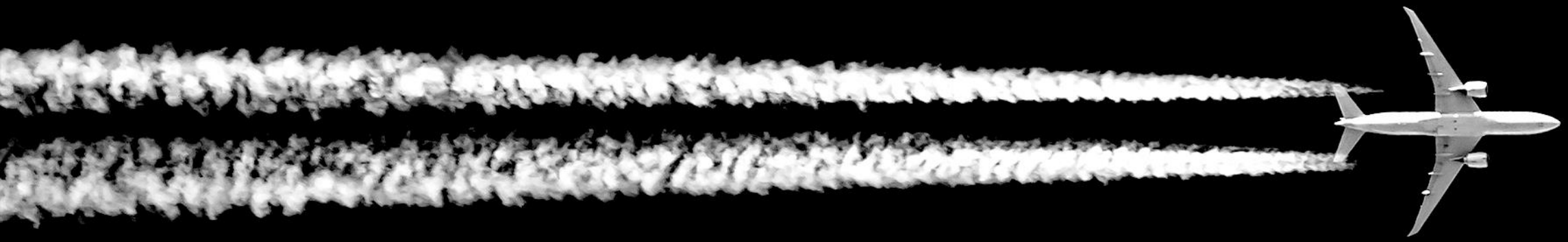
CERTIFICATION GUIDANCE FOR AI/ML

NASA CROSSCUTTING WORKING GROUP: AUTONOMY VALIDATION AND VERIFICATION DEEP DIVE



DR. DARREN COFER

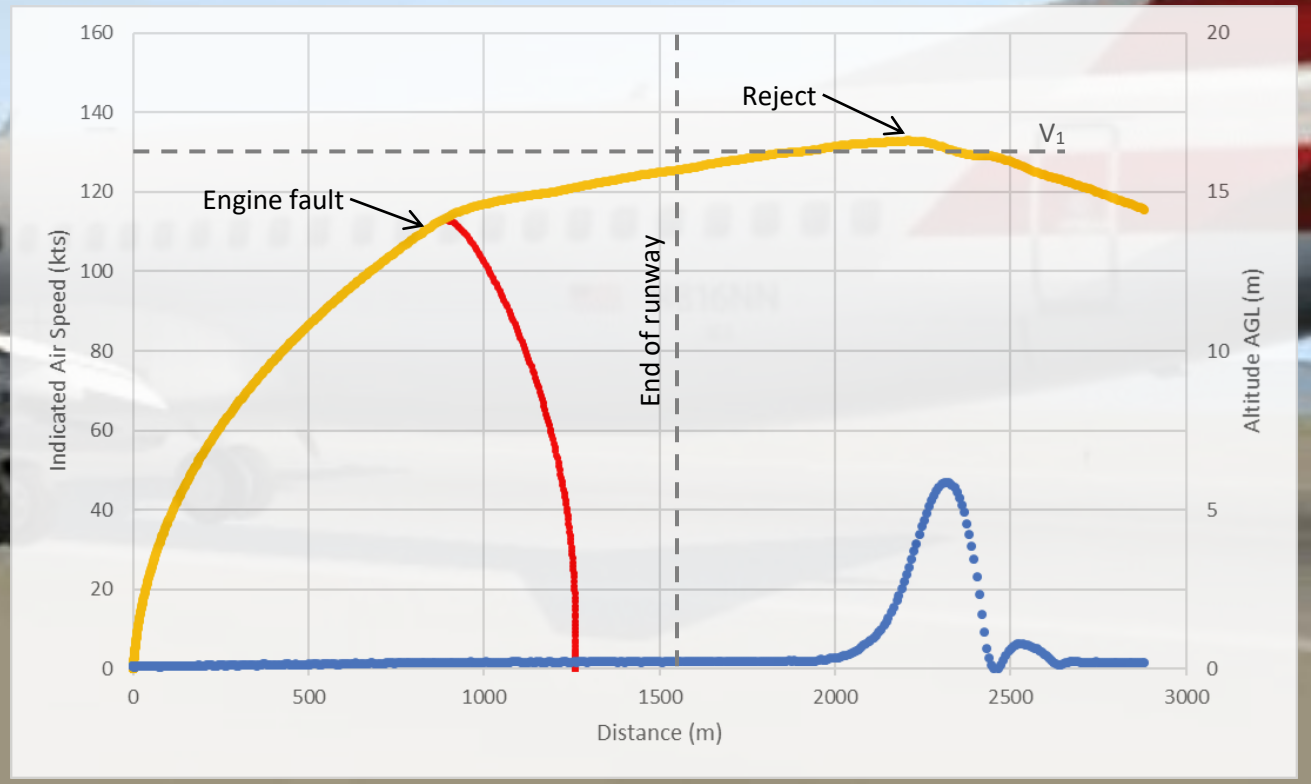
VISION 2045

19 JULY 2022

Collins Aerospace | APPLIED RESEARCH & TECHNOLOGY

WHAT COULD POSSIBLY GO WRONG?

