EC7.17

# Regulation constraints compliance (Implementation Process Assurance)

**?Document reference number for ANR?**

contact@confiance-ai.fr | www.confiance.ai

**Document reference: XXX**

## Contributors

| | Name | Organisation | Role |
|---|---|---|---|
| Responsible for the deliverable | Cristian MAXIM | IRT-SystemX | Author |
| Scientific responsible | | | |
| Co-authors | Christophe GABREAU | Airbus | Author |
| | | | |
| | | | |

## Document Control

| Revision | Date | Commentary | Author |
|---|---|---|---|
| v1.0 | | | |
| | | | |
| | | | |

# Contents

# A. Introduction

Despite being a topic of active research for several year, there is a lack of experience and methods for securing and certifying ML applications. Apart from the fast progress in ML around research and development, the challenges are rooted in the fundamental programming differences between classic software and ML. Classic software components are usually a combination of manually defined functions, the logic within these functions being explicitly specified by a human. In contrast, the core of ML is to learn from data while the user only specifies the model framework. Often, the result is a highly complex black-box that returns results without explaining the underlying decision-making process. The main challenge of certifying such a system is validating a black-box model representing a highly complex function. Furthermore, an ML algorithm always entails risks related to data. Hence, within the certification process, not only does the proper functionality of the model need to be verified but also the representability of the data for training and testing. This is a challenging task, because the reason for developing ML often is that the underlying problem and cause-effect relations are too complex to be understood, formulated and solved analytically and manually.
Identifying errors in machine learning (ML) development is crucial for ensuring the reliability and accuracy of models. Here are some common errors and strategies for identification: data quality issues, overfitting or underfitting, incorrect feature scaling, model selection issues, hyperparameter tuning problems, leakage of information, inadequate cross-validation, class imbalance, model deployment issues and lack of interpretability. These expels can appear at different stages in the ML implementation from training to deployment.
In this document we focus only on those errors that can appear once the training is finished and the used model is produced. In other words, we only focus on the implementation process. This process transforms requirements, architecture and design into actions that create a system element according to the practices of the selected implementation technology, using appropriate technical specialities or disciplines. This process results in a system element that satisfies specified system requirements, architecture and design.

# B. Standardisation and certification

One key enabler for the application of artificial intelligence (AI) and machine learning (ML) is the establishment of standards and regulations for practitioners to develop safe and trustworthy AI systems. Despite several efforts by different research institutions and governmental organizations, there are currently no established guidelines. The most influential attempt stems from the European Union with the European Commission's AI Act (AIAct (2021)).
Besides the European AI Act there are several other regulatory attempts. The EU has been actively involved in standardization activities related to AI and plans to implement liability rules on AI to renew the existing rules on product liability. The new rules will hold manufacturers and importers of AI-powered products accountable for any defects or malfunctions that may cause harm to consumers (Council of European Union, 2022). Additionally, there are a number of ISO standards either in active development or already in use that target AI systems in particular. The ISO/IEC JTC 1/SC 42 is the technical committee within the International Organization for Standardization (ISO) that is responsible for developing standards for AI. Currently, there are 57 standards planned, some of which are already published. An overview can be found on the web page of the committee. Among the most important ones are: the ISO/IEC 5259x (2023) series on data quality, ISO/IEC DIS 5338 (2023) on AI system life cycle processes, ISO/IEC FDIS 8183 (2023) on a data life cycle framework, ISO/IEC 23894 (2023) on risk management for AI systems and, ISO/IEC DIS 42001 (2023) for AI management systems.

In the automotive domain, the main recommendations about *functional safety*[1] are gathered in the ISO 26262 series of standards ("Road vehicles - Functional Safety"), a declination of the IEC 61508 addressing the specific needs of electrical and/or electronic systems embedded within road vehicles.
The ISO 26262 gives procedures and measures to prevent and handle failures resulting from random hardware faults or systematic HW/SW faults. It is complemented by the ISO 21448 "Safety of the Intended Functionality" to address *hazardous behaviour caused by the intended functionality or performance limitation of a system that is free from the faults addressed in the ISO 26262 series*. Quoting the standard, these limitations include:

- *The inability of the function to correctly comprehend the situation and operate safely [...]*
- *Insufficient robustness of the function with respect to sensor input variations or diverse environmental conditions*

Considering the definition of the "embarcability" problem, the analysis of certification standards should normally consider all HW and SW development activities. In the aeronautics domain, this would mean considering all objectives from the ED-12C/DO-178C or ED-80/DO-254 standards. In order to anticipate future EASA guidance and requirements for safety-related machine learning (ML) applications, EASA proposed a concept paper in December 2021 EASA (2021). In the search for strategies to verify CNN-based systems against hardware faults at the inference stage, EASA launched the project titled "Concepts of Design Assurance for Neural Networks" (CoDANN) Daedalean (2021a) and its suite Daedalean (2021b). These projects examine the challenges posed by the use of neural networks in aviation, in the broader context of allowing

[1] Functional safety is defined as the *absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems*

machine learning and more generally artificial intelligence on-board aircraft for safety-critical applications.

