Data-centric Operational Design Domain Characterization for Machine Learning-based Aeronautical Products

Fateh Kaakai[1], Sridhar ("Shreeder") Adibhatla[2],
Ganesh Pai[3]*[0000−0002−9848−3754](✉), and Emmanuelle Escorihuela[4]

[1] Thales, 3 Avenue Charles Lindbergh, 94628 Rungis, France
fateh.kaakai@thalesgroup.com
[2] Rockdale Systems LLC, Cincinnati OH 45246, USA
adiabatic@gmail.com
[3] KBR / NASA Ames Research Center, Moffett Field, CA 94401, USA
ganesh.pai@nasa.gov
[4] Airbus Operations (SAS), 316 Route de Bayonne, 31060 Toulouse, France
emmanuelle.escorihuela@airbus.com

Abstract. We give a first rigorous characterization of Operational Design Domains (ODDs) for Machine Learning (ML)-based aeronautical products. Unlike in other application sectors (such as self-driving road vehicles) where ODD development is scenario-based, our approach is *data-centric*: we propose the dimensions along which the parameters that define an ODD can be explicitly captured, together with a categorization of the data that ML-based applications can encounter in operation, whilst identifying their system-level relevance and impact. Specifically, we discuss how those data categories are useful to determine: the requirements necessary to drive the design of ML Models (MLMs); the potential effects on MLMs and higher levels of the system hierarchy; the learning assurance processes that may be needed, and system architectural considerations. We illustrate the underlying concepts with an example of an aircraft flight envelope.

Keywords: Aeronautical products · Development assurance · Machine learning · Operational design domain · System safety.

1 Introduction

Artificial Intelligence (AI)-enabling technologies like Machine Learning (ML) have the potential to transform the aviation industry by creating new products and services, and by enhancing the existing ones. However, ML introduces a new paradigm for design activities since the intended behavior of a function is inferred from a body of data using statistical learning algorithms, rather than being specified and programmed. Data is thus central to the implementation of a final product design.

In traditional aviation domain systems engineering, operational requirements capture the conditions under which an end-product is expected to fulfill its missions. Those

* Contribution to the paper with support from the System-wide Safety project under the Airspace Operations and Safety Program of the NASA Aeronautics Research Mission Directorate.

requirements, which are an expression of stakeholder needs, contain parameters and values that define an operational environment, or *operational domain* (OD), in which an aviation system must properly operate. When requirements are elicited from and allocated to different layers of the system design—namely: *function* or *system*, *subsystem*, and eventually *item*[1]—the OD is also correspondingly allocated, resulting in *operational design domains* (ODDs) corresponding to those layers.

Motivation and Contributions Specifying, refining, and allocating ODs to the system layers that will eventually integrate ML are activities not as well-understood as they are when going from a system/function layer to lower layers in conventional aviation systems development processes. Thus, a key challenge for the aviation systems domain is how to define, analyze, and manage the ODDs resulting from the allocation of the OD to the system layers integrating ML.[2] Addressing this challenge is especially important because it not only drives the data collection activities needed to ensure that a dataset representative of the intended operations is gathered, but also it influences the design of those layers and the underlying ML Models (MLMs).

To that end, this paper makes the following main contributions: (i) in Section 2, an identification of the dimensions along which the parameters that define an ODD for ML-based aeronautical products can be explicitly captured; (ii) in Section 3, a rigorous data-centric characterization of ODDs based on categorizing the data that ML-based functionality can encounter in operation. An aircraft flight envelope example also concretizes the underlying concepts; and (iii) in Section 4, an illustration of how the identified data categories can be used to determine the potential effects on the system layer integrating ML, along with the learning assurance activities and the system architectural considerations needed to mitigate those effects.

The approach in this paper is one of the cornerstones of a future process guidance document [15] for the development and certification/approval of safety-related aeronautical products implementing AI. That guidance is currently being developed through an aviation industry-based consensus process, jointly by the SAE Committee for AI in Aviation (G-34), and the EUROCAE working group for AI (WG-114).

Related Work The concept of ODD was initially introduced and developed by the automotive systems industry [16]. As such, the current literature on specifying, developing, and using ODDs is largely in an automotive systems application context. For example, ODD specification for automated driving systems (ADSs) can be aided by a domain-specific language (DSL) using structured natural language founded on a formal, machine-processable domain model, to support both human comprehension and programmatic manipulation [7]. A *divide-and-conquer* approach to automotive function ODD development can be employed using a concept of so-called μODD [11] that partitions an ODD to place useful bounds on various safety-relevant parameters. Such

[1] We use the standard aviation domain terminology for the layers of a system hierarchy/design.

[2] Additional related challenges (not in scope for this paper), such as the need to adapt requirements definition and validation processes to account for dataset requirements, have been comprehensively elaborated in [6].

partitions can then be tied to validation tests, whilst also encoding situation-specific parameter information. This approach is closest to our work, although the partitioning we achieve is data-centric, and orthogonal to μODD-based partitions. In [17], a hierarchical ODD definition is used to develop a scenario-based test framework for ADSs.

