processing algorithms/techniques will be verified separately as a part of verification of the UPSTREAM-COMPONENT using the hybrid certification approach.

- **Assumptions on the** DOWNSTREAM-COMPONENT: Our Innocuity argument relies on NN-SW-COMPONENT-DOWNSTREAM-REQUIRE MENTS written on the DOWNSTREAM-COMPONENT. However, to show the possession of Innocuity, it is sufficient to show that the NN-SW-COMPONENT-DOWNSTREAM-REQUIREMENTS exist to sufficiently mitigate all NN-SW-COMPONENT-FAILURE-CONDITION in conjunction with the NN-SW-COMPONENT-REQUIREMENTS. We assume that the satisfaction of the NN-SW-COMPONENT-DOWNSTREA M-REQUIREMENTS will be verified separately as a part of the hybrid approach.

- **Plan to show possession of the OPs**: Our OPs premises only specify *what* needs to be shown to claim the possession of the OPs. The premises have been kept abstract and generic so that it is possible to adapt them to different ANN-based systems. The responsibility of ensuring that the premises hold falls with the engineers who must ensure that the appropriate processes and criteria are followed for specific products. This involves creating a DAL-appropriate plan to show how OPs possession will be demonstrated for a particular system with similar rigor of details that one would expect to see in a certification plan for traditional standards (e.g., a DO-178C PSAC (Paul, et al.,

2023) clearly states the DAL-based level of independence needed for satisfying certain objectives).

It is important to note that for a given **NN-SW-COMPONENT**, all three OPs must be analyzed within the same context. This restriction ensures that the claims of possession of Intent, Correctness, and Innocuity for a given **NN-SW-COMPONENT** are evaluated against the same set of information and are consistent with one another (Paul, et al., 2024).

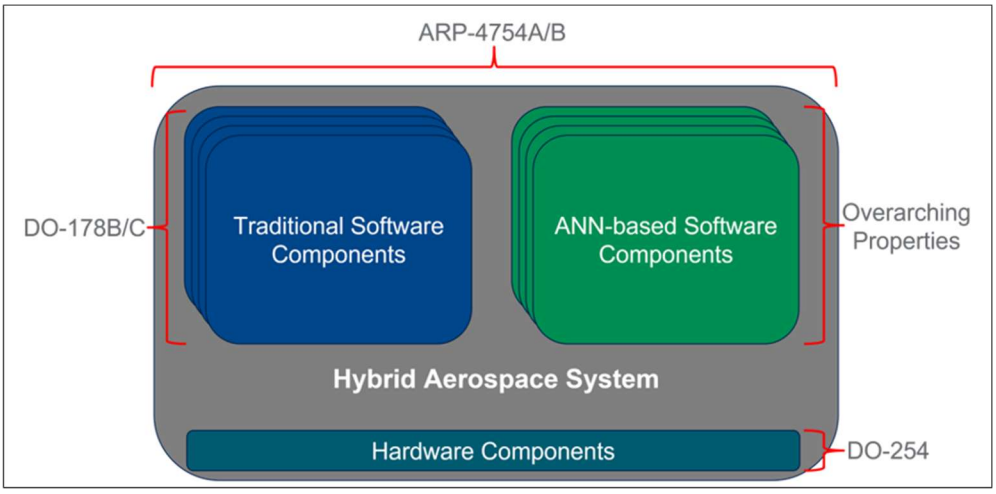

Figure 15: Hybrid certification approach

11 Supplementary information for OPs-related arguments

11.1 Specifying foreseeable operating conditions

The foreseeable operating conditions are defined as "*external and internal conditions in which the system is used, encompassing all known normal and abnormal conditions*" (Dassault Systemes, n.d.). For different agents involved in the **AI-SYSTEM** development and certification lifecycle, there will be different needs from the specification of the foreseeable operating conditions. Details of the foreseeable operating conditions are not only needed to design the ANN model, but to also write requirements on the **SOFTWARE-DEVELOPMENT-DATA** and the **SOFTWARE-QUALIFICATION-DATA**, and the appropriate error tolerances for different aspects of the **NN-SW-COMPONENT** (EASA, 2023). Therefore, we believe that it falls upon a hierarchy of stakeholders (which includes the customer, the internal stakeholders, the safety engineers, the system engineers, the AI engineers, the regulatory agency representatives, and others) to specify, audit, and agree on the foreseeable operating conditions at different levels of granularity to ensure that the desired behavior can be satisfied, the product can be effectively designed, and the final product can be verified.

