Toward the certification of safety-related systems using ML techniques: the ACAS-Xu experience

Christophe Gabreau, Adrien Gauffriau, Florence De Grancey, Jean-Brice Ginestet, Claire Pagetti

Toward the certification of safety-related systems using ML techniques: the ACAS-Xu experience

Christophe Gabreau*†, Adrien Gauffriau†, Florence de Grancey*‡, Jean-Brice Ginestet§, Claire Pagetti¶
* IRT Saint Exupéry, † Airbus, ‡ Thales, § DGA, ¶ ONERA

***Abstract*—In the context of the use of Machine Learning (ML) techniques in the development of safety-critical applications for both airborne and ground aeronautical products, this paper proposes elements of reasoning for a conformity to the future industrial standard. Indeed, this contribution is based on the EUROCAE WG-114/SAE G-34 ongoing standardization work that will produce the guidance to support the future certification/approval objectives. The proposed argumentation is structured using assurance case patterns that will support the demonstration of compliance with assurance objectives of the new standard. At last, these patterns are applied to the ACAS-Xu use case to contribute to a future conformity demonstration using evidences from ML development process outputs.**

***Disclaimer*: This paper is based on the EUROCAE WG-114/SAE G-34 standardization results at the time of the writing. Though some of the authors are active members of the working group, it is a free interpretation of the current draft work and only reflects the authors' view. As the working group has not published any released outcomes yet, some parts of the described argumentation may have to be modified in the future to conform to the final standard objectives.**

I. Introduction

A. Context

In the avionics context, the certification of aircraft systems is ruled by the regulation authorities, e.g. EASA for Europe and FAA for the United States. EASA developed Certification Specifications (CS 2x.1301/1309) defining the requirements that rule systems airworthiness. In addition to this, Authorities published AMC/AC (Acceptable Means of Compliance/Advisory Circular) to recognize that developing systems using industrial standards (ED-79A/ARP4754A for complex systems, ED-12C/DO-178C for software item and ED-80/DO-254 for hardware item) are acceptable means to show evidence that a system behavior, operating functions implemented by software and/or hardware items, is compliant with the regulation requirements.

Among the methodologies used for certification purposes, the assurance case concept is not new. The safety domain was one of the first to elaborate the safety cases concept. Safety cases were originally theorized by Tim Kelly [KBMB97] and then generalized by John Rushby [Rus15]. In particular, in [Rus15], Rushby claims that the introduction of this kind of methodology in the industries are *a significant contribution to system and software assurance and certification*.

B. Objectives of the paper

The first objective of the paper is to present the guidelines drafted by the EUROCAE WG-114/SAE G-34 joint working group (WG-114 subsequently) on the certification of ML-based systems. The draft guideline [EUR21] is called AS6983 in the rest of the document. Those guidelines cover 3 levels of engineering:

1) System/Subsystem: Classical system and safety assessment processes are used to capture the requirements allocated to the ML-based function and identify the items that will be used to implement the system. Among these requirements is the DAL (Design Assurance Level) that modulates the fulfillment of the assurance objectives. The modulation is not used in the paper because the levelling of objectives has not been discussed yet in WG-114.
2) ML Constituent: This level has been introduced by the WG-114 between subsystem and item in order to support the design of one ML-based subsystem function that can be deployed on several items. This level encompasses the data management process (design the datasets that will be used to train, test and validate the ML part of the ML constituent) and the ML model design process (train/validate the model to fit the intended function and verify that its properties are compliant with its ML model requirements).
3) Item: Both classical and ML specific process will be used to transform the designed model into an implementation model that will be hosted in a SW or and HW item.

These 3 levels of engineering are described in an end-to-end ML-based system development workflow (see Fig. 1). The semantics of the arrows is as follows: plain thick arrows mean *is an input*, plain arrows mean *produces* and dashed arrows mean *uses*.

The second objective is to structure those guidelines with a set of assurance cases patterns that will reflect the learning assurance objectives developed by WG-114 and support a possibly future demonstration of conformity for certification purposes.

The third objective is to apply the assurance cases patterns on a detailed use case. Indeed, we will apply them on the hybrid architecture promoted by [DDGG+21] to implement an Airborne Collision Avoidance System (ACAS-Xu) for drones. The purpose of this architecture is to embed several neural

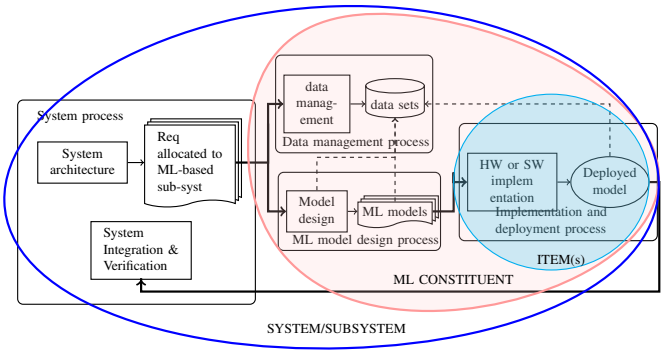

Fig. 1. End-to-end ML-based system development workflow

networks (NNs) to replace the ACAS-Xu standardized Look-Up Tables (LUT) along with a safety net based on extracts of the LUT. In that former work, we proposed an initial end-to-end certification strategy already inspired from the WG-114 works and which was reflecting the progress status of the standardization approach at this point. The WG-114 approach has evolved since and we will present an updated version in line with the current AS6983 guidelines.

II. Standardization approach

Today the WG-114 is a worldwide group of more than 500 engineers that draws its expertise from a large variety of industry fields such as air-framers, Unmanned Aircraft systems (UAS), Urban Air Mobility (UAM), electric Vertical Take-Off and Landing (eVTOL) manufacturers, engine manufacturers, airborne and ground equipment manufacturers, regulators or air navigation service providers.

A. Overview of the guidelines

As mentioned in their "Statement Of Concerns (SOC)" [EUR20a], the WG-114 anticipated *a growing commercial pressure for Artificial Intelligence (AI) solutions within the aerospace industry over the coming few years, there is an urgent call for regulation and the emergence of norms around acceptable usage*. The role of this joint group will be to produce a new standard for the development and the certification of aeronautical products using artificial intelligence (once recognized as an acceptable means of compliance by the adhoc authorities). The standard will be multi-domains, the joint working group will evaluate key applications for AI usage within aeronautical systems, with a scope encompassing ground-based equipment, airborne vehicles and Air Traffic Management (ATM) systems. In terms of processes, the full life cycle will be under consideration, from design and manufacture, to operation and through-life maintenance. In its first issue, the standard will focus on *offline trained Machine Learning (ML) based systems* meaning that the embedded ML algorithms are only trained on ground and does not keep on learning during operation.