The ODD concept is being progressively matured in the automotive industry via ODD-related standards [1], [9], as well as automotive system-centric safety standards concerning ML and AI [8], [18]. Each of those guidance documents gives a mutually consistent definition for the ODD concept, emphasizing its relationship to safety. Nevertheless, automotive domain guidance cannot be directly applied to safety-critical aeronautical products owing to a variety of constraints, including: (a) differences in the regulatory approach between the automotive and aviation sectors; (b) the need for standards to be compatible with aviation regulations and regulatory acceptance of the associated compatibility arguments; (c) the stringency of assurance requirements in the aviation sector; and (d) consistency with the existing ecosystem of recommended engineering practices, e.g., for safety assessment [13], and aviation system development [14].

All of those factors, besides the key challenge discussed earlier, have additionally motivated the work in this paper. Next we give our notion of ODD.

2 System-level Considerations

Operational Domains (ODs) When designing a product system, it is an established and well-understood aviation systems engineering practice to capture and analyze stakeholder needs at an early stage, along numerous dimensions such as the mission to be fulfilled, the expected performance in different system operating phases, and specific environmental conditions encountered. An OD is one of the results of such early-stage analysis, and it is embodied by the operational requirements for that system. In other words, the OD is captured in the form of requirements via a *specification* activity of a well-defined requirements development process. Thus, we consider an OD to be a specification of all foreseeable operating conditions under which an end-product is expected (and should be designed) to fulfill its missions. For instance, a *flight envelope* specifies, at a minimum, a combination of altitude and *Mach number*[3] values that define an operational environment in which an aircraft type must properly operate.

Operational Design Domains (ODDs) We define the *allocation of an OD* to be the *operational design domain* (ODD). This is largely aligned with other definitions of ODD [12], [15], [16], [18]. Just as requirements are allocated across the different layers of the system design, and then refined with various criteria in mind, e.g., safety, architectural options, implementation choices, and physical considerations, an OD is also allocated to the lower design layers, and further refined so that each layer has its own ODD, i.e., the portion of the associated OD in which it should properly function. Such refinement can potentially (but not always) lead to rich and complex ODDs.[4] The principles and procedures governing OD allocation rely upon established aerospace practices [14]. As such, we can allocate the entire OD or a portion thereof to the subsystems

[3] Mach number is the ratio of true airspeed to the local speed of sound.
[4] Characterizing the complexity of an ODD is not in scope for this paper.

that will be implemented using ML technologies (which we refer to, henceforth, as *ML-based subsystems*). Moreover, refining requirements as indicated earlier will bring forth corresponding enhancements of the OD reflecting the same considerations.

Describing ODs and ODDs To describe an OD or ODD we elicit a variety of *parameters*, their range of admissible values, and, when relevant, distributions of occurrences over particular time intervals. In general, these define a multi-dimensional region. In practice, an OD or ODD is often likely to be a subset of that region. Although there are many ways to group parameters, the following is typical in practice:

- *Environmental Parameters*: These are variables outside the product (e.g., aircraft) system boundary, including weather conditions (ambient air temperature and pressure, wind conditions, humidity/rain/snow/ice, dust or sand levels, etc.) as well as application-specific parameters, e.g., brightness, contrast levels, and blur levels for optical sensor systems.
- *Operational Parameters*: These are parameters within the system boundary, examples of which include altitude and Mach number limits specified by a flight envelope, as well as ranges for angle of attack, pitch, roll, yaw angles, or their rates of change.
- *System Health Parameters*: These specify whether the system is expected to work only under nominal (non-failure) conditions, or whether it should be able to handle deterioration over time, sensor failures, or failures in specified system components (e.g., a failed actuator or a damaged flight control surface).

ML Constituent (MLC) Traditional systems engineering activities need to transition to ML activities at a certain stage of system development when integrating ML. In light of this, current regulatory guidance for introducing ML technologies into safety-related aeronautical applications [5], as well as ongoing standardization activities [15] have introduced a concept of *ML Constituent* (MLC) for systems integration purposes.

Effectively, an MLC represents the lowest-level of a functional decomposition from a system perspective that supports a subsystem function. It is a grouping of hardware and/or software items implementing one or more ML Models (MLMs) and their associated data pre- and post-processing items. Pre-processing may include (but is not restricted to) data cleanup, normalization, and feature computation. Similarly, post-processing may involve, among other actions, denormalization and blending of outputs from sub-models.

We qualify the ODD based on its allocation. Thus, allocating an OD to an MLC gives an MLCODD (i.e., the design space for an MLC), and likewise, the allocation to an MLM results in an MLMODD. An MLMODD may be identical to the MLCODD, though in practice it may be smaller. Additionally, an MLC can contain multiple MLMs each of which have their respective MLMODDs. Also, an MLMODD (or MLCODD) may be the same as the OD for the system, its superset (to provide robustness), or a subset thereof (to limit the design to a feasible region).

Thus, when a product will eventually integrate ML (e.g., as software whose design was learned through an ML training process) understanding the MLCODD is crucial to ensure that: (1) the data used for training is representative of that OD; and (2) the ML designer comprehends the complexity of the portion of the OD that has been allocated to machine learned functionality.

3 New ODD Concepts for Aviation

From the preceding narrative, it should be evident that developing an OD/ODD is itself not a new phenomenon in aviation systems engineering practice. However, it is the transition from an OD/ODD description to data collected for MLM training that is the major change relative to the way ODs are typically specified during conventional (i.e., non-ML based) product development. This change requires alternative approaches that are the focus of learning assurance processes [5], [10].