11.2 Qualification and development activities

Both the **SOFTWARE-DEVELOPMENT-DATA** and the **SOFTWARE-QUALIFICATION-DATA** must be statistically related to the foreseeable operating conditions. As a consequence, they will also be statistically related to each other. Nevertheless, we believe that to ensure that the development and the qualification activities are not prone to unintended influence that can undermine their effectiveness, the two datasets must be logically independent and must not be shared across the two activities. Otherwise, if the **SOFTWARE-QUALIFICATION-ACTIVITY** is exposed to the **SOFTWARE-DEVELOPMENT-DATA**, then the activity may be biased to pass/fail the **NN-MODEL**. Similarly, if the **SOFTWARE-DEVELOPMENT-ACTIVITY** is aware of the **SOFTWARE-QUALIFICA TION-DATA**, then the **NN-MODEL** may be biased to perform well only on the **SOFTWARE-QUALIFICATION-DATA**[4]. Regardless of how the two datasets are collected and curated, we believe that as long as the premises are met and the development and qualification datasets and activities have logical separation, the risk of unintended influence will be low.

11.3 Premise decomposition and evidence generation

The argument premises presented in this paper are high-level generic premises that specify what must be done to show possession of each of the OPs. When applying these arguments to a specific system, the premises must be instantiated for the system to create system-specific arguments. In many cases, the premises may also need to be decomposed into system-specific sub-premises to specify how custom evidence can be used to support the system-specific arguments (EASA, 2023). This allows our OPRAs to be applicable for a wide variety of ANN-based systems

11.4 Design time assurance

One thing to note about our OPs-based approach is that it is primarily a *design time assurance* technique (Wasson, Kimberly; Holloway, Michael, 2022) that can be used to demonstrate that pre-defined safety objectives are satisfied. In many cases, design time assurance involves the analysis of an AI model at various stages of development, such as design, training, optimization, compiling, and inference (Wasson, Kimberly; Holloway, Michael, 2022). At each stage, a different version of the model is analyzed against some assurance criteria to ensure that the model meets its pre-defined safety objectives. However, our OPs approach simplifies that into the analysis of the model at two stages— first after training, pruning, and optimization, which is

[4] This is not an issue *iff* the **SOFTWARE-QUALIFICATION-DATA** completely captures the foreseeable operating conditions. But since the **SOFT WARE-QUALIFICATION-DATA** can only be *sufficiently representative* for continuous operating state spaces, this may have risks.

the NN-MODEL, and second when deployed at the target platform (or a suitable proxy), which is the NN-MODEL-IMPLEMENTATION. This is because we believe that the combination of our premises across the three OPs creates enough "logical safeguards" to ensure that verifying the NN-MODEL and the NN-MODEL-IMPLEMENTATION against the SOFTWARE-QUALIFICATION-DATA will be sufficient to effectively identify any shortcomings.

12 Influence of AARA on OPRAs

The analysis of the risks associated with the RPS NN-SW-COMPONENT opened up a lot of questions that needed to be considered in the updated OPs arguments. Some of the most prominent questions that influenced the set of updated arguments presented in Section III are:

- How should the arguments account for any data pre-processing that may be needed?
- How can NN-SW-COMPONENT-REQUIREMENTS be written for the NN-SW-COMPONENT?
- Given that ANN models are never expected to be 100% accurate, how should failure conditions of the NN-MODEL be defined to account for such behavior?
- How can an incorrect behavior of an RPS at runtime be mitigated if there is no human to supervise the system?