The SOC document has allowed the whole industry to align on the main concerns related to the use of AI in the safety-related systems that make the aeronautical ecosystem. One of the main challenges raised by the document was to determine how the future standard was going to interfere with the other existing standards that currently rule the development and the certification of the aeronautical systems. This paper intentionally focuses on avionic systems processes and their applicable standards: safety assessment (ARP4761 [SAE96]), system development (ARP4754A/ED-79 [SAE10]), software development (DO-178C/ED-12C [RTC11]) and hardware development (DO-254/ED-80 [RTC00]). In order to propose an end-to end certification approach, the WG-114 considered all these processes and has identified the gaps with the existing standards in the chapter 4 of the SOC document. At this point of time, there is a collegial consensus around a *process-oriented standardization approach* introducing new assurance objectives (a.k.a. learning assurance objectives) to cover the development processes and fill the identified gaps. Therefore this paper considers the ongoing work of the working group and zooms on the levels of engineering that are impacted so far.

B. Application to the use case

The airborne anti-collision system ACAS X [KHC12] was developed to overcome some limitations of TCAS systems and as a response to flights traffic increase. In the ACAS Xu standard, the design relies on a set of offline computed lookup tables to make avoidance decisions. Some works [KBD+17] proposed to replace those LUT with some surrogate neural networks: the purpose was to reduce the memory footprint and thus to improve the execution time. The former work [DDGG+21] relies on neural networks as proposed by [KBD+17] but also on a safety net based on an extract of the LUT to ensure the compliance with the reference behavior given by the LUT. This ACAS Xu hybrid architecture allows the use of ML algorithms while guaranteeing the safety of the system in all operational domain. At operation (see Figure 2), when the check module meets a pre-identified zone where the ML controller predictions are incorrect, the system switches to the safety net. In the other cases, the ML controller is operated.

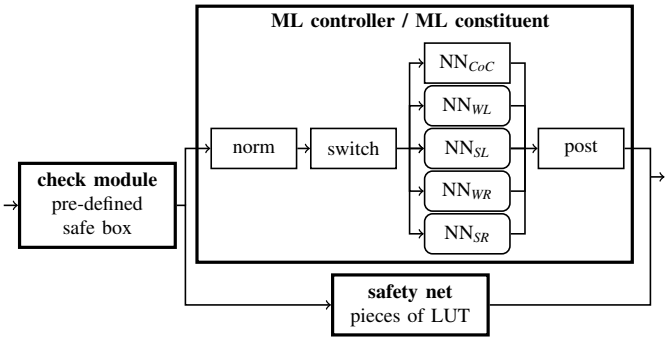

Fig. 2. Hybrid Architecture Overview

We will apply the assurance case patterns on the ACAS-Xu use case and demonstrate the compliance to learning

assurance objectives with supportive evidences. Indeed we will use evidence artefacts that have been developed during the design (ML algorithm development, performance measures, generalization demonstration using formal methods), the implementation artefacts demonstrating that the inference model is preserving the properties of the design model and the tests of the hybrid controller performed with simulation means.

C. Overview of the assurance case concept

Assurance Cases are a method to structure an argumentation in order to make a demonstration of conformity. It has been used for a while for safety demonstration in the whole industry. This method has been theorized by John Rushby in [Rus15] with this definition: *Assurance cases are a method for providing assurance for a system by giving an argument to justify a claim about the system, based on evidence about its design, development, and tested behavior*. Indeed, John Rushby emphasizes that *the argument provides a context in which to justify and assess the quality and relevance of the evidence submitted, and the soundness of the reasoning that relates this to the hierarchy of subclaims that leads to the top-level claim*. When building the argumentation related to a novel technique of development (such as Machine Learning), this may help to identify the lacks, weaknesses of the argumentation tree and thus the places where some additional research is needed.

In the updated edition of "The uses of argument" [Tou03], Toulmin states that the argumentation is mixing logic and epistemology. Considering the latter, he states that the claim of knowledge leading to the argument soundness should not be questionable, *a man who puts forward some proposition, with a claim to know that it is true, implies that the grounds which he could produce in support of the proposition are of the highest relevance and cogency: without the assurance of such grounds, he has no right to make any claim to knowledge*.

This being said, Rushby in [Rus15], clearly separates 2 kinds of argumentation strategies:

- Reasoning steps *are interpreted logically: we must determine if the conjunction of subclaims in a step deductively entails its claim*. So it may lead to a problem in logic: *do the subclaims truly imply the claim?*.

- Evidential steps *are interpreted epistemically: they are the bridge between our concepts (expressed as subclaims) and our knowledge of the world (recorded as evidence)*. Therefore it may lead to a problem in epistemology: *does the evidence amount to knowledge that the claim is true?*

Both questions are justified and emphasized in [Lev11] which highlights the possible misuse of an assurance case due to the confirmation bias (experts may try to build assurance cases enforcing the compliance of their own system) and illustrated in the Nimrod accident report [HC09]. However as underlined by [Lev11]: *the type of evidence required and assurance arguments used are straightforward with prescriptive regulation*. As mentioned earlier in the paper, this is actually the approach of the WG-114 standardization group. Indeed the working group is a college of experts challenging themselves and working on a consensus base to provide the best process-oriented assurance activities to ensure the certifiably of aeronautical systems.

Most of the recent papers on assurance cases rely on the GSN notation developed by Tim Kelly from York university [KW04] and standardized by [ACW18]. The graphical notation is really a good format to ease the analysis, favour the use of patterns, ease the concurrent working and allow for a global view of the work. However we do think that this is not enough to fully describe the assurance case content, permit a fine configuration control and express the changes between versions that are necessary to control the credit for the final goal of the process assurance: the demonstration of conformity. Therefore we prone to mix graphical and textual notation to provide the interesting aspects of the both notations as it is done in the certification argumentation of the SAFEGUARD system in [MSG21]. At last, and to echo the previous paragraph about the confirmation bias, one can challenge the reasoning or the fact that evidences may not be sufficient to fully achieve the demonstration. To help solving the legitimate interrogation inherent to an assurance case in [Rus15]: *doubting that the subclaims to a reasoning step really do entail its claim, or that the evidence cited in an evidential step has adequate weight*, Rushby was proposing the useful concept of defeater. The GSN standard V3 [ACW21], by introducing dialectic principles now allows for challenging the reasoning or the evidential steps.