In the ISO/IEC DIS 5338 (2023) document on AI system life cycle processes, the details related to implementation process can be found in the chapter 6.4.9. The AI-speciffic particularties mentioned in the standard are the following:

- AI systems can be seen as traditional software systems that apply one or more AI models, and therefore their implementation involves the same practices, with some particularities that also introduce new elements. An example is the general best-practice of working with an actively maintained list of agreed- upon work items (backlog). Involving the AI work in such a backlog facilitates cross-disciplinary coordination, planning and evaluation.
- The AI model is typically part of an application that, apart from the model, is developed without any machine learning or knowledge engineering. Because data and knowledge each have very specific roles in AI, this document introduces AI model engineering as a part of the implementation process and also introduces a separate process for AI data engineering. Data and model engineering are closely related, and many implementation activities combine both parts, yet they are different in nature.
- In the case of machine learning, Al model engineering requires training a model using training data. This is an iterative optimization in which the type of model is selected, its hyperparameters are configured and changed until the model performs adequately on the training dataset. Therefore, the AI data engineering process typically interacts with AI model engineering and often strongly relies on experience of the experts involved. Automated machine learning (AutoML) is an approach that the whole or part of these processes are automated, to reduce this reliance and to make the work more efficient. Efficiency can also be gained by, where possible, distributing the work between experts and computer resources to experiment in parallel. Depending on the algorithms used, and the application of AutoML, model training and other optimizations possibly need substantial computing power and waiting times.
- When a model performs in line with either exceptions or predefined specifications, or both, it can be further tuned using validation data and then tested using test data.
- A special type of model engineering is transfer learning, in which an existing machine learning model is used as a starting point to further train for a slightly different use case. By building on previous model engineering success, efficiency gains can be made.
- For implementation, existing software frameworks can be built upon. These frameworks typically offer various built-in Al models and solutions for data processing, model training, testing and orchestration.
- In the case of knowledge engineering, model engineering is the specification of knowledge in declarative or procedural form. The knowledge is acquired from either experts (knowledge elicitation) or through data analysis, or both.

Embedding machine learning in cyber-physical systems (CPS) introduces unique challenges and potential errors due to the integration of physical processes with computational systems. Here are some common errors and challenges associated with implementing machine learning in cyber-physical systems:

- Real-time Constraints related errors: Delays in model inference or decision-making can lead to suboptimal or unsafe outcomes, especially in applications requiring real-time responses.
- Uncertainty in Physical Environments: Machine learning models may struggle to adapt to

uncertainties and variations in the physical environment, leading to inaccurate predictions.

- Sensor and Actuator Failures: Failures in sensors or actuators can lead to inaccurate input data or unsafe control actions based on faulty predictions.
- Data Quality and Availability related errors: Poor-quality or unavailable data can impact the performance of machine learning models, especially in situations where real-world data is noisy or incomplete.
- Model Drift: Over time, the relationship between input data and the physical system may change (concept drift), leading to model degradation.
- Integration Challenges: Integrating machine learning models into existing cyber-physical systems can be challenging, leading to compatibility issues and system disruptions.
- Resource Constraints related errors: Limited computational resources in cyber-physical systems may constrain the complexity and size of machine learning models.
- Regulatory non-Compliance: Failing to comply with industry regulations or standards can lead to legal and safety issues.

When embedding machine learning in cyber-physical systems, it's crucial to perform thorough testing, validation, and continuous monitoring to identify and mitigate potential errors. Additionally, involving domain experts, conducting risk assessments, and adhering to best practices for safety-critical systems can contribute to the successful integration of machine learning in CPS.

# C. Safety argumentation

## C.1. Setting the scene

- Regulatory expectations and classical mitigation means
- Focus on implementations errors
- Guidance for mitigation

## C.2. Aeronautical regulatory context

### C.2.1 Introduction

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.
Specifically in the airborne context, a safety-critical system cannot be certified as long as it is not demonstrated that this system safely performs its intended function under all foreseeable operating and environmental conditions. In the classical way of developing, the system design follows an ARP4754A-compliant SAE/EUROCAE (2010) process whereas associated SW and HW items are developed following ED-12C/DO-178C RTCA/EUROCAE (2011) and ED-80/DO-254 RTCA/EUROCAE (2000) standards.
All these existing standards have been recognized as acceptable means of compliance to regulation requirements for a long time. When it comes to the development of Machine Learning (ML) based systems, there is no such recognized guidance to make it. Therefore, the WG-114/G-34 joint standardization group of EUROCAE/SAE is currently in charge of producing the standard for the certification/approval aeronautic products using ML techniques (ED-324/ARP6983). In particular, the standard has defined novel learning assurance objectives covering the whole spectrum of the development of ML-based systems.
In the context of EC7, we focus on the implementation aspect of the development and therefore, this study is about identifying, detecting and mitigating any error which can be introduced during the transformation of a designed model into an inference model hosted on a target platform.