In this section, we discuss the impact of these questions on the final design of the OPRAs. The data pre-processing concerns resulted from the characteristics of the open-source RPS ANN model we used. The model requires the raw camera images to undergo a series of pre-processing steps (Julian, Lee, & Kochenderfer, 2020). However, pre-processing can often lead to drastic changes in the raw data from sensors (sensors being the cameras in our use case). Therefore, the potential effects of data pre-processing must be given equal consideration during development, and qualification. Additionally, it must be ensured that the runtime data receives exactly the same pre-processing treatment as the data used for development and qualification. Otherwise, any properties verified during qualification may cease to hold at runtime. To address these concerns, new assumptions on the upstream components were added to the OPs context to ensure consistent pre-processing of data at all stages of the NN-SW-COMPONENT's lifecycle.

One challenge with AI models is that unlike traditional systems, requirements for AI models cannot always be written in a way that allows them to be traced to higher-level requirements (FAA, 2024). Moreover, the black-box nature of AI models also makes it difficult to clearly trace requirements to specific "parts" of the models. This led to our choice of writing requirements for the NN-MODEL by identifying potential behavior that can impact the safety of the N N-SW-COMPONENT

and consequently the **AI-SYSTEM**. For the AARA, we first performed an RPS-level safety assessment to identify some failure conditions of the RPS **NN-MODEL** that could contribute to the identified failure conditions for the AARA and RPS (Table 5). Then, derived requirements[5] were written specifically for the RPS **NN-MODEL** to mitigate such failure conditions. To mitigate the failure condition ANN-FC-1.1 (robustness failure (Blood, Herbert, & Wayne, 2023)), the derived requirement LLR-Req-01 was written[6], which states that "*The ANN training data shall include data augmentation that exercises all known non-adversarial input parameter variations that can affect the output of the model for the foreseeable operating conditions.*"

AI models used in continuous unbounded state spaces are inherently inaccurate since there is always a chance that the models may produce an incorrect output for a previously unseen nominal input. This is a perfectly "acceptable" behavior of an AI model and not necessarily a "failure" but can contribute to failure conditions at higher levels. In the context of the AARA system, if the right RPS **NN-MODEL** produces an incorrect output at runtime, then that can lead to RPS-FC-02 and consequently AARA-FC-02 (Table 4). For this reason, we decided to classify any behavior of the **NN-MODEL** that can lead to a failure condition at a higher level as a failure condition at the ANN level. Other vulnerabilities of ANNs (e.g., robustness, performance bias, etc. (Blood, Herbert, & Wayne, 2023)) can also contribute to higher-level failure conditions and, therefore, are subsumed in this generic definition of ANN-level failure conditions.

The DAL A classification of the AARA introduced challenges in designing the Innocuity argument. Consider the case where there is only the right RPS and at runtime, the right RPS **NN-MODEL** produces an incorrect output for a nominal input. Since the RPS has no context beyond the **SOFTWARE-DEVELOPMENT-DATA** that was used for its development, it is not possible to catch such an issue within the RPS itself. However, if not mitigated, then such issues can contribute to higher-level failure conditions. Because of the catastrophic nature of the AARA failure conditions, we had to use three redundant RPS components so that the FMS may be able to catch such issues at runtime and take appropriate actions[7]. A requirement FMS-Sys-01 ("*The FMS shall announce a lateral position determination failure, when lateral correction parameters from all three lateral correction inputs disagree for 10 consecutive cycles*") was written on the FMS to address this. This influenced our Innocuity premises where we updated premise **C2** to include both **NN-SW-COMPONENT-REQUIREMENTS** and **NN-SW-COMPONENT-DOWNSTREAM-REQUIREMENTS** to mitigate all identified **NN-SW-COMPONENT-FAILURE-CONDITION**.

[5] As per DO-178C terminology, derived requirements are requirements that are not directly traceable to higher-level requirements.

[6] Analysis of the quality of this notional requirement, albeit important, is out of the scope of this paper.

[7] There are several ways to do this, but we chose redundancy as a simple notional example to illustrate the concept.

Overall, the AARA use case and its intricacies significantly influenced the OPRAs, premises, and context outlined in Section III, thereby facilitating better consideration of the safety risks associated with DAL A ANN-based applications. However, several challenging questions remain to be answered before we can rigorously validate the effectiveness of the arguments.