D. Notation

The assurance case extracts are patterns containing AS6983 guidance objectives as proposed by [HG18] for DO-178 and [DPP20] for the multi-core guidelines formerly detailed in the CAST-32A. From such patterns, it is up to the applicant to instantiate them for a given product. In this paper, we illustrate the instantiation on the ACAS Xu use case. The assurance case patterns were designed with the GSN V3 [ACW21] (note that they are standard assurance cases for GSN as we did not use the GSN pattern notations). The instantiations are mainly described textually. In addition, we use the following colour codes all along the paper: the grey boxes indicate that the goal is supported by classical well-known guidance, the white boxes are used to structure the argumentation and yellow boxes contain objectives from the AS6983 draft guidance. Any other colour is used to link the assurance case extracts to one another in order to ease the navigation.

III. ML-BASED SUBSYSTEM DEVELOPMENT

The future ML standard will propose an end-to-end guidance to develop and certify a system containing a ML-based function. However the WG-114 approach will try to stick on the existing guideline as much as possible. From an airborne perspective, this means using the ARP4754A [SAE10] guidance whenever possible to integrate the ML-based function at subsystem level. Most of the processes at system/subsystem level of engineering (safety assessment, system requirements capture, architecture, integration, validation and verification

processes) are reused from the ARP4754A. The AS6983 overlaps a bit with the system/subsystem ARP guidance for the beginning of the *ML constituent* development.

A. Assurance case pattern

The argumentation is based on the application of the ARP4754A objectives [SAE10] from table A-1//2.0 Aircraft and System Development Process and Requirements Capture and //5.0 Implementation verification process. As this is a pure ARP4754A pattern, this argumentation is not detailed it in the paper. It leads to the identification of the ML constituent development goal: *The ML constituent performs its intended function at acceptable level of safety for allocated DAL.*

B. Application to ACAS Xu

The ACAS Xu use case subsystem is composed of:

1) ML controller (or ML constituent): contains all the ML models approximating the LUTs in every points of the input space. The models are hosted in the *SW item 3* whereas the traditional functions (normalisation, selection of the NNs and post processing to compute the advisory maneuver) are hosted in *SW item 2*.
2) SW item 1: contains the *safety net* and the *check module*.
3) HW item: the Texas Instrument keystone platform hosts the 3 SW items defined above.

The process flow chart to produce the ACAS-Xu system (see figure 3) is an application of the ML-based system/subsystem development process from WG-114.

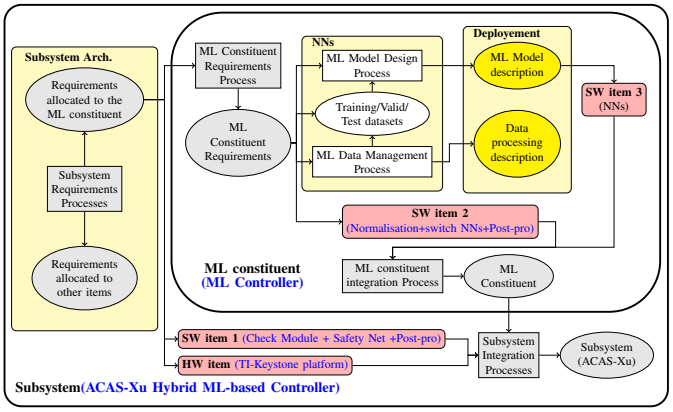

Fig. 3. ACAS-Xu system development workflow

The ARP4754A objectives from table A-1//2.0 Aircraft and System Development Process and Requirements Capture are supported by the following evidences:

- ARP4754A-2.3 (System requirements): The hybrid ML-based controller requirements are defined. The ACAS Xu function is specified using the RTCA SC-147 /EUROCAE 75 standardized tables.
- ARP4754A-2.4 (Derived requirements): The hybrid ML-based controller derived requirements are defined and rationale explained. The subsystem Operational Design Domain (ODD in the rest of the document) is partitioned between the ML controller and the safety net.
- ARP4754A-2.5 (System architecture): The strategy for the architecture definition has been developed earlier in the section.
- ARP4754A-2.6 (Item allocation): The hybrid ML-based controller requirements are allocated to items. The strategy to satisfy the objectives is to develop the entities (actually 2 items and a ML constituent) defined by the subsystem architecture. The traditional SW items Check-Module, Safety-Net and Post-pro are developed to DO-178C guidance and the HW item (the NVIDIA Jetson Xavier platform) to DO-254 guidance.

The ARP4754A objectives from table A-1//5.0 Implementation verification process are supported by the following evidences:

- ARP4754A-5.3 (Item implementation): The hybrid ML-based implementation complies with the subsystem requirements. The ML constituent development and verification processes are further elaborated to detail the demonstration of compliance to the future ML standard.
- ARP4754A-5.2 (System verification): The subsystem verification demonstrates the intended function and the confidence of no unintended function impacts to safety. The Hybrid ML based controller is tested against its functional, safety and operational requirements in a simulated environment (including a LUT simulator for comparison).

IV. ML CONSTITUENT DEVELOPMENT

The *ML constituent* contains a ML model and possibly traditional items. Its current definition is: *a defined and bounded set of either hardware item(s) and/or software item(s) that implement ML model(s) and associated ML data processing which are grouped for integration purpose to support(s) one subsystem function.*

A. Assurance case pattern

The certification argumentation is described through the fulfillment of the goal previously described at subsystem level: *The ML constituent performs its intended function at acceptable level of safety for allocated DAL.* Considering the ML constituent may contain both ML model and traditional items, the argumentation strategy is based on both new and classical guidance. The upper level goals of the ML constituent are described in the figure 4.