### C.2.2 Error concept

With respect to the definitions stated in the DO-178C RTCA/EUROCAE (2011), a software error is a *mistake in requirements, design or code*. It may lead to a fault (*manifestation of an error in software*) that may cause a failure of the system it supports, i.e. *the inability of a system or system component to perform a required function within specified limits*. Refer to Figure C.1).
When you introduce ML techniques in the software development, then new type of errors should be considered.

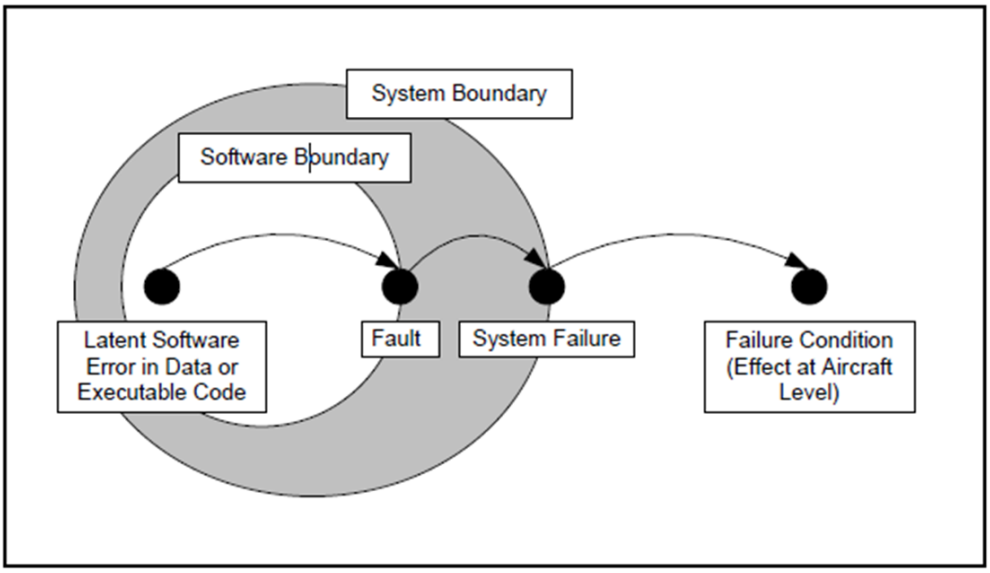


**FIGURE 2-2: SEQUENCE OF EVENTS FOR SOFTWARE ERROR LEADING TO A FAILURE CONDITION**

Figure C.1: Figure from RTCA/EUROCAE (2011)

The ML component, or ML Constituent (MLC) as per WG-114/G-34, is composed of ML model(s) and associated pre/post data processing that can be deployed on one or several *items*. The use of the term "item" complies with the definition given in the existing airborne system development guidance ED-79A/ARP4754A SAE/EUROCAE (2010). The MLC is an item that may contain other items: it is an *items-container*.
The implementation of a MLC as an items-container may lead to the introduction of latent errors either in the data or executable code (of its conventional HW/SW part), or in the data used for machine learning, or in the executable code/implementation of the ML model(s) and its integration with the aforementioned conventional HW/SW items. Some errors may be introduced during the MLC design process that were not detected and removed, due to weaknesses in the learning assurance process applied to the data/datasets or to the ML model design (training and optimization). Some errors may be introduced during the implementation phase and these are the errors that we will address in this chapter.

## C.3. ML implementation errors

### C.3.1 ML implementation process

The scope of this section is to focus on the implementation process part of the MLC development, specified by both the MLMD and the pre/post processing requirements validated during the design phase (refer to Figure C.2 extracted from ED-324/ARP6983 draft 5B). So, at this stage, the ML model(s) have been trained, validated and verified against the ML data and model requirements. The ML model(s) will be hosted onto SW/HW item(s) identified in the MLC architecture design process and specified through the decomposition of the MLMD into MLMID(s). Then, the ML-based Item Implementation & Verification process will transform