13 Related work

OPs applied to traditional aerospace systems: Daw and Beecher (2023) have applied OPs (Holloway C. M., 2019) and OPRAs (Wasson, Kimberly; Holloway, Michael, 2022) on an Auxiliary Power Unit. They have also applied a hybrid approach to OPs on a UAV collision avoidance system (Daw, Beecher, Holloway, & Graydon, 2021). Aiello et al. (2020) developed an assurance case based on OPs for a TQL1 AdaCore Code Generator. Graydon and Cronin (2021) retrospectively showed OPs possession of a geofence-restriction software. Durling et al. (2021) investigated the use of adaptive stress testing (AST) to generate OP-based certification evidence for airborne software.

OPs applied to AI-based aerospace systems: Späth et al. (2025) demonstrated OPs for three different ML systems with different application domains under different training processes and datasets, and used defeaters to assess uncertainty and doubt into the OPRAs. Luettig et al. (2024) presented a case study where the OPs were applied to create bespoke assure arguments for an automated peripheral detection system (ADIMA).

Failure conditions and requirements for AI/ML: Pereira et al. (2020) have identified different hazards (such as incorrect data definition, incomplete testing, etc.) that can affect safety during the AI/ML development lifecycle. Ahmat et al. (2023) conducted a systematic study of requirements engineering problems discovered from surveying several case studies and recognized the importance of eliciting requirements on training and verification data. Blood et al. (2023) defined a set of AI/ML failure modes that provides a good guidance for our component-level safety assessment in our OP-based approach. Späth et al (2024) presented an approach to detect potential unintended behavior of AI/ML systems by analyzing unseen points within the foreseeable operating conditions.

14 Conclusion

This report presented an Overarching Properties (OP)-based approach for the safety assurance of Artificial Intelligence/Machine Learning (AI/ML)-based digital aerospace systems. An AI-Assisted Autonomous Runway Alignment (AARA) system was introduced to evaluate and augment our OP-based approach to ensure that potential safety risks are sufficiently identified

and mitigated by the premise-based OPs-Related Arguments (OPRAs) for AI. We demonstrated how safety assessments can be performed at different levels for AI/ML-based systems to identify failure conditions that can be caused by the nature of AI/ML. We also showed with examples how requirements can be written to mitigate the risks caused by black-box AI/ML-based components whose behavior cannot be pre-determined and/or controlled. Finally, we provided detailed discussion on different aspects of our OP-based approach, such as the foreseeable operating conditions, the development and training activities, evidence generation to support premises, hybrid certification, design assurance, necessary assumptions, and a plan to show OPs compliance, that will enable users to get a good understanding of how to apply the techniques presented. One thing to note is that although the scope of our arguments is currently limited to ANNs designed using supervised learning, it may be possible to adapt them for other AI techniques with necessary modifications.

Although our approach provides a good foundation for the application of the OPs for the assurance of AI/ML-based systems, much work remains to be done to ensure that the approach is robust and practicable. One interesting direction of future work would be to apply the approach to a variety of use cases across different criticality and autonomy spectrums to identify potential logical gaps in the premises and close them by augmenting the premises. Other challenges include the investigation of pertinent questions such as:

- How can off-nominality for data in complex input spaces be defined for effective input filter design?
- How can failure probabilities of AI/ML models similar to failure probabilities of traditional sensors be quantified so that appropriate requirements can be written to mitigate such failures?
- Can the OP arguments incorporate some degree of design explainability for the AI/ML models?
- For complex operational state spaces, how can the representativeness of datasets be efficiently tested?
- What are robust approaches to detect AI/ML model failure conditions at runtime?

Examining these questions is crucial for advancing towards a comprehensive, provable, and reliable assurance strategy for AI/ML applications in safety-critical aerospace systems.

15 References

Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., & Grundy, J. (2023). Requirements engineering for artificial intelligence systems: A systematic mapping study. *Information and Software Technology, 158*, 107176.