The learning assurance objectives are divided into 4 goals. The first *Goal req* concerns the ML constituent requirements capture. This goal is refined as an assurance case shown figure 5). *The ML constituent requirements are a satisfactory refinement of the allocated system requirements for the selected DAL.* The strategy to fulfill this goal is supported by 2 levels of requirements: the ML functional requirements and their breakdown into ML model requirements and traditional items identified by the ML constituent architecture:

- ML functional requirements: The ML functional requirements are a satisfactory refinement of the allocated subsystem requirements (functional intent, ODD specifica-

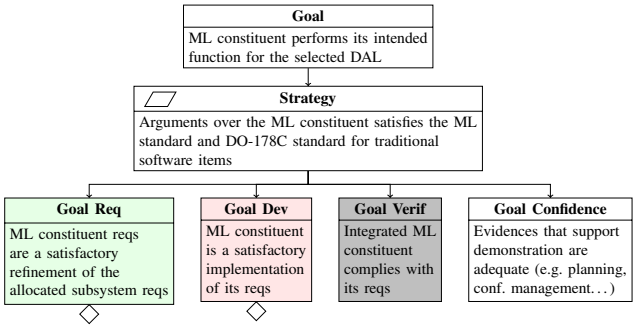

Fig. 4. ML constituent pattern

tion, robustness, DAL, system performance and resources constraints).

- Item requirements: Traditional item requirements are a satisfactory refinement of the allocated ML functional requirements for the selected DAL. The items are specified using the DO-178C guidance.
- ML model/data requirements: The ML model and data requirements are a satisfactory refinement of the allocated ML functional requirements for the selected DAL.

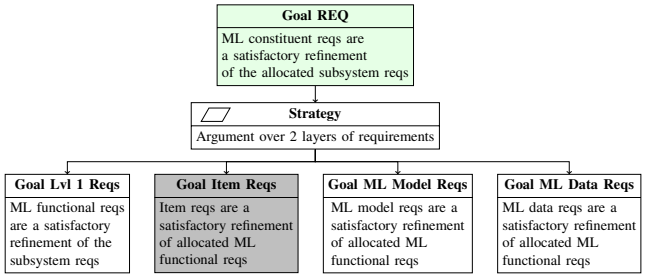

Fig. 5. ML constituent requirement capture pattern

The second goal of figure 4 is *Goal Dev* and concerns the ML constituent development. It is further developed as the assurance case of figure 6. We will detail it later, at the end of this section, as it is central in the WG114 guideline.

The third goal of figure 4 is *Goal Verif* and concerns the ML constituent integration and verification. The ML inference model and the traditional items are integrated to make the ML constituent. The goal is then to show evidence that *The integrated ML constituent (ML inference model and traditional items) complies with the ML constituent requirements*. This goal is not further detailed as it will be evidenced by traditional process artifacts.

The fourth goal of figure 4 is *Goal Confidence*. We have to show confidence that the evidences supporting this argumentation are adequate to the ML constituent development process. As a consequence, the argumentation will rely on the fulfillment of transverse objectives (from AS6983) regarding planning documentation, configuration, change management, quality assurance process and certification liaison. This latter part will not be further detailed.

Let us go back to *Goal Dev* detailed in figure 6. The related strategy is developed in the context of the ML constituent architecture (breakdown into the ML model and the traditional items).

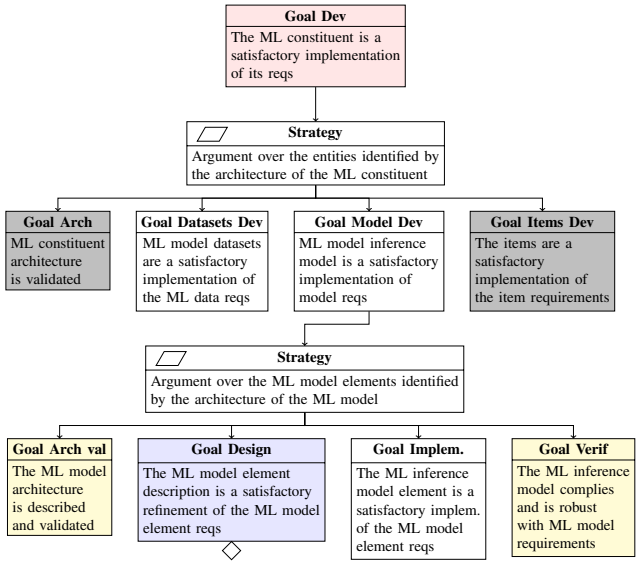

Fig. 6. ML constituent development pattern

It leads to a first level of 4 sub-goals: the *Goal Arch* validates the ML constituent architecture. The *Goal Items Dev* covers traditional items development to classical guidance. The *Goal Datasets Dev* addresses the datasets development, it will be further developed in section V. Eventually the *Goal Model Dev* (*"The ML inference model is a satisfactory implementation of the ML model requirements"*) is elaborated taking into account that the ML model may be decomposed in several ML model elements. This architecture is identified as an element of context, so that the argumentation can be based on each model element identified by the architecture. From this point, the ML model is designed, implemented and verified:

- *Goal Arch Val*: The ML model is breakdown into model elements and the architecture is validated.
- *Goal Design*: Each ML model element is trained, validated and verified using the datasets. Then the main goal is to demonstrate that *"the ML model element description is a satisfactory refinement of the ML model requirements"*. This goal is detailed in section VI.
- *Goal Implem.*: When the ML model design is terminated and frozen, each of the ML model element is implemented to make a ML inference model element. This means that the main goal becomes *"the ML inference model element is a satisfactory implementation of the ML model element description"*. This goal is detailed in the section VII.
- *Goal Verif*: When all the ML model elements are implemented, they are integrated to make the ML inference

model. Then the verification goal becomes: "*the ML inference model complies and is robust with ML model requirements*".

B. Application to ACAS Xu

The *ML constituent* (ML controller) logical architecture is composed of 4 parts (see figure 2) among which 3 traditional parts and 1 ML model:

- the 3 traditional parts are:
  - the *norm* normalizes the values of $\rho, \theta, \psi, v_{own}, v_{int}$ in the interval [-1, 1];
  - the *switch* is in charge to select the $\text{NN}_{pa,\tau}$ that will compute the advisory. The selection is done depending on the inputs pa and τ. In effect, the value of τ is decomposed in 9 possible ranges (e.g. $\tau = 0$ is the first range);
  - the *post*-processing instructions;
- the ML model is composed of 45 elements more precisely, 45 NNs named $\text{NN}_{pa,r}$ with $pa \in \{\text{CoC, SR, SL, WR, WL}\}$ and $r \in [1, 9]$.

Provided that the ML controller architecture is an element of context, the assurance case pattern of figure 4 is applied to the ACAS-Xu use case.