UNINTENDED BEHAVIOR



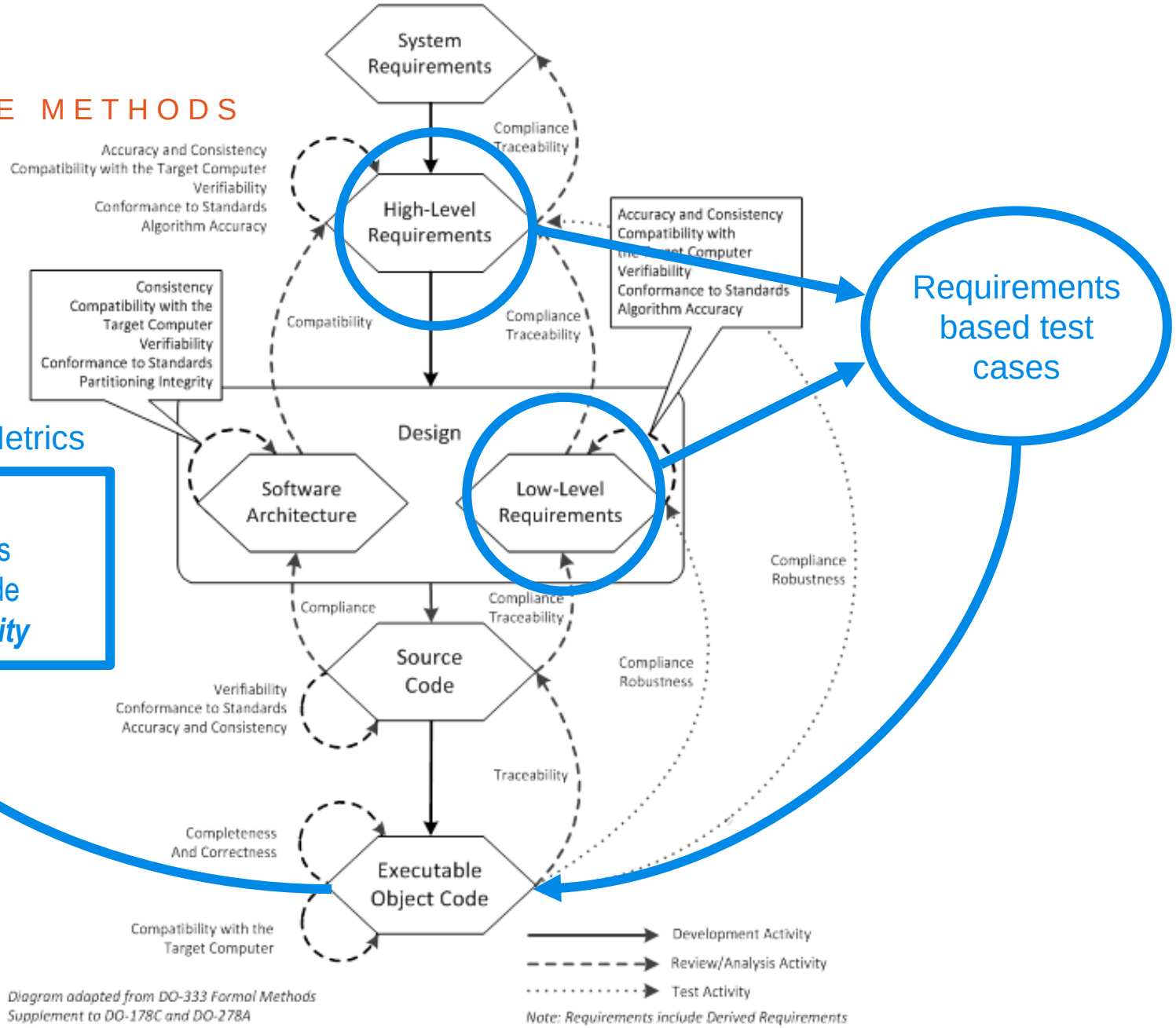
ENGINE FAULT AT 110 KTS

NN trained using reinforcement learning to make reject takeoff (RTO) decision for 737 aircraft (DARPA Assured Autonomy project)

DO-178C

TRADITIONAL DESIGN ASSURANCE METHODS

- Demonstrate that software implements its requirements
- ***and nothing else (no surprises!)***



Structural Coverage Metrics

- Missing requirements
- Inadequacy of test cases
- Dead or deactivated code
- ***Unintended functionality***

In traditional software, structural coverage helps detect unintended behaviors

When have we tested enough?