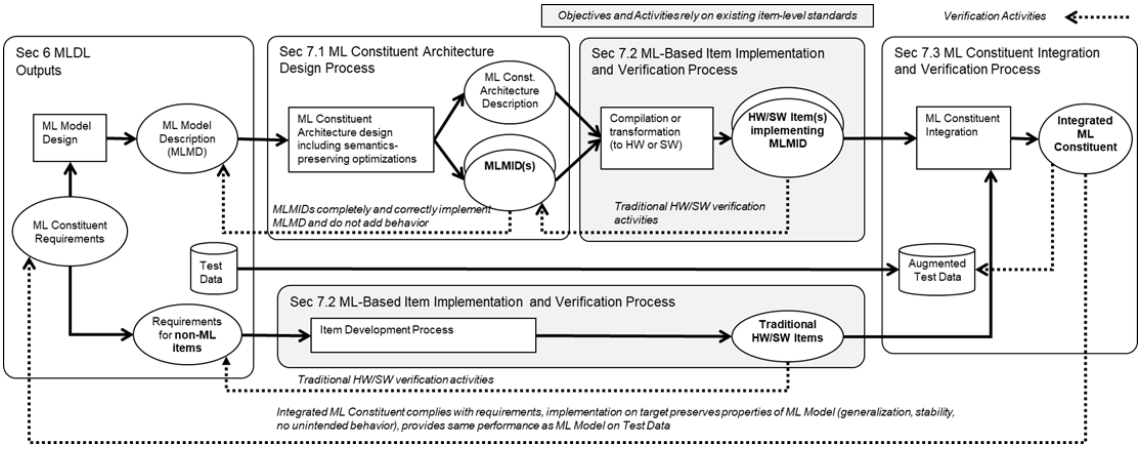


Figure C.2: WG-114- ML Constituent implementation and verification process

the ML model from its design representation (MLMD and MLMID) to the "*inference*" model and ultimately to its target representation (called "*deployed*" model). Eventually, the MLC integration and verification process will have to demonstrate that these transformations **have not introduced errors** and therefore altered the ML model properties that could be detrimental to the safety objectives. Again the scope is to figure out how errors can be introduced when transforming the "*design*" model into the "*deployed*" model, so only the implementation aspect of the (architecture design and item implementation) above processes will be scrutinized.

### C.3.2 Where implementation processes can introduce errors

Indeed the semantics of the designed ML model can be altered by any error introduced either by the very last step of the design (ML model description) or the transformations performed during the implementation steps. A list of potential errors related to these steps are detailed below (the list is not supposed to be comprehensive however relevant to ML specificity):

1. *Incorrect ML model description* - The design ends up with the expression of the implementation needs (MLMD and processing requirements). The objective of this description is to specify all the characteristics, behaviours and performances of the designed ML model(s) to be implemented. This description may contain errors that can lead to model incorrect predictions. This ML model description contains the following items:
   (a) *ML model graph* - The graph of the ML model(s) should be described to such an extent that its semantics is unambiguous otherwise misinterpretation can lead to the introduction of errors during the specific target implementation. For instance, NN convolution operations implemented by the native framework such as Tensor flow or PyTorch can eventually be implemented a different way on the final target platform.
   (b) *Control flow* - The execution flow of the inference ML model could be different (due to implementation constraints) from the one executed in the context of the design environment and lead to incorrect execution.
   (c) *Pre/post model/data processing requirements* - The necessary treatments before and after the model computation should be correctly defined to not introduce any error in the implementation of the intended function supported by the ML model(s).
2. *Incorrect implementation requirements*

(a) *Interface requirements* - The interfaces with the resources of the selected target platform should be correctly described (e.g. libraries, drivers, OS...). The user interfaces and the limitations should be known to preclude interface errors.
(b) *Implementation constraints* - Some low-level design may have to be considered to efficiently adapt the model(s) execution to the host platform resources capabilities (e.g. initialisation and sequence can be very specific to optimise the execution time).