1) *ML controller requirements capture* (figure 5 pattern):
   - The ML controller requirements are refined from the requirements of the Hybrid ML-based controller subsystem (ODD, real-time constraints, anti collision performance, memory size constraints, DAL).
   - The SW item 2 (Normalisation+switchNNs+Post-pro) is specified using the DO-178C guidance.
   - The capture of the NNs requirements is described in section VI.
   - The NNs data requirements are specified by the Operational Design Domain (ODD), defined through the input points of the LUTs from the RTCA SC-147 Minimum Operational Performance Standards (MOPS) For ACAS-Xu. The ODD is divided into sub-ODDs to fit the 45 ML model elements of the ML model architecture.
   - The NNs data requirements are checked for traceability against hybrid ML-based Controller requirements, consistency and compatibility with NNs requirements.
2) *ML controller development* (figure 6 pattern): The ML controller architecture document is created and validated. The SW item 2 is developed according to DO-178C guidance. The datasets are developed and verified against the ML model data requirements defined previously. The ML model (NNs) is developed:
   - Goal design instantiation: The ML model is breakdown into 45 ML model elements (45 NNs). Specifically, it is verified that the union of the 45 ODDs makes the ML Controller ODD. The ML Model Description (MLMD) document is created with the NNs architecture and validated. Each NN is trained, validated and verified using the related datasets. Each NN description is added to the MLMD.
   - Goal implem instantiation: The 45 NNs are implemented from their ML model element description. See section VI for details.
   - Goal verif instantiation: The in-sample generalization capability of the NNs has been formally proven in the design phase. In addition, it has been demonstrated that the implementation process does not alter the semantics of the NNs. Therefore there is no need to demonstrate the robustness/stability in the inference environment. The Requirement-based verification is covered by the verification activities performed at ML controller level.
3) *ML controller test* (figure 4 pattern): The ML controller is verified against its requirements in a simulated environment (including the LUT simulation for functional/safety testing). These activities are covered by classical guidance.

V. Data Management

The objective of the data management process is to deliver trustworthy training, validation and test datasets which will be used to design, implement and integrate the ML model, in order to achieve the delivery of the ML inference model that meets the functional and operational requirements. The figure 7 is an overview of the data management process as per WG-114 current work.

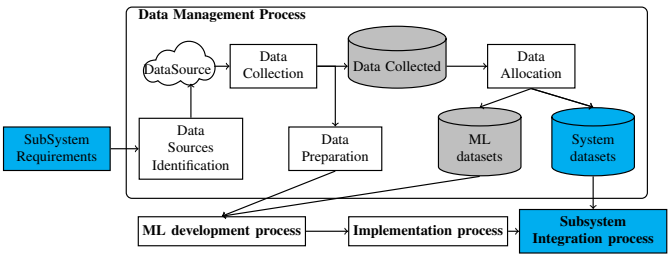

Fig. 7. Data Management Process

A. Assurance case pattern

The data management process refined the *Goal Datasets Dev* of figure 6 and is detailed in the full assurance case (but not in this paper due to lack of space). This process has to be covered by specific learning assurance objectives in order to guarantee the training of ML algorithms in the context of safety-related functions. The argumentation is split in 2 strategies applicable to ML model data: data management (sources identification, data collection, preparation and allocation) and verification.

B. Application to ACAS Xu

The ACAS Xu use case has only few activities in terms of data management as the ML algorithms are trained with the standardized LUTs from the MOPS [EUR20b]. Thus

dataset = LUTs

VI. ML MODEL DEVELOPMENT

The *ML constituent development process* has been fully described in the section IV-A. In particular, the ML constituent has first been decomposed into ML model and traditional items. This section describes the AS6983 assurance objectives that cover the *ML model (only, that is part of the ML constituent) development process* and its output: the ML model description. This artefact is essential, it shall contain all the necessary information for the implementation of the ML model.

We remind that the *ML model* itself has been broken down into *ML model elements* that have to be trained, validated and verified. The development of the *ML model element* is based on 2 main goals identified in the previous sections:

- *Goal Model Reqs* (Figure 5): *The ML model requirements are a satisfactory refinement of the allocated ML functional requirements*. There is a second level of refinement to define the ML model element requirements. The strategy is double: capture the requirements that are necessary for the model element development (e.g. specification, performances, generalization, robustness, stability) and validate them. Due to the lack of space, the assurance case about the ML model element requirements capture are not detailed in this paper.
- *Goal Design* (Figure 6): For each model element of the ML model architecture, *the ML model element description is a satisfactory refinement of the ML model element requirements*. This process is detailed in the next paragraph.

A. Assurance case pattern

This section focuses on the design assurance process of the ML model element. As described in figure 8, the argumentation to fulfill the *Goal design* is based on 2 strategies:

1) The ML model element is designed.
2) The ML model element is verified.

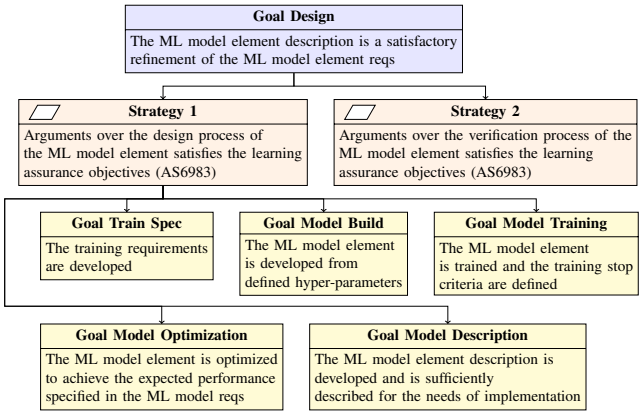

Fig. 8. ML Model design pattern

Sub-assurance case rooted from Strategy 1 (cf figure 8): The design assurance process of the ML model element is based on the following goals:

- *Goal Train Spec*: develops the training requirements. ML training activity could introduce the use of randomization that may alter the determinism and the repeatability of the design process of the ML model design process. In such cases, additional data (such as seeds values used to generate random numbers) should be defined as derived ML training requirements and managed through configuration management.
- *Goal Model Build*: selects/optimizes the hyper-parameters of the ML model element from the ML model requirements and training requirements. The ML model element is developed from these hyper-parameters.
- *Goal Model Training*: determines the ML model element parameters using the appropriate ML training algorithm and the training/validation datasets to meet the applicable requirements of the ML model element and the ML training. The initial values of the ML model element parameters, the loss function, the evaluation metrics and the training stop criteria are defined from the ML training requirements.
- *Goal Model Optimisation*: consists in performing changes in the ML model element after the training phase to achieve the expected performance specified in the ML model element requirements. In case the performance is deemed not satisfactory, derived requirements are developed to specify the required changes.
- *Goal Model Description*: develops the sufficient documentation of the ML model element design to permit the implementation of the ML model element (into a ML inference model element) including the pre/post processing instructions. The ML model element description is a part of the ML model description. There are 2 types of implementation of a ML model element, either an exact or an approximated replication of the ML model element semantics. Each ML model element description contains the design characteristics of the model element ensuring its exact (or approximated) replication in the execution environment: hyper parameters and parameters, analytical/ algorithmic syntax and semantics, replication criteria, execution environment.