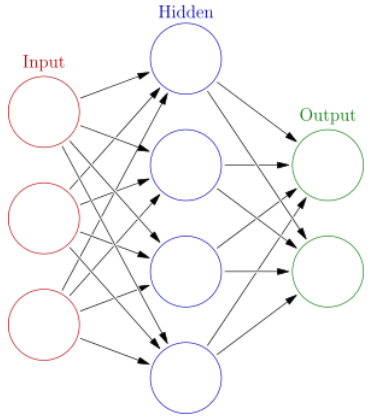
Why is this a problem for ML?

Collins Aerospace

STRUCTURAL COVERAGE FOR NEURAL NETWORKS?

- No branches or relational operators
- Complete coverage with single test case!
- Complexity and design intent reside in data, not code

How do we measure completeness of requirements and verification results to ensure the absence of unintended functionality?




```
1.function [y1] = simulateStandaloneNet(x1)
2.  % Input 1
3.  x1_step1_xoffset = 0;
4.  x1_step1_gain = 0.200475452649894;
5.  x1_step1_ymin = -1;
6. % Layer 1
7.  b1
= [6.0358701949520981;2.725693924978148;0.58426771719145909;
-5.1615078566382975];
8.  IW1_1 = [-14.001919491063946;4.90641117353245;-
15.228280764533135;-5.264207948688032];
9.   % Layer 2
10.  b2 = -0.75620725148640833;
11.  LW2_1 = [0.5484626432316061 -0.43580234386123884 -
0.085111261420612969 -1.1367922825337915];
```

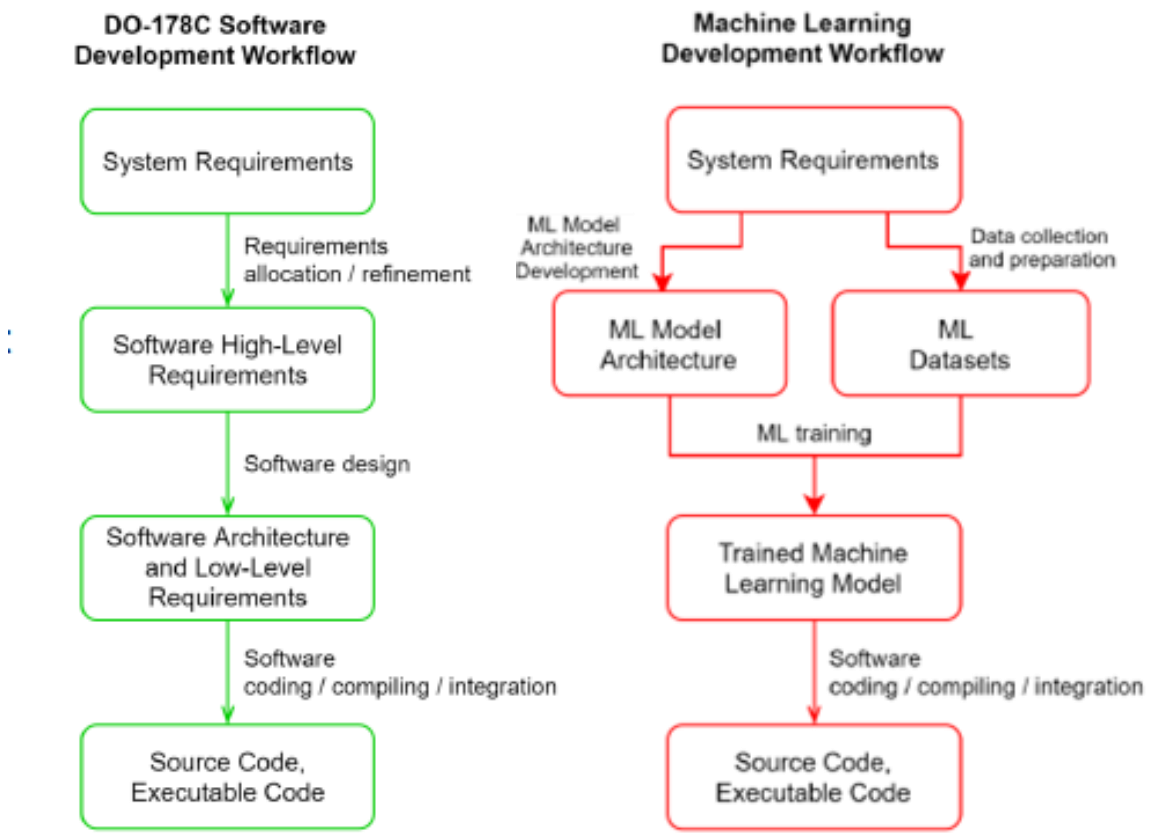
```
1.% Output 1
2.  y1_step1_ymin = -1;
3.  y1_step1_gain = 0.2;
4.  y1_step1_xoffset = 0;
5.  % ===== SIMULATION ========
6.  % Dimensions
7.  Q = size(x1,2); % samples
8.  % Input 1
9.  xp1 = mapminmax_apply(x1,x1_step1_gain,
x1_step1_xoffset,x1_step1_ymin);
10.  % Layer 1
11.  a1 = tansig_apply(repmat(b1,1,Q) + IW1_1*xp1);
12.  % Layer 2
13.  a2 = repmat(b2,1,Q) + LW2_1*a1;
14.  % Output 1
15.  y1 = mapminmax_reverse(a2,y1_step1_gain,
y1_step1_xoffset,y1_step1_ymin);
16.end
```