We now give a data-centric conceptual characterization for ODDs, that partitions them based on *categories* and *kinds* of data. Henceforth, when we refer to "ODD" and "ML", we mean the MLMODD (or MLCODD), and the MLM (or MLC), respectively, and we will qualify our usage of those terms when it is not clear from context.

Categories and Kinds of Data We define the following data categories:

(i) *Nominal*: Set of data points that lie in the interior of an ODD statistical distribution, that is *correct* with respect to the corresponding ML requirements.

(ii) *Outlier*: Set of data points outside an ODD. Some data can be mistaken to be *Outlier* data when they should have been *Nominal* data, had that ODD been correctly characterized by including at least one additional parameter.

(iii) *Edge Case*: Set of data points on an ODD boundary where exactly one ODD parameter has a valid extreme (maximum and minimum) value.

(iv) *Corner Case*: Set of data points where at least one ODD parameter is at their respective extremum (minimum and maximum value) of the range of values for those parameters that are admissible (or valid) for a given ODD (see Fig. 1 for examples). There are two types of *Corner Case* data:
   - *Feasible*: those that are part of the functional intent and, thus, within a given ODD (specifically at the vertices[5] of that ODD);
   - *Infeasible*: those that are not part of the functional intent and, thus, outside the ODD. Note that all *Infeasible Corner Case* data are a special case of *Outlier* data.

(v) *Inlier* (`InL`): Set of data that lie in the interior of the ODD following an error during data management, e.g., due to incorrect usage of units and dimensions. *Inlier* data are difficult to distinguish from *Nominal* data, and hence difficult to detect/correct.

(vi) *Novelty*: Set of data within an ODD according to the parameters used to describe that ODD, but which should have been considered to be *Outlier* data, had that ODD been correctly described by introducing at least one additional ODD parameter. In this sense, *Novelty* data points for an ODD could be seen as *duals* of those data points that are mistakenly considered to be *Outlier* data, when they should, in fact, have been *Nominal* data for that ODD. *Novelty* data usually arise from insufficient ODD characterization.

[5] ODDs without vertices e.g., an oval region, will therefore not have feasible corner cases.

We can group the *Inlier*, *Outlier* (including *Infeasible Corner Case*), and *Novelty* categories into a single *Anomaly* data category. Data drawn from all the aforementioned categories may also be characterized as among the following kinds of sets:

(a) *In-Sample* ($\texttt{InS}$): Data used during MLM learning which the implementation of the MLM will have to process during inference in operation.

(b) *Out-of-Sample* ($\texttt{OutS}$): Data not used during MLM learning that the implementation of the MLM will have to process during inference in operation. It is on out-of-sample data that acceptable generalization behavior (and a corresponding guarantee) of the implemented MLM can be reasonably expected.

(c) *In-MLMODD* ($\texttt{InMOD}$): Data that the implemented MLM will have to process during inference in operation. In-MLMODD data contribute to the intended function(s) of the MLM. We have: $\texttt{InMOD} = \texttt{InS} \cup \texttt{OutS}$ and $\texttt{InS} \cap \texttt{OutS} = \emptyset$

(d) *Out-of-MLMODD* ($\texttt{OutMOD}$): Data not seen during MLM learning that the implemented MLM *should not process* during inference in operation. Out-of-MLMODD data contributes to the intended function(s) of the MLC, e.g., specific processing to detect anomalies (see Section 4 for more details). We have: $\texttt{InMOD} \cap \texttt{OutMOD} = \emptyset$.

(e) *In-MLCODD* ($\texttt{InCOD}$): Data contributing to the intended function(s) of the MLC. We have: $\texttt{InCOD} = \texttt{InMOD} \cup \texttt{OutMOD}$.

(f) *Out-of-MLCODD* ($\texttt{OutCOD}$): Data not seen during MLM learning that the implemented MLC *should not process* during inference in operation. Out-of-MLCODD data contributes to the intended function(s) of the ML-based subsystem. We have: $\texttt{InCOD} \cap \texttt{OutCOD} = \emptyset$.

Real Data in Operation (and their associated statistical distributions), $\texttt{RDO}$, can now be defined as the set of all data seen in operation: $\texttt{RDO} \supseteq (\texttt{InCOD} \setminus \texttt{InL}) \cup \texttt{OutCOD}$.

The preceding concepts will serve as reference terms in forthcoming aviation industry specific guidance [15]. Nevertheless, we believe they are generic enough to be applicable in other domains, although there are some differences, e.g., our concept of *Edge Case* data differs from what is considered in [18].

Illustrative Example (Aircraft Flight Envelope) We now give an illustrative example of an aircraft flight envelope to concretize the preceding concepts. Informally, a flight envelope specifies the allowable combinations of two parameters—altitude ($\texttt{Alt}$) and airspeed, given here as a Mach number ($\texttt{Mach}$)—at which an aircraft design should function. Intuitively, this characterization of a flight envelope represents an Operational Domain (OD) of the aircraft system, and we refer to it, henceforth, as the *system* OD (SOD). This is closely related to a *functional* OD for the system which may include a specification of, for example, aircraft takeoff gross weights, the city pairs between which flight operations are intended, the routes (flight paths) that aircraft of a particular type design are expected to fly, the airports involved, the climb segments, and the landing approaches to be followed.

Fig. 1 presents a notional flight envelope covering all phases of flight (shown as the irregular hexagonal region A). $\texttt{Mach}$ and $\texttt{Alt}$ values within this SOD are allowed, and therefore they are expected to be encountered in operation. Values of those parameters outside the SOD are disallowed since operating outside the flight envelope is usually dangerous in most circumstances.