Aiello, A., Comar, C., & Ruiz, J. (2020). An assurance case based on overarching properties for a TQL1 code generator. *ERTS2020*.

Blood, J. C., Herbert, N. W., & Wayne, M. R. (2023). Reliability Assurance for AI Systems. *2023 Annual Reliability and Maintainability Symposium (RAMS)*, (pp. 1–6).

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

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)*, (pp. 1–8).

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 (DASC)*, (pp. 1–10).

Durling, M., Herencia-Zapana, H., Meng, B., Meiners, M., Hochwarth, J., Visser, N., . . . Valapil, V. T. (2021). Certification Considerations for Adaptive Stress Testing of Airborne Software. *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, (pp. 1–9).

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

FAA. (2024, August). *Roadmap for Artificial Intelligence Safety Assurance, Version 1.* Federal Aviation Administration. Federal Aviation Administration.

Giunchiglia, F. (1993). Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine, 16*.

Graydon, M. S., & Cronin, J. D. (2021). *Retrospectively documenting satisfaction of the Overarching Properties: An exploratory prototype.* Tech. rep.

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

Holloway, M. C. (2020). The friendly argument notation (FAN).

Julian, K. D., Lee, R., & Kochenderfer, M. J. (2020). Validation of image-based neural network controllers through adaptive stress testing. *2020 IEEE 23rd international conference on intelligent transportation systems (ITSC)*, (pp. 1–7).

Katz, G., Huang, D. A., Ibeling, D., Julian, K., Lazarus, C., Lim, R., . . . others. (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, July 15-18, 2019, Proceedings, Part I 31*, (pp. 443–452).

Katz, S. M., Corso, A., Chinchali, S., Elhafsi, A., Sharma, A., Kochenderfer, M. J., & Pavone, M. (2021). NASA ULI Aircraft Taxi Dataset. Stanford Digital Repository. *NASA ULI Aircraft Taxi Dataset. Stanford Digital Repository*. Retrieved from https://purl.stanford.edu/zz143mb4347

Luettig, B., Akhiat, Y., & Daw, Z. (2024). ML meets aerospace: challenges of certifying airborne AI. *Frontiers in Aerospace Engineering, 3*, 1475139.

Paul, S., Alexander, C., Durling, M., Siu, K., Prince, D., Meng, B., . . . Stuart, D. (2023). Automated DO-178C Compliance Summary through Evidence Curation. *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, (pp. 1–10). doi:10.1109/DASC58513.2023.10311159

Paul, S., Iyer, N., Prince, D., Tang, L., Durling, M., Meiners, M., . . . Mandal, U. (2024). Assurance of AI/ML-Based Aerospace Systems Using Overarching Properties. *43rd Digital Avionics Systems Conference (DASC).* doi:10.1109/DASC62030.2024.10749102

Paul, S., Prince, D., Iyer, N., Durling, M., Visnevski, N., Meng, B., . . . Meiners, M. (2023). Towards the Certification of Neural Networks using Overarching Properties: An Avionics Case Study. *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, (pp. 1–10). doi:10.1109/DASC58513.2023.10311280

Pereira, A., & Thomas, C. (2020). Challenges of machine learning applied to safety-critical cyber-physical systems. *Machine Learning and Knowledge Extraction, 2*, 579–602.

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

RTCA. (n.d.). DO-254 Design Assurance Guidance for Airborne Electronic Hardware. *DO-254 Design Assurance Guidance for Airborne Electronic Hardware*.

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

Saswata, P., Prince, D., Iyer, N., Durling, M., Visnevski, N., & Meng, B. (2023). *Assurance of Machine Learning-Based Aerospace Systems: Towards an Overarching Properties-Driven Approach.* Tech. rep., United States. Department of Transportation. Federal Aviation Administration ….

Späth, H., & Daw, Z. (2024). Towards Detecting Unintended Behaviors in Machine Learning Algorithms. *43rd Digital Avionics Systems Conference (DASC).* doi:10.1109/DASC62030.2024.10749477

Späth, H., Varchev, T., Staudacher, S., Daw, Z., & Holloway, M. (2025). Arguing machine learning assurance for certification. *Software Engineering 2025–Companion Proceedings*, (pp. 10–18420).