Sub-assurance case rooted from Strategy 2 (from figure 8 and refined in figure 9): The verification assurance process of the ML model element is based on the following goals:

- *Goal Validation*: checks the training requirements for correctness and completeness against the ML model element requirements. ML training requirements should conform to the ML design standards and be traceable or justifiable, verifiable and consistent.
- *Goal Performance*: ensures that the performance requirements, including functional and non-functional aspects are met.
- *Goal Robustness*: ensures that the ML model element can

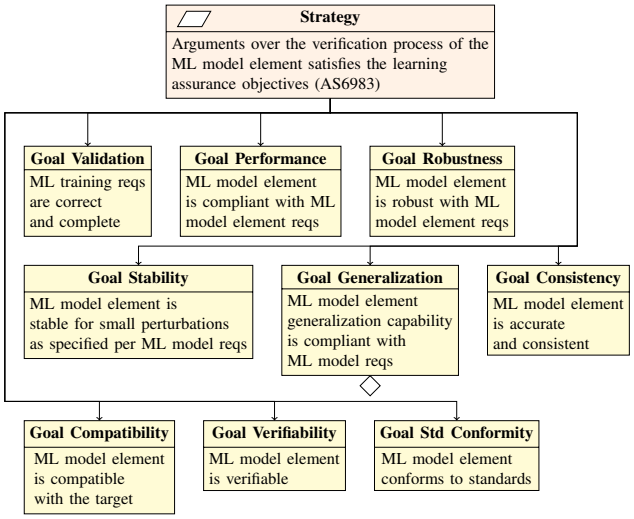

Fig. 9. ML Model verification pattern

continue to operate correctly despite abnormal inputs and conditions.

- *Goal Stability*: ensures stability of the ML model element, i.e. that small perturbations in the inputs do not activate unintended behavior. The expected level of perturbation that the ML model element should sustain has been specified in the ML model element requirements.
- *Goal Generalization*: The verification of the ML model element generalization capability is to ensure that the model element will show the same performance with unknown inputs as the one measured during the ML model element training. This argumentation (cf figure 10) is further detailed in the next paragraph.
- *Goal Accuracy and consistency*: determines the correctness and consistency of the ML model element (e.g. stack usage, memory usage, fixed point arithmetic overflow).
- *Goal Compatibility*: ensures that no conflict exist between the ML model element and the hardware/software features of the target platform, especially for the system response time and the input/output hardware.
- *Goal Verifiability*: ensures that the ML model element does not contain elements or structures that cannot be verified, for example neurons which can never be activated during ML Model element testing or random functions which cannot be reproduced during testing.
- *Goal Std Conformity*: ensures that the ML design standards are followed during the design process of the ML model element and that deviations from these standards are justified.

B. Application to ACAS Xu

Requirement considerations - each of the 45 ML model elements (NN) is specified and validated.

- Behaviour: each NN shall replicate the LUT prediction in its allocated ODD (LUT property).
- Performances: memory footprint, timing and accuracy
- Generalization capability: the LUT property shall be preserved whatever the position of the ownship and the intruder in the input space of the ODD allocated for the ML model element.

Design considerations - the strategy 1 is detailed:

- Goal train spec instantiation: Training is defined to perform a regression task. The training metrics is defined as the mean square error between truth costs and predicted costs.
- Goal model build instantiation: The identified hyperparameters consist in a set of model architecture parameters (numbers of neurons, number or layers, activation function) and training parameters (size of batches, learning rate). Hyperparameters are defined after an optimization search phase performed with Bayesian optimization.
- Goal model training instantiation: The NN is trained until a fixed and large number of epochs is attained. An early stopping criteria is added to avoid overfitting behaviour.
- Goal model optimization instantiation: Pruning methods are used to optimize the NNs memory footprint. Performances are measured and the best model element is retained.
- Goal model description instantiation: The ML controller contains the architecture description of 45 ML model elements (NN). Each NN description contains the design characteristics of the network: hyperparameters and parameters, analytical/algorithmic syntax and semantics, replication criteria, execution environment. The exact replication is selected so that the inference model can inherit from the model performances attained during the design phase if the model semantics is preserved during implementation.

The strategy 2 is substantiated regarding the verification of the performance (each NN accuracy and memory footprint are verified against the NN requirements). There are no verification needs for the robustness and the stability because there are no such requirements.

The generalization aspect is fundamental to the certification demonstration. The NN generalization property is demonstrated by proving that the LUT property is preserved for each NN whatever the ownship/intruder situation in the input space (cf figure 10). The argumentation is split in 2 parts:

1) Identification - All the input situations where the NN and the LUT predictions are different, are considered as incorrect (the NN does not preserve the LUT property).
2) Mitigation - This part is already addressed per the subsystem architecture design: the ACAS-Xu hybrid ML-based controller switches from the ML model (NNs) to the LUTs (Safety Net) when incorrect situations are detected (this is already described in the ACAS-Xu subsystem architecture document).