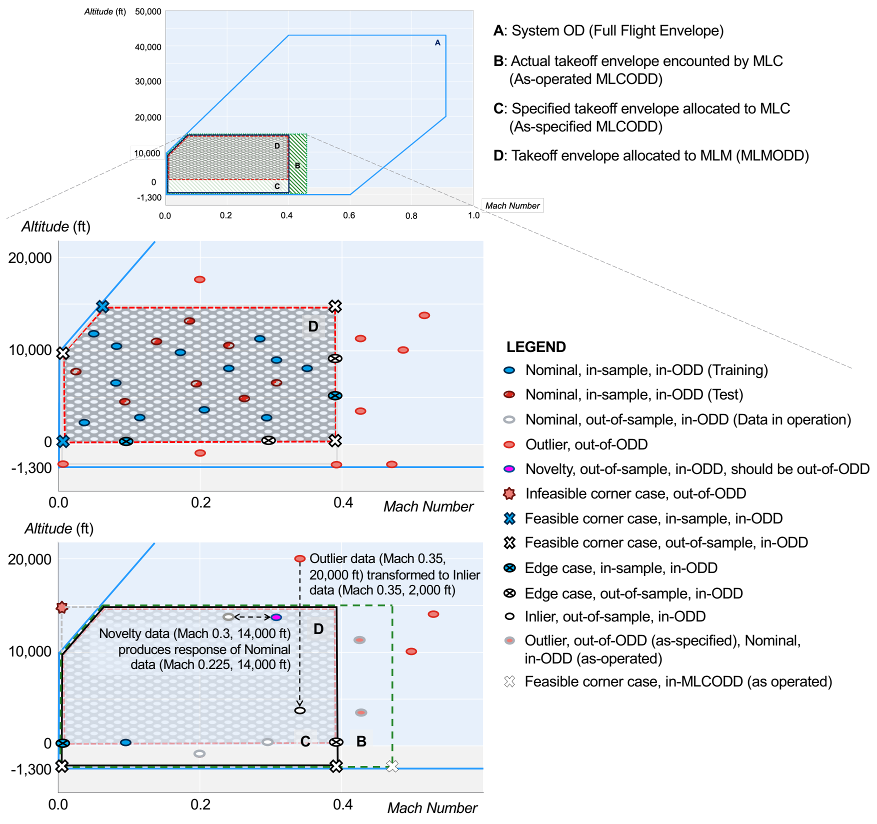

Fig. 1. An example flight envelope (region A) representing an aircraft system OD, whose refinement and allocation to an MLC and MLM give, respectively, an *As-operated* MLCODD (region B), containing an *As-specified* MLCODD (region C), itself containing the MLMODD (region D), for the takeoff regime. The shapes representing the different ODDs are practically congruent, but have been shown slightly offset here to differentiate each from the other. Zoomed-in views of the respective ODDs highlight the different categories and kinds of data used to characterize them.

Consider that a portion of this SOD is allocated to an ML-based subsystem to be used during the *takeoff* flight phase. Its ODD (not shown in Fig. 1) is the takeoff regime at the bottom of the SOD which, in turn, we refine and allocate to an MLC (contained by that ML-based subsystem). The resulting MLCODD parameters are: $0 \leq \texttt{Mach} \leq 0.4$ and $-1300\,\text{ft} \leq \texttt{Alt} \leq 15\,000\,\text{ft}$. In Fig. 1, this *As-specified* MLCODD is the irregular pentagon with the solid dark border (region C).

For this ODD, observe that the upper bound for the airspeed parameter is $\texttt{Mach}\ 0.4$. However, aircraft with greater maximum takeoff weights, e.g., cargo aircraft, can often exceed this bound during takeoff. Thus, there are two possibilities: either the design was to be restricted to non-cargo aircraft, or there is a missing requirement that would be discovered in operation with cargo aircraft. For the latter case, the *as-operated* MLCODD would then have an increased upper bound on airspeed, e.g., $0 \leq \texttt{Mach} \leq 0.5$.

In Fig. 1, this is the irregular pentagon (region B) that includes the earlier *As-specified* MLCODD (region C). Now, further consider that there is insufficient takeoff data for altitudes below sea-level to apply ML. Hence, we restrict the MLM to takeoff operations for `Alt` $\geq 0\,\mathrm{ft}$. Thus, the MLMODD is a sub-region of the MLCODD allocated to the MLM contained in the MLC. In Fig. 1, this is the irregular pentagon with the dashed border (region D), with the same range for `Mach` as its containing MLCODD, but with sea-level as the lower bound on `Alt`. Fig. 1 also zooms into these regions to give examples of the various categories and kinds of data described earlier.

Data inside the MLMODD (and/or MLCODD) can be drawn from the *Nominal*, *Edge Case*, *Feasible Corner Case*, *Inlier*, and *Novelty* data categories. The following observations are noteworthy: first, the MLM must demonstrate generalization from *Nominal*, *In-Sample* training data to *Nominal*, *In-Sample* test data, as well as to *Nominal*, *Out-of-Sample* data, all of which are *In-MLMODD*. Moreover, the MLM must exhibit *correct* behavior (i.e., the behavior meets the allocated requirements) on *Edge Case* as well as *Feasible Corner Case* data.