Wasson, Kimberly; Holloway, Michael. (2022). An Introduction to Constructing and Assessing Overarching Properties Related Arguments (OPRAs): Version 1.0.

A SADL Encoding of Artifacts for RACK

The Ontology for System Design

```
//Author: Saswata paul

 // This sadl file contains additional ontology beyond the RACK core
needed to specify system design


 uri "http://sadl.org/DESIGN.sadl" alias rd.

 import "http://arcos.rack/PROV-S".
 import "http://arcos.rack/SYSTEM".
 import "http://arcos.rack/SOFTWARE".
 import "http://arcos.rack/HARDWARE".
 import "http://arcos.rack/REQUIREMENTS".
 import "http://arcos.rack/REVIEW".
 import "http://arcos.rack/TESTING".
 import "http://arcos.rack/HAZARD".
 import "http://arcos.rack/ANALYSIS".
 import "http://arcos.rack/PROCESS".
 import "http://arcos.rack/AGENTS".
 import "http://arcos.rack/CONFIDENCE".



//-- Different levels of requirements
 SystemLevelRequirement (note "System Level Requirement") is a type of
REQUIREMENT
      described by rd:satisfiedBy with values of type REQUIREMENT //
redundant. Will use "satisfies" to show upward traceability
      described by rd:derivedFrom with values of type string // Should
we enumerate this in the future?
      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.
```

```
 //-- Additional properties on the RACK REQUIREMENT class
 rd:name describes REQUIREMENT with values of type string.
 architectureAllocation describes REQUIREMENT with values of type
THING.
 rd:derivedRequirement 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.



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

The Ontology for FHA

```
//-- Author: Saswata Paul
//-- An ontology to represent hazrad analysis for the FAA AI/ML
Certification Project

 uri "http://sadl.org/FHA.sadl" alias fha.
 import "http://arcos.rack/PROV-S".
 import "http://arcos.rack/DOCUMENT".
 import "http://arcos.rack/SYSTEM".
 import "http://arcos.rack/SOFTWARE".
 import "http://arcos.rack/HARDWARE".
```

```
 import "http://arcos.rack/REQUIREMENTS".
 import "http://arcos.rack/REVIEW".
 import "http://arcos.rack/TESTING".
 import "http://arcos.rack/HAZARD".
 import "http://arcos.rack/ANALYSIS".
 import "http://arcos.rack/PROCESS".
 import "http://arcos.rack/AGENTS".
 import "http://arcos.rack/CONFIDENCE".

 //-------------------------------------------------------------------------
-------------------------
 //-- 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
                                    ),
                              Minor (note "Failure conditions that would
not significantly reduce airplane safety and involve crew actions that
                                 ),
                              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
               ),
                              Hazardous (note "Failure conditions that
would reduce the capability of the airplane or the ability of the crew
                                          ),
                              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
                             )}.

 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"),
                              PSSA (note "Preliminary System Safety
Assessment")}.

 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.
```

The Ontology for OPRAs

```
/* Copyright (c) 2020, General Electric Company, Galois, Inc.
 *
 * All Rights Reserved
 *
 * This material is based upon work supported by the Defense Advanced
Research
 * Projects Agency (DARPA) under Contract No. FA8750-20-C-0203.
 *
 * Any opinions, findings and conclusions or recommendations expressed
in this
 * material are those of the author(s) and do not necessarily reflect
the views
 * of the Defense Advanced Research Projects Agency (DARPA).
 */

 //-- This is a formal data model to express OP arguments in RACK-
ingestable form
 //-- Inspired by the Friendly Argument Notation
 //-- Author: Saswata Paul

 uri "http://sadl.org/OP.sadl" alias op.

 import "http://arcos.rack/PROV-S".
 import "http://arcos.rack/DOCUMENT".
 import "http://arcos.rack/SYSTEM".
 import "http://arcos.rack/SOFTWARE".
 import "http://arcos.rack/HARDWARE".
 import "http://arcos.rack/REQUIREMENTS".
 import "http://arcos.rack/REVIEW".
 import "http://arcos.rack/TESTING".
 import "http://arcos.rack/HAZARD".
 import "http://arcos.rack/ANALYSIS".
 import "http://arcos.rack/PROCESS".
 import "http://arcos.rack/AGENTS".
 import "http://arcos.rack/CONFIDENCE".