Therefore only the identification is at stake. Indeed, all the inputs where the NN predictions are incorrect should be identified, i.e. wherever the LUT property is not true in the

input space. The method is to partition the input space (ODD) into into p-boxes (where corners are the points of the LUTs). Then the LUT property is checked in all p-boxes: for each p-box, it is verified that the NN prediction is the same as the true prediction of one of the p-box corner points. As per the paper [DDGG$^+$21], the verification is performed using 3D boxes. Formal methods are used to make the demonstration. The argumentation is decomposed into 3 goals:

- The LUT property is correctly defined. Actually this proof is already available in the ML model requirements validation report.
- The input space (ODD) is correctly decomposed into 3D-boxes (proof: Generalization analysis report).
- The LUT property is formally checked in each 3D-box of the input space (proof: Generalization analysis report)

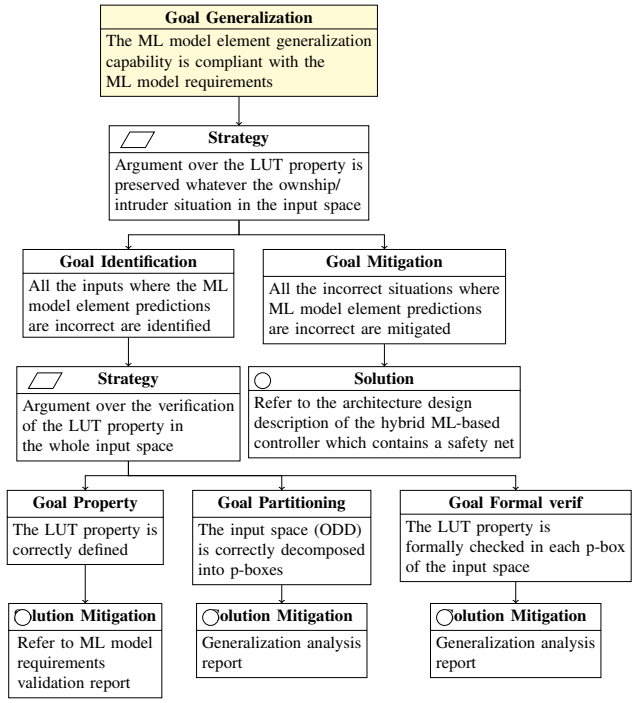

Fig. 10. ML Model Generalization demonstration for ACAS-Xu

VII. Implementation and deployment

The input for the implementation phase should be a trained ML model element that has a complete ML model element description. The implementation process (see figure 11) produces a ML inference model element that is capable to infer on a HW or SW item. It may also optimize the ML model element (without possible retraining) in order to increase the computation performance or better fit the targeted hardware resources, however it must ensure that any optimization preserves the semantics of the input ML model element or at least leads to an acceptable deviation (e.g. merging of convolution/batchnorm/ReLu layers or Winograd algorithms).

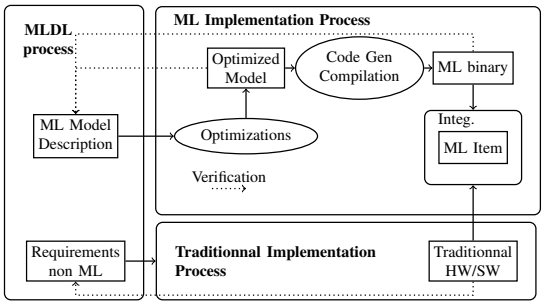

Fig. 11. Implementation process

A. Assurance case pattern

The main goals for the implementation process are the following (the assurance case extract is not provided due to the lack of space):

- *The description of the ML inference model element is a satisfactory refinement of the ML model element description* - The sub-goals are:
  - *Semantics preservation*: Any modification of the model element semantics due to the transformations (optimizations, conversion to the target environment) should be analysed for their impact on the model element performance and behavior with respect to the model element requirements.
  - *Training and target environment differences*: Differences between the 2 environments are identified to assess the impact on behavior and performance of the ML model element, and to evaluate to what extent the activities performed in the design environment can be used as verification credit for the demonstration that the inference model element complies with the model element requirements.
- *The SW or HW item is a satisfactory implementation of the ML model element description allocated to the item and meets the target constraints*: According to the AS6983 current guidance, this objective can be fulfilled using the current standard practices.
- *The integrated HW/SW items (host of the ML inference model element) comply with the ML model element description* - The sub-goals are:
  - *Compatibility with the target*: Timing, memory, latency, throughput and other non-functional requirements should be satisfied by the ML inference model element running on the target environment.
  - *Unintended behavior detection*: The ML inference model element does not introduce unintended behavior relative to the ML Model element.
  - *Design performances preservation*: the performance of the HW/SW item(s) implementing the ML inference Model on the test data set should be verified and documented. Any deviation should be measured and reported to the safety assessment process.

- *The verification of the verification is achieved*: The verification procedures are correct and complete against the ML model requirements. Functional and structural coverage are verified.

B. Application to ACAS Xu

The ML controller is implemented on a TI Keystone platform. The 45 NNs are hosted in a SW item (SW item 3 on a ARM unit). In this context, the ML element description document is updated and validated.

Hereunder the substantiated goals:

- *ML inference model element*:
  - *Semantics preservation*: No post-training optimization is performed. The 45 models are converted to a format that is compatible with the inference platform. The models semantics is described in the ML model description and is preserved during the implementation. This should be detailed in a dedicated dossier for certification purposes.
  - *Training and target environment differences*: The inference environment is different from the development environment, however the execution on the ARM processor is expected to be identical provided that the numerical representation and resolution are the same on both platforms. Then, credit that can be sought for the formal verification of the generalization capability performed during the design phase. A representativeness dossier should be provided for certification purposes.
- *SW or HW item implementation*: The NNs are coded in C language using specific libraries for target integration.
- *Integrated HW/SW items*:
  - *Compatibility with the target*: OTAWA tool is used to compute the memory footprint and consolidate the theoretical memory footprint evaluated during the design phase.
  - *unintended behavior*: As the exact replication of the ML model element is demonstrated then a verification credit can be sought for the formal verification activities of the design phase. This covers the objective.
  - *Design performances preservation*: Test dataset should be used to verify that the 100% accuracy objective is met in the target environment and that timing requirements is attained. This is not yet done.
- *Verification of the verification*: Not performed.

VIII. Related work

In parallel to WG-114, research field groups work on determining and solving the challenges to certify AI-based systems. For instance, the ANITI[1]/DEEL[2] research project released a comprehensive list of certification issues in their white paper "Machine Learning in certified systems" [DCW21]. The main challenges and the way the WG-114 is tackling them are synthesized in the article [FK21].