Next, the preceding data categories are disjoint relative to a specific allocation. For example, *Outlier* data for an MLM cannot also be a *Feasible Corner Case* for that MLM, though it can be one for the containing MLC. In Fig. 1, the data point (`Mach` 0.4, `Alt` $: -1300\,\mathrm{ft}$) is one such example of an *Outlier* for the MLMODD that is also an *Out-of-Sample*, *Feasible Corner Case* for the containing *As-specified* MLCODD.

We associate data points with specific categories relative to an allocation. In Fig. 1 for instance, the data point at (`Mach` 0.1, `Alt` $: 0\,\mathrm{ft}$) is an *Edge Case* for the MLMODD, but is *Nominal* data for the containing *As-specified* MLCODD. Similarly, each of the data points at (`Mach` 0, `Alt` $: 0\,\mathrm{ft}$), and (`Mach` 0.4, `Alt` $: 0\,\mathrm{ft}$) is a *Corner Case* from an MLMODD perspective but an *Edge Case* for the MLCODD.

Likewise, we can have *Outlier* data to the MLMODD that are within the MLCODD. In Fig. 1, examples of this case comprise any data point in the region of the *As-specified* MLCODD not included in the MLMODD, i.e., in the region defined by $0 \leq$ `Mach` ≤ 0.4, and $-1300\,\mathrm{ft} \geq$ `Alt` $> 0\,\mathrm{ft}$. As shown, such points are *Outlier* data for the MLMODD, but can be *Nominal*, *Edge Case* or *Feasible Corner Case* data for the MLCODD. In the same way, points in the rectangular region of the takeoff envelope between $0.4 <$ `Mach` ≤ 0.5 and $0\,\mathrm{ft} \leq$ `Alt` $\leq 15\,000\,\mathrm{ft}$ are *Outlier* data to both the MLMODD, and the *As-specified* MLCODD, but are within the *As-operated* MLCODD. For example, the data point at (`Mach` 0.5, `Alt` $: -1300\,\mathrm{ft}$) is a *Feasible Corner Case* for the *As-operated* MLCODD.

Recall that *Infeasible Corner Case* data are a special case of *Outlier* data that may not be reasonably encountered in operation, where two or more ODD parameters simultaneously take the extreme values admissible for that ODD. Fig. 1 (bottom right) shows one such example: the corner case at (`Mach` 0.0, `Alt` $: 15\,000\,\mathrm{ft}$) is infeasible for both the MLMODD and MLCODD because no airport runways exist at $15\,000\,\mathrm{ft}$ altitude.

Inlier data are within the MLCODD and/or MLMODD due to errors in data processing, scaling, normalization, and usage of incorrect units. In Fig. 1, the *Inlier* data point at (`Mach` 0.35, `Alt` $: 2000\,\mathrm{ft}$) is the result of a data preparation and scaling error of the *Outlier* data point at (`Mach` 0.35, `Alt` $: 20\,000\,\mathrm{ft}$). The result of processing such *Inlier* data is an incorrect response from the MLM, for example a flight control parameter value appropriate for the outlier data point is incorrectly produced at a lower altitude within the takeoff envelope.

Novelty data are within the MLMODD (and thus, also within the MLCODD), but are, in fact, data that should have been Out-of-MLMODD (or MLCODD). *Novelty* data are not excluded from the MLMODD due to an insufficiency in the number and variety of parameters used to specify the MLMODD. In Fig. 1, the data point (`Mach` 0.3, `Alt` : $14\,000\,\text{ft}$) is *Novelty* data producing a response appropriate for the *Nominal* data point at (`Mach` 0.225, `Alt` : $14\,000\,\text{ft}$). This occurs because the SOD and, in turn, the MLCODD and MLMODD have been specified using only two parameters (altitude and airspeed), either ignoring the effect of additional parameters such as air temperature, or implicitly assuming that the operations occur in the same environment as that in which the in-sample data were collected. In this example, operating in warmer air temperatures results in a lower Mach number, due to which the MLM receives an input that is invalid for the operating context, but is nonetheless *Nominal*.

In general, discovering data from the *Inlier*, *Novelty*, and *Outlier* categories that should be part of the required (or intended) MLCODD or MLMODD, occurs either during testing, during validation of the relevant ODDs, or from analysis of the data gathered from in-service experience. That usually results in re-defining the respective ODDs, e.g., by expanding its dimensions by including additional parameters, or modifying the admissible range of existing parameter values.

4 Support for System-level Analysis

The combination of the category and kind of real data in operation, `RDO`, facilitates *partitioning* an MLMODD (and equivalently, an MLCODD) at a higher level than, say, partitioning by equivalence classes of inputs.[6] Then, from a safety standpoint for example, for each such partition we can analyze the contribution of the MLM (or the corresponding MLC) to system hazards in terms of the effects produced in response to inputs drawn from that partition. Examples of such effects include: an underperformance of the MLM; a hazardous failure condition; MLM or MLC malfunction; or, more generally, MLM and MLC failure modes and *hazard contribution modes* [3].

Subsequently, we can establish the (high-level) requirements that an MLM and its containing MLC should fulfill. These can include, for instance, restrictions on MLM behavior, constraints on data processing, limitations of use, as well as requirements necessary to manage the safety impact of the identified effects. The latter, in turn, also informs the selection of the mitigation measures appropriate for sufficient safety assurance. Such mitigations include the application of learning assurance processes (at the MLM layer), architectural mechanisms (at the MLC, ML-based subsystem, and system layers), as well as traditional development assurance processes as appropriate.