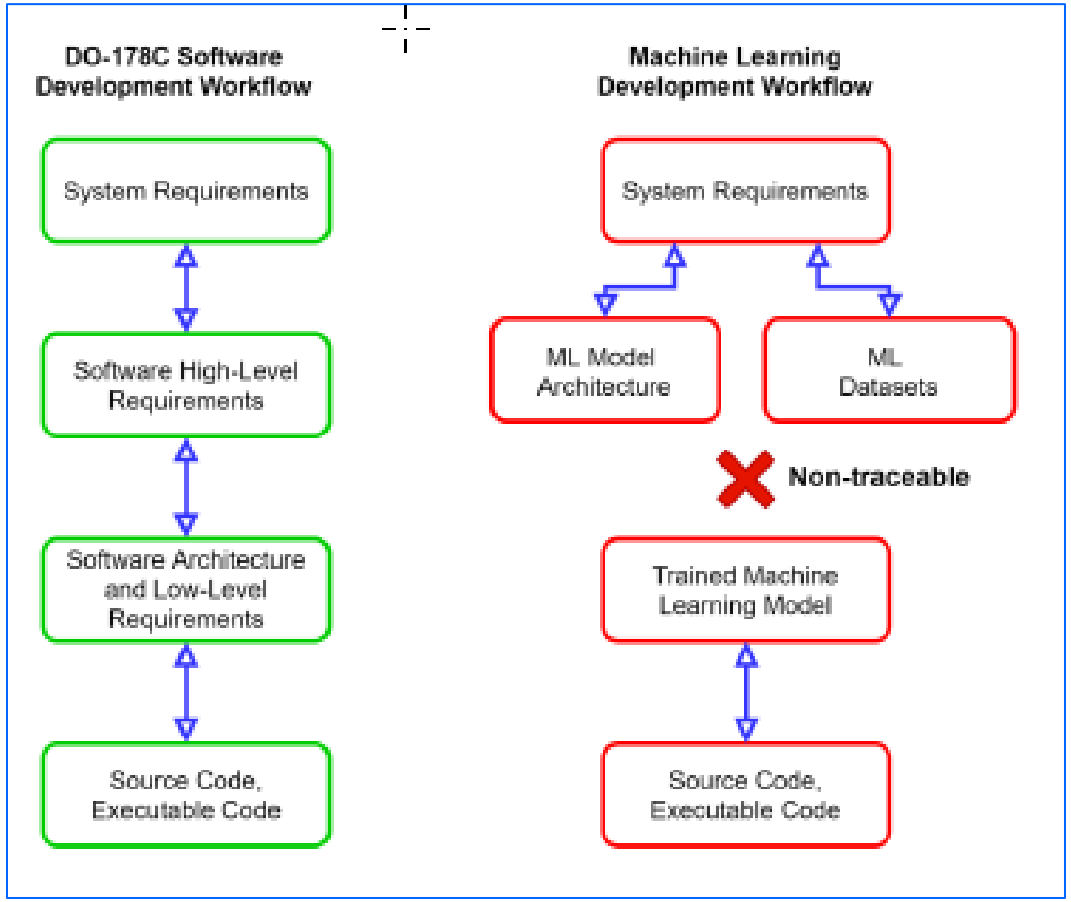
```
1.% ===== MODULE FUNCTIONS ========
2.% Map Minimum and Maximum Input Processing Function
3.function y = mapminmax_apply(x, settings_gain, settings_xoffset,
settings_ymin)
4.  y = bsxfun(@minus,x,settings_xoffset);
5.  y = bsxfun(@times,y,settings_gain);
6.  y = bsxfun(@plus,y,settings_ymin);
7.End

8.% Sigmoid Symmetric Transfer Function
9.function a = tansig_apply(n)
10.  a = 2 ./ (1 + exp(-2*n)) - 1;
11.End
12.
13.% Map Minimum and Maximum Output Reverse-Processing Function
14.function x = mapminmax_reverse(y, settings_gain, settings_xoffset,
settings_ymin)
15.  x = bsxfun(@minus,y,settings_ymin);
16.  x = bsxfun(@rdivide,x,settings_gain);
17.  x = bsxfun(@plus,x,settings_xoffset);
18.end
```