3. *Model optimisation error* - Transformations can be done during the implementation in order to improve the efficiency of the ML model inference on the target and respect the operational constraints linked to the target (timing and memory resources). These transformations are sources of errors when not correctly realized. Examples of some optimization techniques are listed below:
   - Network pruning: removing the neurons or weights of weak importance in order to reduce the memory footprint of the network.
   - Parameter quantization: reducing the number of bits used to represent weights and bias.
   - Algebraic optimization: mathematical operations can be optimized while producing the same results. For instance, in case of a Convolutional Neural Network, Winograd optimization (reducing the number of multiplication) preserves the semantic of convolutions.
   - Fusing batch normalization and convolution in runtime: saving computational resources and simplifying the network architecture at the same time.
4. *Incorrect architecture decomposition (from MLMD to MLMIDs)*
   (a) The decomposition of the MLMD into MLMIDs should be correctly performed so that the MLMD semantics is not altered, leading to incorrect model computation outcomes.
   (b) The interfaces between the MLMIDs are not properly implemented
5. *Incorrect item implementation*
   (a) *Incorrect conversion (from MLMID to code)* - The MLMID format is independent from the platform constraints. A conversion is then needed to transform the MLMID in a representation compatible to the target platform (e.g. C source code). This step may involve the simplification of the graph components and the removal of parts that are not needed to the inference in the operational environment. Whatever the changes are, they should not modify the model semantics and so impact the model behaviour and performance.
   (b) *Numerical representation* - The on-target inference may require the use of numerical representation of the weights/bias of the ML model that are different from the design environment. This constraint may alter the semantics of the design model and introduce model(s) misbehaviour.
   (c) *Floating-point errors* - The use of floating-point arithmetic and numbers when computing predictions on target is source of many errors (e.g. cancellation, absorption, rounding, comparison). Refer to RTCA/EUROCAE (2012) section 4.17 for details.
   (d) *Integration error* - The integration of the software on the target using the platform tool chain (compiler, linker) can introduce some implementation errors (e.g. compiler warnings, incorrect hardware addresses, memory overlaps).

## C.4. Guidance for mitigation

### C.4.1 Certification assumptions

The assurance case concept used in this document has been previously described in Christophe Gabreau (Airbus). It is developed as per the GSN v3 notation (refer to ACWG (2021) guidance). The assurance case is based on the development assurance objectives developed in the ED-324/ARP6983 draft 5B. Anticipating that the standard will be recognized as an acceptable Means of Compliance (MoC), this document assumes that the demonstration of the compliance with the standard objectives will demonstrate the conformity with the regulation text (CS 2x.1301/1309). The following assurance case develops the demonstration that the ML Constituent (MLC) performs its intended function at acceptable level of safety for the allocated item Design Assurance Level (iDAL). The iDAL is determined by the safety assessment process (performed at system level), it is based upon the contribution of software to potential failure conditions and leads to the modulation of the applicable objectives.

### C.4.2 MLC implementation Assurance case

The complete certification argumentation (related to MLC development) deals with 5 main objectives as per figure C.3:

1. MLC requirements: *The MLC requirements are a satisfactory refinement of the allocated system requirements for the selected iDAL*
2. MLC design: *The ML Models and the data processing requirements are a satisfactory refinement of the MLC requirements for the selected iDAL*
3. MLC implementation into items: *The ML models and the data processing requirements are satisfactorily implemented into items for the selected iDAL)*
4. MLC integration and verification: *The integrated MLC is a satisfactory implementation of the MLC requirements for the selected iDAL*
5. Confidence: *The evidences which support the MLC conformity demonstration to ED-324/ARP6983 standard objectives are adequate for the selected iDAL*

This document only details the assurance case on the implementation part of the MLC certification argumentation:

- It starts with the specification of the ML model(s) to be implemented (MLMD) which is actually the end of the design phase (part of the objective #2)
- Then it deals with objectives #3 and #4 with respect to Figure C.2. The argumentation to fulfill these objectives are detailed underneath (figure C.4).

### C.4.3 MLMD considerations

This subsection is linked to the format of the ML Model Description (MLMD) as established per the outcomes of the action sheet EC7.18. Indeed a correct expression of the MLMD as per EC7.18 guidance will mitigate the risks to introduce the errors defined in section C.3.2.1(a), (b) & (c). This is covered by the context statement "*The implementation requirements are fully established within the ML Model Description (MLMD)*" in figure C.4.

### C.4.4 MLC architecture

This subsection covers the demonstration that the MLC architecture is correctly developed with respect to the ED-324/ARP6983 guidance (cf figure C.5): "*The MLC architecture into MLMIDs*