There are a lot of recent works fostering the use of assurance cases. In [Rus15], John Rushby explains the fundamentals of the theory for the use and the evaluation of the assurance cases. Michael Holloway [HG18], on behalf of FAA, translates the EUROCAE/RTCA ED-12C/DO-178C standard [RTC11] in an assurance case and expresses the underlying arguments which justify the assumption that the document meets its stated purpose of providing guidelines for avionic embedded software. [CPH20] presents patterns that can be used to develop assurance arguments for demonstrating the safety of the ML components. The argument patterns provide reusable templates for the types of claims that must be made in a compelling argument. [MSG21] is the first complete and published demonstration that a system (SAFEGUARD system enforces geofencing restrictions on unmanned aerial vehicles) possesses the overarching properties for certification approval purposes. At last, the work [DPP20] is proposing a domain-agnostic method to design and evaluate patterns (*the design pattern approach is a way of describing a recurring problem and its associated solution based on best practices*) of assurance cases. This will be very useful when time comes for constructing and releasing patterns from the collection of assurance cases available on the field. At last the safety group of university of York, with their "Guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS)" [RHH21], *defines a safety argument pattern that can be used to explain how and the extent to which the generated evidence supports the relevant ML safety claims, explicitly highlighting key assumptions, tradeoffs and uncertainties*. They suggest an end-to-end process, from system safety requirements to ML component safety case, providing guidance for both the applicant and the certification authority.

IX. Conclusions

This paper has proposed an interpretation of the current WG-114 work (future standard for offline machine learning development) through the development of assurance case patterns. These patterns have been applied to the ACAS-Xu use case, in order to structure a possible argumentation for demonstrating the conformity to the objectives of the standard. The demonstration is obviously not complete, however the main learning assurance objectives have been tackled to show evidence that the proposed process for ACAS-Xu development is certifiable. Beyond this use case, this is paving the way towards the certification of the safety-critical aeronautical products based on surrogate models. The way forward will be to complete the assurance activities on the implementation process and adjust the argumentation to the final guidance when the WG-114 standard (AS6983) is released.

[1]https://aniti.univ-toulouse.fr/en/

[2]https://www.deel.ai

References

[ACW18] Assurance Case Working Group ACWG. The goal structuring notation community standard version 2, 2018. Safety-Critical Systems Club, York, UK.

[ACW21] Assurance Case Working Group ACWG. The goal structuring notation community standard version 3, 2021. Safety-Critical Systems Club, York, UK.

[CPH20] Richard Hawkins Radu Calinescu Chiara Picardi, Colin Paterson and Ibrahim Habli. Argument patterns and processes for machine learning in safety-related systems. 2020. University of York, York, U.K.

[DCW21] IRT StExupery DEEL Certification Workgroup. White paper - machine learning in certified systems, 2021.

[DDGG+21] Mathieu Damour, Florence De Grancey, Christophe Gabreau, Adrien Gauffriau, Jean-Brice Ginestet, Alexandre Hervieu, Thomas Huraux, Claire Pagetti, Ludovic Ponsolle, and Arthur Clavière. How to certify a reduced footprint acas-xu system: A hybrid ml-based solution. In *International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, 2021.

[DPP20] Kevin Delmas, Claire Pagetti, and Thomas Polacsek. Patterns for certification standards. In *Proceedings of the 32nd International Conference on Advanced Information Systems Engineering (CAiSE'20)*, pages 417–432, 2020.

[EUR20a] EUROCAE / SAE. ER-022/AIR6988 - AI in Aeronautical Safety-Related Systems: Statement of Concerns, 2020.

[EUR20b] EUROCAE WG 75.1/RTCA SC-147. Minimum Operational Performance Standards For Airborne Collision Avoidance System Xu (ACAS Xu), 2020.

[EUR21] EUROCAE WG-114/SAE joint group. Certification/approval of aeronautical systems based on AI, 2021. on going standardization.

[FK21] Christophe Gabreau Baptiste Lefevre Fateh Kaakai, Béatrice Pesquet-Popescu. Ai for future skies: On-going standardization activities to build the next certification/approval framework for airborne and ground aeronautical products, 2021. AISafety 2021.

[HC09] C. Haddon-Cave. The nimrod review: an independent review into the broader issues surrounding the loss of the raf nimrod mr2 aircraft xv230 in afghanistan in 2006, 2009. report, vol. 1025. DERECHO INTERNACIONAL.

[HG18] Holloway and Graydon. Explicate '78: Assurance case applicability to digital systems, 2018. FAA report DOT/FAA/TC-17/67.

[KBD+17] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[KBMB97] Tim Kelly, Iain Bate, John McDermid, and Alan Burns. Building a preliminary safety case: An example from aerospace. In *1997 Australian Workshop on Industrial Experience with Safety Critical Systems and Software, Australian Computer Society, Sydney, Australia*, 1997.

[KHC12] Mykel Kochenderfer, Jessica Holland, and James Chryssanthacopoulos. Next generation airborne collision avoidance system. *Lincoln Laboratory Journal*, 19:17–33, 2012.

[KW04] Tim Kelly and Rob Weaver. The goal structuring notation /- a safety argument notation. In *Workshop on Assurance Cases*, 2004.

[Lev11] Nancy Leveson. The use of safety cases in certification and regulation, 2011. Aeronautics and Astronautics/Engineering Systems MIT.

[MSG21] Hampton Virginia Mallory S. Graydon, Jared D. Cronin Langley Research Center. Retrospectively documenting satisfaction of the overarching properties: An exploratory prototype, 2021.

[RHH21] Chiara Picardi Radu Calinescu Richard Hawkins, Colin Paterson and Ibrahim Habli. Guidance on the assurance of machine learning in autonomous systems - amlas. 2021. University of York, York, U.K.

[RTC00] RTCA/EUROCAE. DO-254/ED-80 - Design Assurance Guidance For Airborne Electronic Hardware, 2000.

[RTC11] RTCA/EUROCAE. DO-178C/ED-12C - Software Considerations in Airborne Systems and Equipment Certification, 2011.

[Rus15] John Rushby. The interpretation and evaluation of assurance cases. Technical report, 2015. Technical Report SRI-CSL-15-01 Computer Science Laboratory, SRI International, Menlo Park, CA, July 2015. URL="http://www.csl.sri.com/users/rushby/papers/sri-csl-15-1-assurancecases.pdf".

[SAE96] SAE. Aerospace Recommended Practices ARP4761- guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment is an aerospace, 1996.

[SAE10] SAE/EUROCAE. Aerospace Recommended Practices ARP4754a/ed-79a- development of civil aircraft and systems, 2010.

[Tou03] Stephen Toulmin. The uses of argument, updated edition, 2003. Original edition 1958.