Collins Aerospace

Pei, Cao, Yang, Jana, DeepXplore: Automated whitebox testing of deep learning systems, Symposium on Operating Systems Principles (SOSP), 2017

TRACEABILITY AND ML?





Dmitriev, Schumann, Holzapfel, Toward Certification of Machine Learning Systems for Low Criticality Airborne Applications, Digital Avionics Systems Conference (DASC), 2021

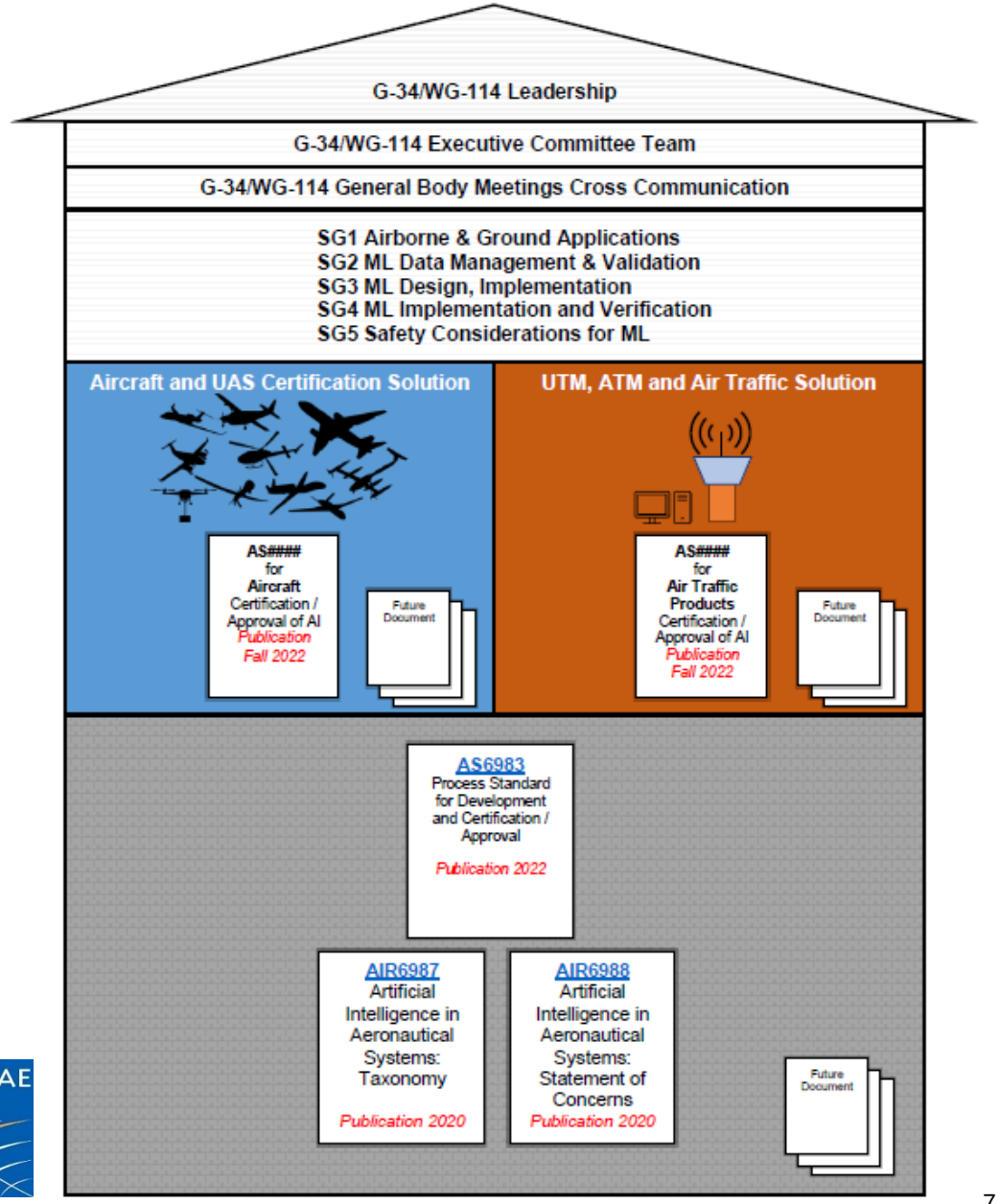
ASSURANCE CHALLENGES FOR ML

- Requirements-based testing requires requirements
  - So does formal verification of requirements
- ML inference models are created from training data, not requirements
  - Neither ML model elements (e.g., layers, neurons, weights) nor individual lines of code represent design choices that can be traced back to specific requirements
- Structural coverage is not a bad thing to do…
  - but it doesn't provide anything like the assurance we are looking for and it is really not reasonable to try to evaluate an ML model this way
- What completeness metrics or verification activities can provide assurance comparable to DO-178C objectives?
  - How can we demonstrate absence of unintended behavior?

SAE COMMITTEE G34

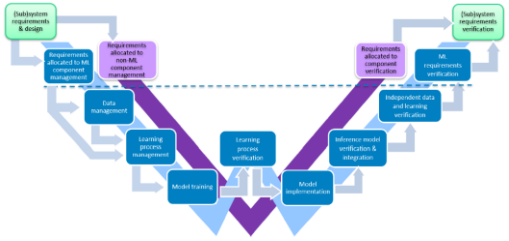
ARTIFICIAL INTELLIGENCE IN AVIATION

- Joint with EUROCAE WG-114
- Certification guidance for machine learning with offline training
- Airborne and ground use cases
- Over 500 members from government and industry
- Influenced by recent EASA reports
  - EASA AI Roadmap **(first approvals in 2025!)**
  - Concepts of Design Assurance for Neural Networks (CoDANN)
  - EASA Concept Paper: First usable guidance for Level 1 machine learning applications
- AIR6988 – Artificial Intelligence in Aeronautical Systems: Statement of Concerns (DRAFT)
- **AS6983 – Process Standard for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing AI (3Q 2023?)**



Collins Aerospace

SAE AS6983 PRINCIPLES



- Learning Process

  Use subsystem requirements to define Operational Design Domain (ODD) and training/test datasets

  - Data generation/management
  - Data is complete and representative relative to ODD
  - Model training to achieve performance target
- Verification of Trained Model

  Show absence of unintended behavior

  - Generalization
  - Stability
  - Robustness
- Inference Model Implementation

  Implement model functionality using traditional methods

  - Verification using traditional methods
- Inference Model Integration/Verification

  Show that implementation of inference model preserves properties of trained model

  - Requirements verification
  - Performance verification
  - Robustness on target hardware
  - Compatibility with target hardware