 //-- The Overarching Properties of SYSTEM
 opIntent describes SYSTEM with values of type boolean.
 opCompleteness describes SYSTEM with values of type boolean.
 opInnoquity describes SYSTEM with values of type boolean.
```

```
 //-- Some classes required for Overarching Properties
 DesiredBehavior is a type of THING. // since it may not have a
tangible representation
 DefinedIntendedBehavior is a type of ENTITY // since it has a
tangible representation
      described by behavior with values of type REQUIREMENT.

 // a strategy/justification
 Justification is a type of THING,
      described by statement with a single value of type string.

 // a statement that may be true or false
 Premise is a type of THING,
      described by statement with a single value of type string
      described by isTrue with a single value of type boolean
      described by evidence with values of type THING. // evidences to
show that a premise is correct

 //-- An Argument structure for intent
 IntentArgument is a type of ENTITY,
      described by system with a single value of type SYSTEM,
      described by justification with a single value of type string
      described by premises with values of type Premise.
```

The AARA System

```
//-- Author: Saswata Paul

 uri "http://sadl.org/aaara.sadl" alias aaara.

 import "http://sadl.org/DESIGN.sadl".

   //-- The system and components involved, and their functions
 AAARA (note "The AI-Assisted Autonomous Runway Alignment System") is
a SYSTEM
      with identifier "AAARA"
      with description "The AI-Assisted Autonomous Runway Alignment
System".
```

```
 RPS (note "The Runway Perception Subsystem") is a SYSTEM
      with identifier "RPS"
      with description "The Runway Perception Subsystem"
      with partOf AAARA
      with function compute-lateral-correction-params
      with function provide-lateral-correction-params.

 RPS-ANN (note "The Runway Perception Subsystem ANN") is a SYSTEM
      with identifier "RPS-ANN"
      with description "The Runway Perception Subsystem ANN"
      with partOf RPS.


 RPS-IF (note "The Runway Perception Subsystem Input-Filter") is a
SYSTEM
      with identifier "RPS-ANN"
      with description "The Runway Perception Subsystem Input Filter"
      with partOf RPS.

 FMS (note "The Flight Management System") is a SYSTEM
      with identifier "FMS"
      with description "The Flight Management System"
      with partOf AAARA.

 AutoPilot (note "The Autopilot") is a SYSTEM
      with identifier "AutoPilot"
      with description "The Autopilot"
      with partOf AAARA.

 //-- Functions
 runway-guide (note "guide an aircraft to the correct path in
alignment with a runway") is a FUNCTION
      with identifier "runway-guide"
      with description "guide an aircraft to the correct path in
alignment with a runway".


 compute-lateral-correction-params (note "compute lateral corrections
parameters") is a FUNCTION
      with identifier "compute-lateral-correction-params"
      with description "compute lateral corrections parameters".

 provide-lateral-correction-params (note "provide lateral corrections
parameters to aircraft control systems") is a FUNCTION
      with identifier "provide-lateral-correction-params"
```

```
      with description "provide lateral corrections parameters to
aircraft control systems".
```

The AARA FHA

```
//-- Author: Saswata Paul
//-- An instance of hazard assessment for the AI-Assisted Autonomous
Runway Alignment (AAARA) system


 uri "http://sadl.org/aaara-fha.sadl" alias aaara-fha.

 import "http://sadl.org/FHA.sadl".
 import "http://sadl.org/aaara.sadl".

  //-- Failure Condition
 AARA-FC-01 is a HAZARD
      with identifier
      with name "Loss of Lateral Steering"
      with description
safe landing"
      with eventPhase LDG
      with H:effect
landing"
      with severityClassification Catastrophic
      with minimumRequiredDal LevelA
      with classificationJustification
aircraft to perform safe landing"
      with verificationMethod PSSA
      with affects AAARA.