The tables given in Fig. 2 and Fig. 3 illustrate how we can use the partitions of an ODD to analyze the impact on an MLC and MLM: the row and column labels for a cell in the table correspond to the kinds and categories of data, respectively, and their combination is the partition of `RDO` we analyze. The content of a cell describes the results

[6] In fact, we can combine those two ways of partitioning an ODD, e.g., by selecting an equivalence class of inputs within a *nominal*, *out-of-sample*, and *In-MLMODD* partition.

KIND OF DATA (Real Data in Operation)			**DATA CATEGORIES**		
			Nominal	**Edge Case**	**Feasible Corner Case (CC)**
In-MLCODD	**In-MLMODD**	**In-Sample**	**E:** MLM underperformance on particular known inputs **A** • Input detection and failover • Input masking/filtering • Input value replacement	**E** • MLM performance degradation • Incorrect MLM response • MLM Malfunction **A** • Extreme value monitoring • Envelope protection and failover **L:** Data augmentation	
		Out-of-Sample	**E:** MLM underperformance in localized regions **A** • Detection of regions of MLM underperformance • Distribution drift monitoring • Input routing/switching to alternative function • MLM output range monitoring and failover • MLM output masking • MLM output value replacement	**E** • MLM performance degradation • MLM malfunction **A** • Extreme value monitoring • Envelope protection and failover • MLM output range monitoring and failover • MLM output masking • MLM output value replacement	
	Out-of-MLMODD		**R:** MLM shall not receive inputs from these data categories **R:** MLC shall receive and process input from these data categories		
			A • Input masking/filtering using pre-processing items of MLC • OOD detection (of Out-of-MLMODD inputs) at ML-based subsystem level • Input routing/switching to alternative function	**A** • Input masking/filtering using pre-processing items of MLC • Extreme value monitoring • OOD detection (of Out-of-MLMODD inputs) at ML-based subsystem level • Input routing/switching to alternative function	
Out-of-MLCODD			**E:** MLC malfunction **R:** MLC shall not receive inputs from these data categories		
			A • Input masking/filtering at ML-based subsystem level • Input routing/switching to alternative function	**A** • Extreme value monitoring • OOD detection (of Out-of-MLCODD inputs) at ML-based subsystem level • Input routing/switching to alternative function	

Fig. 2. Assessing the impact of an ODD on an MLM and the corresponding MLC in relation to the partitions induced by the categories and kinds of real data in operation (specifically the *Nominal*, *Edge Case*, and *Feasible Corner Case* data categories) described in terms of the potential *effects* (**E**) of the data, the *requirements* (**R**) induced, the *learning assurance* (**L**) processes that may be needed, and candidate *architectural* (**A**) options for mitigation.

of a particular analysis for that partition, i.e., the effects of encountering data from that partition, and the considerations that emerge on the requirements, architectural mitigations, and on learning assurance. When the analysis is common to multiple partitions, we show this in a cell that spans multiple columns. Note that these kinds of analyses can be applied to any ML-based subsystem, MLC, or MLM, and is agnostic to their allocated function. Also note that Fig. 2 and Fig. 3 are mainly examples, hence they are not comprehensive or complete. Thus, some effects (and the corresponding architectural options) can be common to the different partitions.

For brevity, here we highlight some specific example options from a combination of analyses. In practice, however, each analysis would be separately undertaken since the identified learning assurance techniques only apply during design, whereas the identified architectural options are primarily relevant in use.

Fig. 2 shows an analysis from a safety standpoint: the potential effects of the ODD partition characterized by *In-MLMODD*, *In-Sample*, *Nominal* data include MLM un-

KIND OF DATA (Real Data in Operation)			DATA CATEGORIES		
			Novelty	Outlier (Including Infeasible CC)	Inlier
In-MLCODD	In-MLMODD	In-Sample	**R**: MLM training data shall not include inputs from these data categories (since functional intent excludes such data) **L**: Data selection and management processes, including pre-processing		
In-MLCODD	In-MLMODD	Out-of-Sample	**E** • Incorrect MLM response (MLM does not meet its requirements) • MLM malfunction **A** • Envelope protection and failover • MLM output range monitoring and failover • MLM output masking • MLM output value replacement **L:** ODD parameter identification	**Excluded by definition:** Outlier and Infeasible CC data are Out-of-MLMODD, therefore they are not In-MLMODD	**E** • Incorrect MLM response (MLM does not meet its requirements) • MLM malfunction **A:** Dissimilar inputs with cross-checking
In-MLCODD	Out-of-MLMODD		• **Excluded by definition:** Novelty data are In-MLMODD, therefore they are not Out-of-MLMODD	**E:** MLM malfunction **R:** MLM shall not receive inputs from this data category **A** • MLC preprocessing based input masking/filtering • OOD detection (of Out-of-MLMODD inputs) at ML-based subsystem level • Input fault flags • Input masking or replacement • Input routing/switching to alternative function **L** • Learning assurance processes shall analyze outlier data for ODD modification.	**Excluded by definition:** Inlier data are In-MLMODD, therefore they are not Out-of-MLMODD
Out-of-MLCODD			**R** • MLC shall not receive inputs from these data categories • ML-based subsystem containing MLC shall receive and process inputs from these data categories **A** • OOD detection (of Out-of-MLCODD inputs) at ML-based subsystem level • Input routing/switching to non-ML items / alternative function		

Fig. 3. Assessing the impact of ODDs characterized by *Anomaly* data, i.e., *Novelty*, *Outlier* (including *Infeasible Corner Case*), and *Inlier* data categories, similar to the assessment in Fig. 2.

derperformance on specific inputs (as observed during training and testing). In some applications, the exact inputs from that partition may also be encountered in operation. Thus, architectural mitigations for such data can include monitoring to detect those specific inputs, together with input value replacement, masking, or filtering, and/or failover.