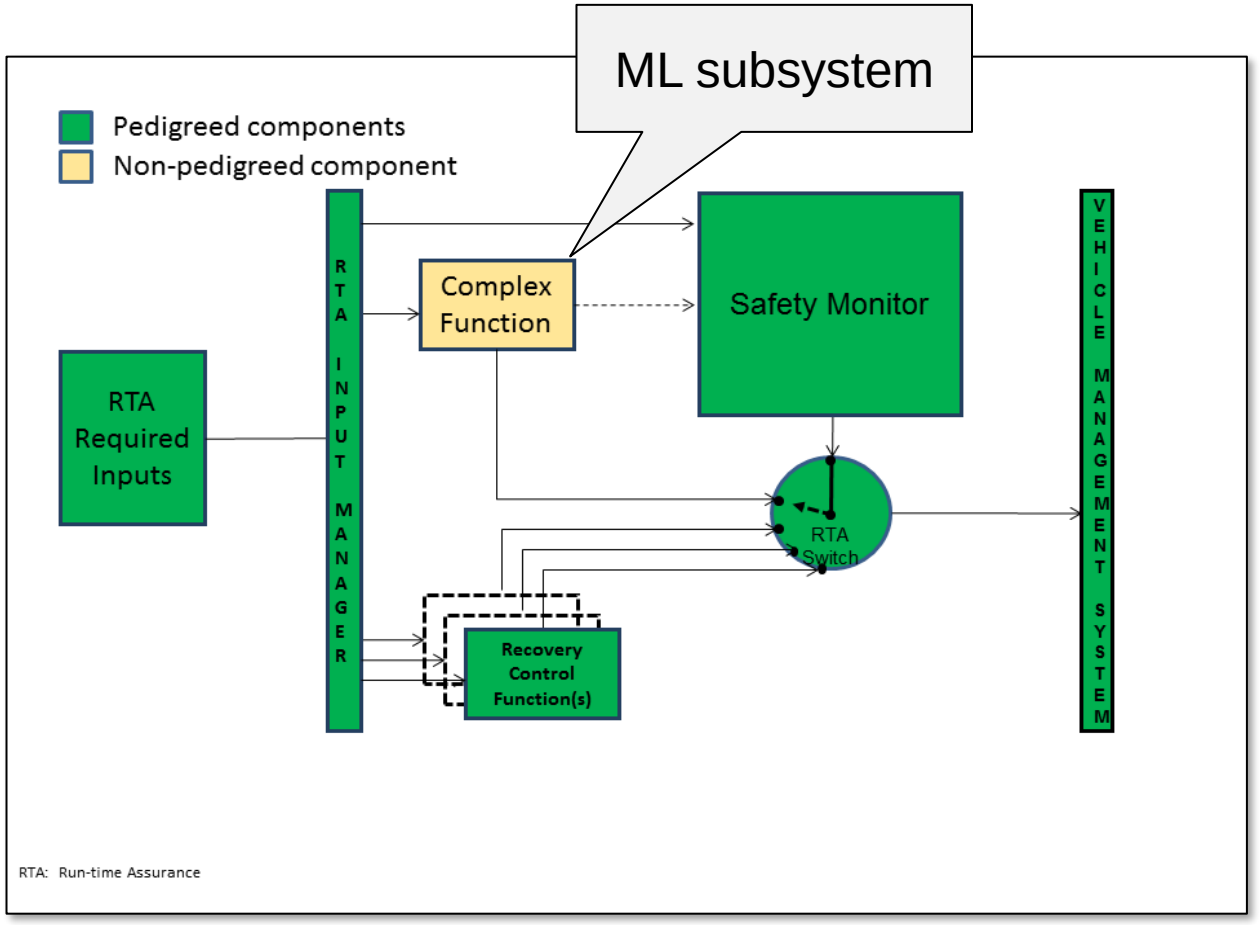
UNINTENDED BEHAVIORS

- Generalization
  - How does system respond to novel/unexpected inputs that were not included in training dataset?
  - In ODD but “between” concrete training points.
- Stability
  - How does system respond to perturbations around training data points?
  - Can small input disturbances result in large output deviations?
  - Related to adversarial inputs
- Robustness
  - How does system respond to inputs near boundary of ODD?
  - Similar to abnormal inputs (robust test cases in DO-178C).

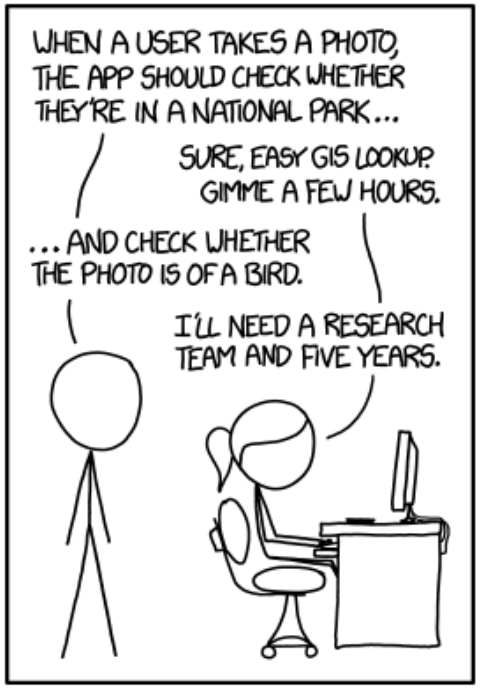
RUN-TIME ASSURANCE ARCHITECTURE

RESIDUAL UNINTENDED BEHAVIOR

- Run-time monitors detect unsafe/unexpected behaviors
- May monitor inputs, outputs, system state
- Switch to alternative safe behavior or recovery function, possibly with limited performance
- ML subsystem may still contain **unintended behavior**, but architecture ensures that there is no impact on system safety
- May allow reduced assurance level for ML subsystem
- ASTM F3269-17
  - *Standard Practice For Methods To Safely Bound Flight Behavior Of Unmanned Aircraft Systems Containing Complex Functions*