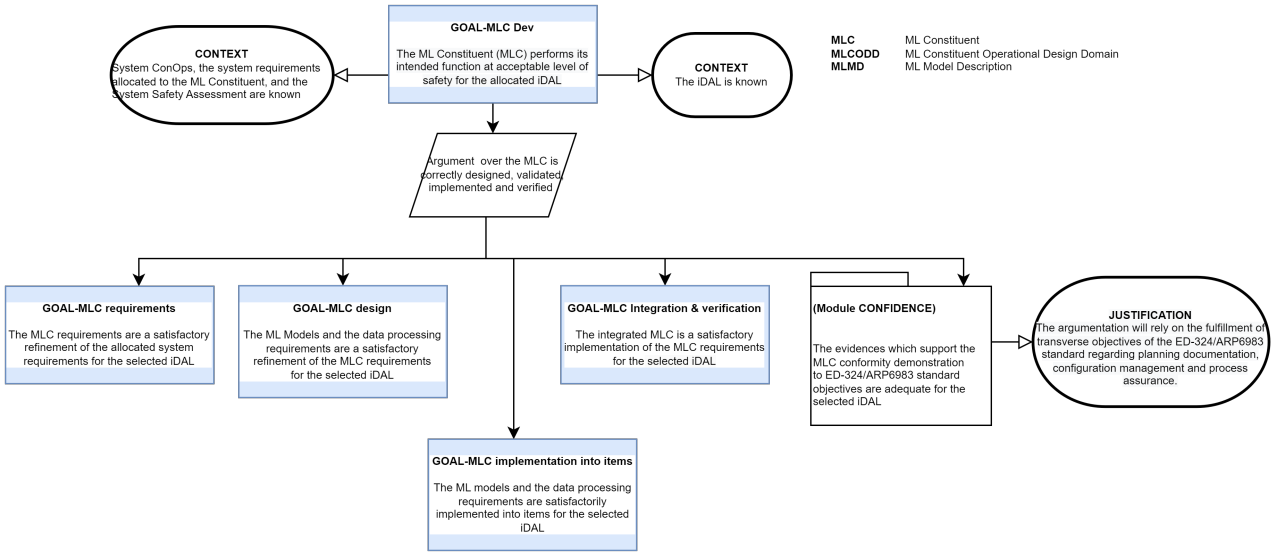


Figure C.3: MLC upper level Assurance Case

*is developed to the ED-324/ ARP6983 guidance for the selected iDAL*". This main objective is decomposed into 4 objectives related to the description and the validation aspects of the architecture development.
*Note: There is a colour code, the green objectives are iDAL D and the red objectives have to be added to fulfill the iDAL C.*

The *description aspect* is developed in the assurance case of the figure C.6 under the main objective: "*The MLC decomposition into items (MLMD into MLMIDs) is satisfactorily documented for a DAL D/C*"
This objective is decomposed into 5 sub-objectives:

1. *The MLC architecture is developed* - This objective requests the development of the architecture from the implementation inputs.
2. *The MLMIDs are developed* - This objective requests the development of one MLMID for each item identified in the the defined architecture.
3. *The MLC Implementation Requirements are refined when needed and allocated to items* - This objective is linked to the specification of the interfaces with the hardware resources of the host target.
4. *The MLC architecture derived requirements are identified, documented and provided to the system safety process and to the ML model design process* - This objective is linked to the identification of any emerging behavior due to the integration on the host target.
5. *Any ML Model optimization applied in the MLMIDs are explicitly documented* - This objective is linked to the description of any optimisation aspect.

The *validation aspect* is developed in the assurance case of the figure C.7 under the main objective: "*The MLC architecture elements are a satisfactory refinement of the MLMD and the implementation requirements for a DAL D/C*".
This objective is decomposed into 9 sub-objectives. 5 of them are particularly relevant to the detection of any semantics alteration that the architecture into MLMIDs can bring.
These 5 sub-objectives are:

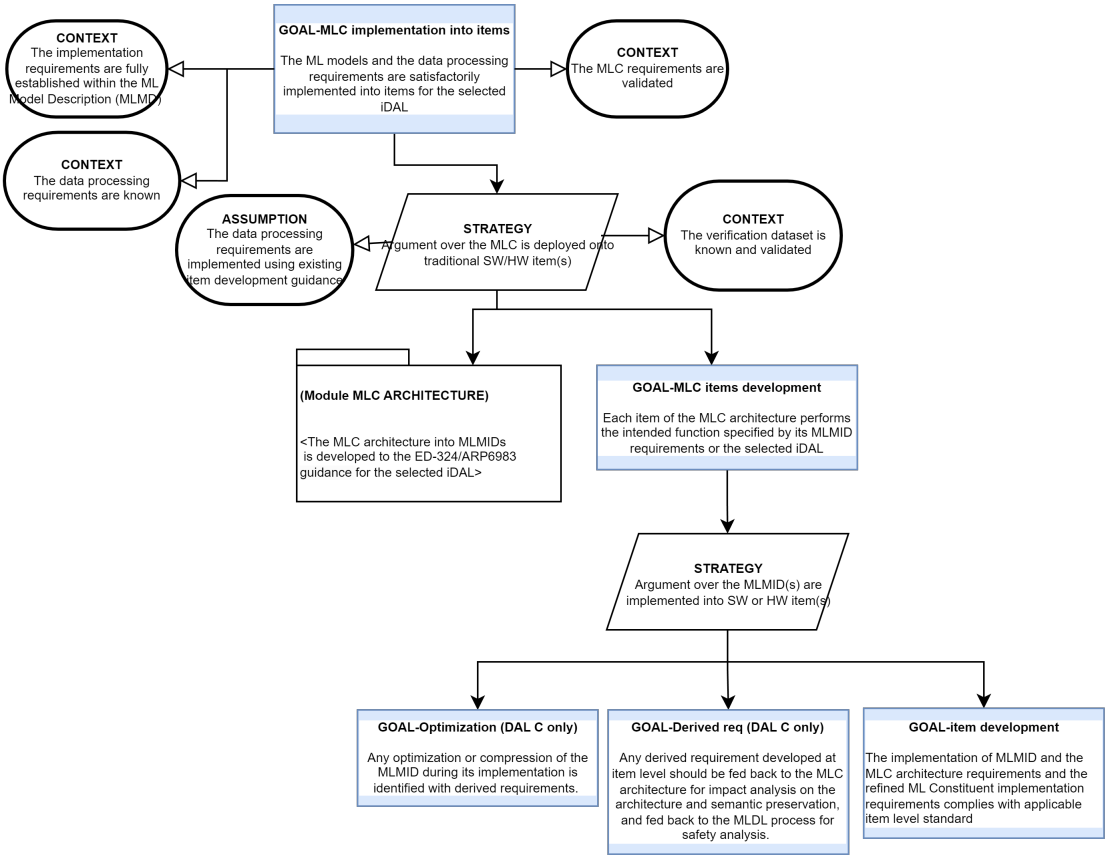


Figure C.4: MLC Implementation Assurance Case

1. *MLMID requirements comply with MLMD requirements*
2. *MLMID requirements are accurate, unambiguous and complete*
3. *MLMID requirements are traceable to MLMD*
   These 3 objectives should ensure that implementation requirements or any derived requirements are not detrimental to the MLMD semantics and covers the errors defined in section C.3.2.2(a) & (b).
4. *MLC architecture complies with MLMD* - This objective should ensure that any optimization of the architecture is consistent with the MLMD and cover the errors defined in section C.3.2.3.
5. *MLC architecture is consistent* - This objective should ensure that the union of the MLMIDs makes the MLMD without altering the semantics of the original ML Model(s) and therefore, cover the errors defined in section C.3.2.4(a) & (b).

### C.4.5 Item implementation and integration

The use of the existing development assurance standards (RTCA/EUROCAE (2011) and RTCA/EUROCAE (2000)) guarantees a proper implementation of the items with respect to their specification (MLMID). Therefore they cover the errors defined in section C.3.2.5.

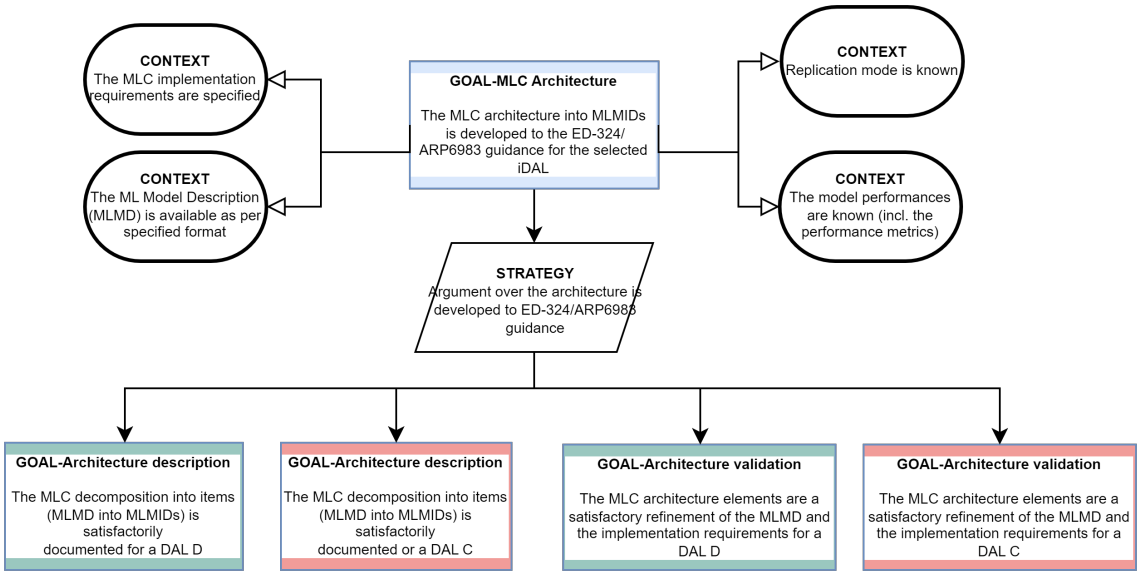


Figure C.5: MLC architecture Assurance Case

### C.4.6 MLC integration and verification

To be developed in Batch 4.