Similarly, other partitions of the ODD can be characterized by *In-MLMODD*, *Out-of-sample*, *Edge Case* (or *Feasible Corner Case*) data. Fig. 2 shows these combined into a single partition since the high-level effects (such as MLM malfunction), as well as the corresponding architectural mitigations (e.g., extreme value monitoring, or envelope protection and failover) are similar for each. However, we note that for particular applications involving a specific MLM, the individual effects (and therefore the necessary architectural mitigations) from *Edge Case* inputs are likely to differ from those resulting from *Feasible Corner Case* inputs.

Likewise, a common requirement is induced by the ODD partition(s) formed by (each of) the *In-MLCODD*, *Out-of-MLMODD*, *Nominal* data (and *Edge Case*, or *Feasi-*

ble Corner Case data respectively). For example, an MLM shall not receive and process input data drawn from those partitions of the ODD. Consequently, the architectural options available are also largely similar, although extreme value monitoring mainly applies to *Edge Case* and *Feasible Corner Case* data, rather than to *Nominal* data. Additionally, note that for *Out-of-MLCODD* kind of data, there is no distinction between *Nominal*, *Edge Case* and *Corner Case* data from an MLC standpoint. However, those categories are distinct from the perspective of the OD allocated to the containing ML-based subsystem, which induces distinct architectural mitigations as shown.

Fig. 3 shows a similar analysis from a system development standpoint for the *Novelty*, *Outlier* (including *Infeasible Corner Case*), and *Inlier* data categories. Such data are not part of the functional intent, and therefore a requirement on MLM development is to exclude such data for model training. As such, the learning assurance process must include data selection and management activities to assure that the training data indeed excludes inputs drawn from those data categories to preclude an MLM from producing responses that are inconsistent with the functional intent.

The partitions of the ODD characterized by *In-MLMODD*, *Out-of-sample*, *Novelty* (and likewise *In-MLMODD*, *Out-of-Sample*, *Inlier*) data need special attention: specifically, *Novelty* data may not be detectable through operational monitoring. Indeed, if such data could be detected at runtime, the relevant features would then have been included in the set of MLMODD parameters, rendering such data *Nominal* rather than *Novelty*. *Inlier* data are also similarly difficult to detect in operation. To mitigate MLM failure conditions resulting from the former, learning assurance activities are particularly important, especially those facilitating a rigorous and comprehensive identification of MLMODD parameters and features.

In some circumstances, it may be possible to detect and recover from the *effects* of *Novelty* data if the responses produced result in a range violation. For those situations a range of output monitoring, masking, replacement, and failover mechanisms offer an architectural solution to risk mitigation. To mitigate the effects of *Out-of-Sample*, *In-MLMODD*, *Inlier* data, dissimilar and/or independent inputs with cross-checking is a candidate architectural pattern.

An MLM cannot receive *In-MLMODD*, *Out-of-Sample*, *Outlier* data since those are, by definition, *Out-of-MLMODD*. However, in a similar vein to *Novelty* and *Inlier* data, the ODD partition characterized by *In-MLCODD*, *Out-of-MLMODD*, *Outlier* data also needs particular attention: as we saw earlier (Section 3, Fig. 1), it is possible to encounter *Outlier* data that *ought to have been included* in the MLCODD—and by allocation, also in the MLMODD—but were not. This situation can occur due to an error in the requirements, a deficiency in the data collection process, or a lack of knowledge (epistemic uncertainty). This induces a learning assurance feedback step (see Fig. 3) to analyze *Outlier* data to validate and potentially update both the MLMODD and the MLCODD from in-service experience.

5 Conclusions and Future Work

We have clarified the dimensions along which the parameters that define an ODD for an ML-based aeronautical product can be captured, whilst identifying the categories and kinds of data that can be encountered in operation. We have concretized the underlying concepts using an aircraft flight envelope example considering its allocation to an ML Model (MLM) for the takeoff regime. Our data-centric ODD characterization gives a useful framework to identify and organize system development, safety, and assurance activities, which we have illustrated through examples of some high-level effects of the data both on the MLM and its containing ML Constituent (MLC), along with the architectural options available for mitigation.

The work described here has emerged from an ongoing, aviation industry-led, consensus based effort. As such, validating the relevance, applicability, and utility of the underlying concepts and approach largely relies on a committee consensus agreement and, eventually, regulatory endorsement. To that end, aviation industry practitioners are applying the approach to a variety of real-world applications such as airborne collision avoidance [2], safe flight termination[7], and time-based separation of transport aircraft in terminal approaches [4]. These use cases corroborate our earlier assertion (see Section 4) that the work in this paper is sufficiently generic to be applicable to ML-based aeronautical products used both in airborne systems, and for air traffic management/navigation services. As a key avenue of future work, we are committed to take the lessons learned from those validation efforts—of the successes, insights, and possible gaps—to refine and further mature our approach. A related, crucial aspect of our future effort is to leverage the concepts and approach presented here to define a rigorous process for MLCODD development and validation, and MLCODD coverage verification (to be elaborated in a forthcoming paper). Such a process does not yet exist in the prevailing aviation standards and guidance on recommended practices. Thus, it will represent a concrete extension to the state-of-the-practice.