 AARA-FC-02 is a HAZARD
      with identifier
      with name "Incorrect Lateral Steering"
      with description
safe landing"
      with eventPhase LDG
      with H:effect
landing"
      with severityClassification Catastrophic
      with minimumRequiredDal LevelA
      with classificationJustification
aircraft to perform safe landing"
```

```
    with verificationMethod PSSA
    with affects AAARA.

RPS-FC-01 is a HAZARD
    with identifier
    with name
    with description
redundant RPS/Camera systems."
    with eventPhase LDG
    with H:effect                                                    g
phase may fail to determine lateral path alignment to the runway,
resulting in an unsafe landing."
    with severityClassification Catastrophic
    with minimumRequiredDal LevelA
    with classificationJustification                                 ta
from all redundant RPS/Camera systems may result in loss of ability
for the aircraft to perform safe landing."
    with verificationMethod FHA
    with H:source RPS
    with affects AAARA.

RPS-FC-02 is a HAZARD
    with identifier
    with name "Hazardously Misleading Lateral Correction Parameter"
    with description                                                 m
the combined set of RPS/Camera systems"
    with eventPhase LDG
    with H:effect                                                    g
phase may determine an incorrect lateral path relative to the runway,
resulting in an unsafe landing."
    with severityClassification Catastrophic
    with minimumRequiredDal LevelA
    with classificationJustification
hazardously misleading lateral correction from the combined set of
RPS/Camera systems may result in the loss of ability to perform safe
landing."
    with verificationMethod FHA
    with H:source RPS
    with affects AAARA.

ANN-FC-1_1 is a HAZARD
    with identifier
    with name "ANN Robustness Failure"
```

```
    with description "Common environmental noise/perturbations not
included in training data Scenario: An image recognition system
incorrectly classifies images that are subtly distorted (e.g., through
noise/tilt/stretching)"
    with eventPhase LDG
    with H:effect "ANN produces incorrect lateral correction
parameters"
    with severityClassification Catastrophic
    with minimumRequiredDal LevelA
    with H:source RPS-ANN
    with affects RPS.

 ANN-FC-1_2 is a HAZARD
    with identifier "ANN-FC-1_2"
    with name "ANN Out of Distribution"
    with description "Changes in the nature of input data over time
(e.g. sensor updates/calibration, processor/unit changes)."
    with eventPhase LDG
    with H:effect "Input filter filters out out-of-distribution input
data and ANN fails to produce an output"
    with severityClassification Catastrophic
    with minimumRequiredDal LevelA
    with H:source RPS-ANN
    with affects RPS.
```

The AARA Requirements

```
uri "http://sadl.org/aaara-requirements.sadl" alias ar.


 import "http://sadl.org/aaara-fha.sadl".


 Sys-Req-01 is a SystemLevelRequirement
  with identifier "Sys-Req-01"
  with description "The Runway Perception Sub-system shall compute the
angle between the aircraft centerline and the runway centerline during
approach and landing phases."
  with requirementType functionalRequirement
  with architectureAllocation RPS.


 HLR-Req-01 is a SoftwareHighLevelRequirement
  with identifier "HLR-Req-01"
```

```
  with description
aircraft centerline and the runway centerline during approach and
landing phases when the ANN receives a filtered image."
  with requirementType functionalRequirement
  with architectureAllocation RPS.


 LLR-Req-01 is a SoftwareLowLevelRequirement
  with identifier      ___
  with description
augmentation that exercises all known non-adversarial input parameter
variations that can affect the output of the model for the foreseeable
operating conditions."
  with requirementType functionalRequirement
  with architectureAllocation RPS-ANN
  with rd:derivedRequirement true.


 FMS-Sys-01 is a SystemLevelRequirement
  with identifier      ___
  with description
determination failure, when lateral correction parameters from all
three lateral correction inputs disagree for 10 consecutive cycles."
  with requirementType functionalRequirement
  with architectureAllocation FMS.
```