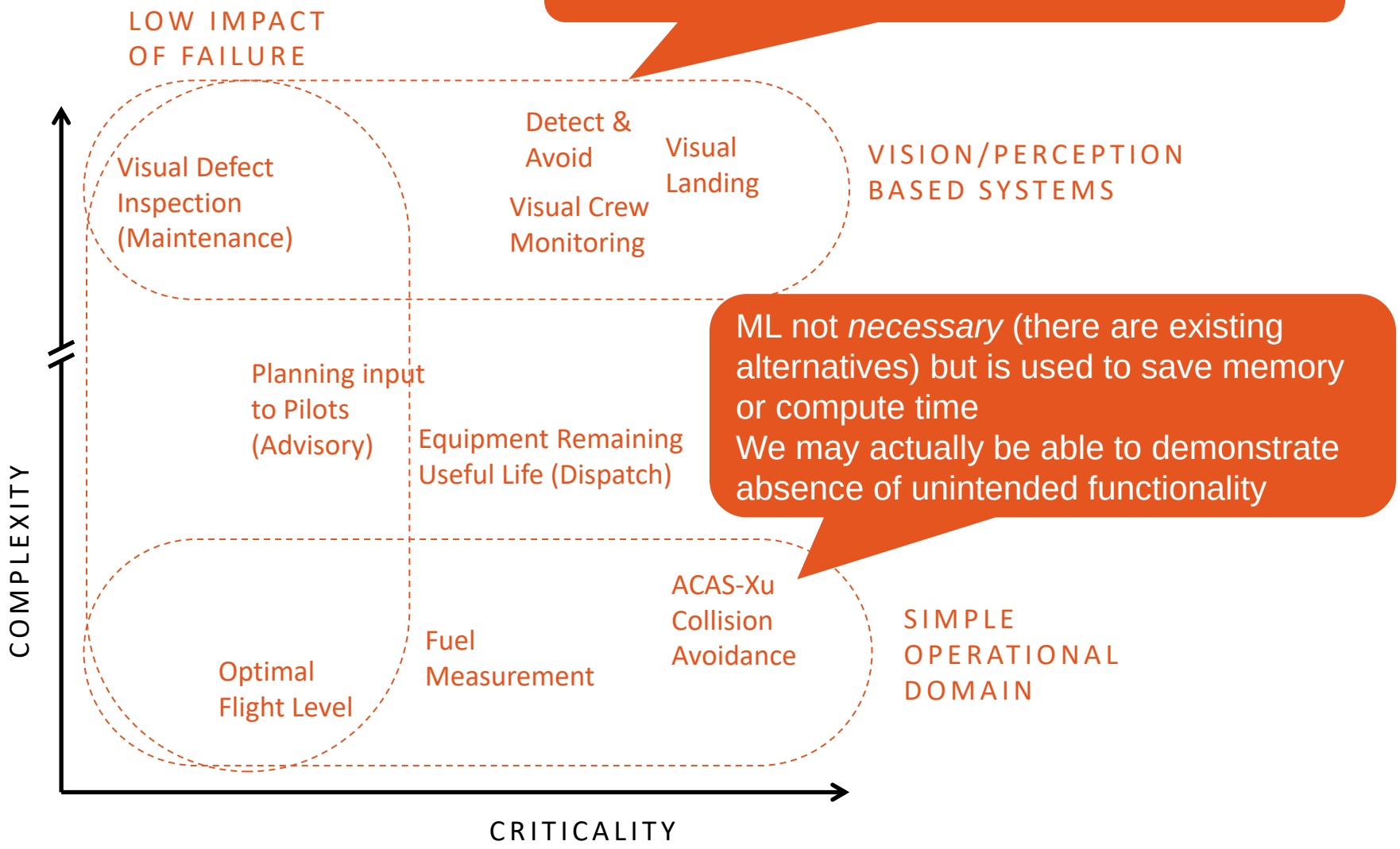


AVIATION ML APPLICATIONS



https://xkcd.com/1425/

Perception is a strong point for ML/DNN, but also hardest to verify (size, requirements)



DISCUSSION / CHALLENGES

- What are the ML-based functions that will be needed to implement more-autonomous aircraft?
- Requirements specification for perception-based systems
  - Completeness and representativeness of training data in image-based input domains
- Scalability of analysis tools for showing generalization, stability, robustness
- Metrics to show “completeness” of testing
- Certification of specialized hardware for large inference models
  - COTS HW in this domain likely to remain proprietary