Our data-centric ODD characterization (Section 3), though rigorous, would benefit from a formalization of the identified categories and kinds of data, and their interrelations. This could facilitate assessing whether certain desirable properties hold, e.g., that the data categories *cover* an ODD in some formally defined sense, and that they are internally *complete*. This paper has mainly considered singleton MLMs and MLCs. We intend to extend our approach to the situations of multiple MLCs within a single ML-based subsystem, and multiple sub-MLMs within a single MLC. These cases have interesting safety and architectural implications from which we expect to gain a deeper insight into hazardous behavior emerging from the interactions of multiple MLMs and MLCs. In a similar vein, the support for system-level analysis (Section 4) can be further elaborated towards a more comprehensive and complete description of the potential effects of real data encountered in operation, together with the requirements induced, architectural options available for mitigation, and the learning assurance activities necessary.

This paper has given a new data-centric characterization for ODDs that is not an extension, enhancement, or tailoring of prior automotive domain ODD concepts. A related avenue of future work is to compare and contrast our ODD concept and principles with those of other safety-critical domains such as automotive, healthcare, rail, and space. We remain cautiously optimistic that our work is sufficiently general to be adopted, extended, and applied in those domains by the associated subject-matter experts.

[7] See: https://safeterm.eu/

Acknowledgments We thank the members of the ODD working group of the joint EUROCAE WG-114 and SAE G-34 committees who contributed to the discussions that shaped the concepts and approach in this paper. We are additionally grateful to the anonymous reviewers whose comments aided us in improving the paper.

References

1. BSI Standards Ltd.: Operational Design Domain (ODD) Taxonomy for an Automated Driving System (ADS) – Specification. BSI PAS 1883:2020 (August 2020)
2. Damour, M., et al.: Towards Certification of a Reduced Footprint ACAS-Xu System: A Hybrid ML-Based Solution. In: Habli, I., Sujan, M., Bitsch, F. (eds.) Computer Safety, Reliability, and Security. pp. 34–48. Springer, Cham (2021)
3. Denney, E., Pai, G., Smith, C.: Hazard Contribution Modes of Machine Learning Components. In: Espinoza, H., et al. (eds.) Proceedings of the AAAI Workshop on Artificial Intelligence Safety (SafeAI). AAAI, CEUR Workshop Proceedings (2020)
4. EUROCONTROL: COAST (Calibration of Optimised Approach Spacing Tool) with Use of Machine Learning Models. White Paper V1.1. (April 2021)
5. EASA: First Usable Guidance for Level 1 Machine Learning Applications. EASA Concept Paper Issue 01 (December 2021)
6. G-34, Artificial Intelligence in Aviation Committee: AIR 6988, Artificial Intelligence in Aeronautical Systems: Statement of Concerns. SAE International (April 2021)
7. Irvine, P., Zhang, X., Khastgir, S., Schwalb, E., Jennings, P.: A Two-Level Abstraction ODD Definition Language: Part I. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 2614–2621 (2021).
8. ISO/TC 22/SC 32: Road Vehicles - Safety and Artificial Intelligence. ISO/AWI PAS 8800 (Under development) (2021)
9. ISO/TC 22/SC 33: Road vehicles - Test Scenarios for Automated Driving Systems - Taxonomy for Operational Design Domain. ISO/DIS 34503 - Draft International Standard (2023)
10. Kaakai, F., Adibhatla, S., et al.: Toward a Machine Learning Development Lifecycle for Product Certification and Approval in Aviation. SAE Intl. Journal of Aerospace **15** (2022)
11. Koopman, P., Osyk, B., Weast, J.: Autonomous Vehicles meet the Physical World: RSS, Variability, Uncertainty, and Proving Safety. In: Romanovsky, A., Troubitsyna, E., Bitsch, F. (eds.) Computer Safety, Reliability, and Security. pp. 245–253. Springer, Cham (2019)
12. NHTSA, US Department of Transportation: Automated Driving: A Vision for Safety. Report No. DOT HS 812 442 (September 2017)
13. S-18, Aircraft And System Development And Safety Assessment Committee: ARP 4761, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. SAE International (Dec 1996)
14. S-18, Aircraft And System Development And Safety Assessment Committee: ARP 4754A, Guidelines for Development of Civil Aircraft and Systems. SAE International (Dec 2010)
15. SAE G-34 Committee for AI in Aviation and EUROCAE WG-114 for AI: Process Standard for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing AI. AS 6983 Draft Standard Work In Progress (February 2023)
16. SAE International: Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Surface Vehicle Recommended Practice J3016 (2018)
17. Thorn, E., Kimmel, S., Chaka, M.: A Framework for Automated Driving System Testable Cases and Scenarios. Report No. DOT HS 812 623, National Highway Traffic Safety Administration (September 2018)
18. Underwriter Laboratories Inc.: ANSI/UL 4600 Standard for Safety for the Evaluation of Autonomous Products (April 2020)