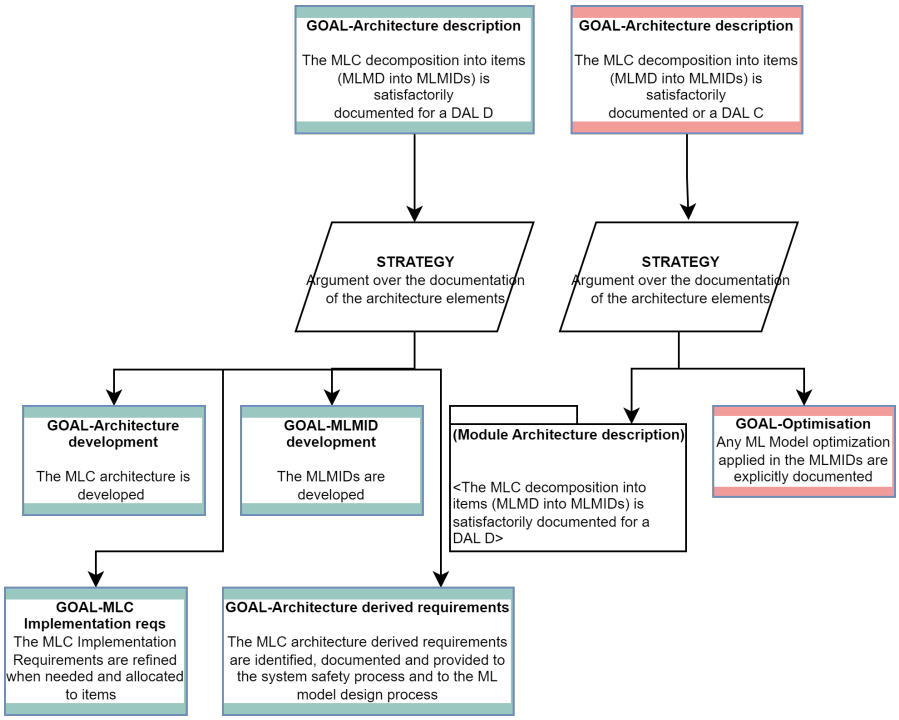


Figure C.6: MLC architecture Assurance Case: *Description aspect*

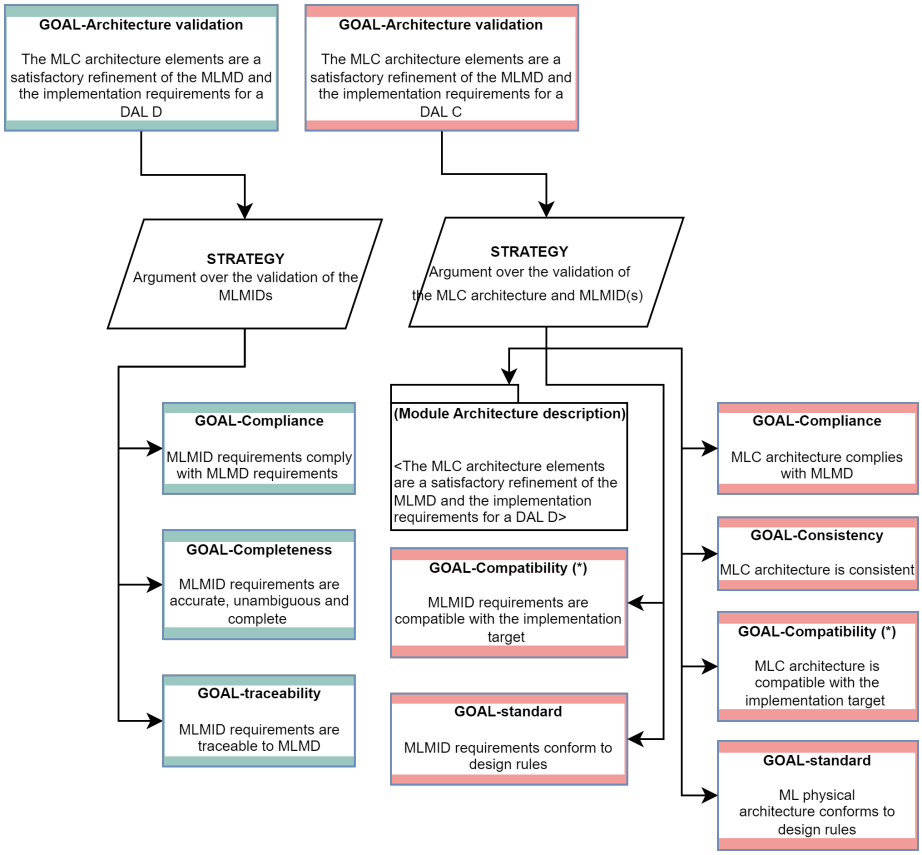


Figure C.7: MLC architecture Assurance Case: *Validation aspect*

# D. Perspectives and conclusion

*Feel free to organize this content according to sub-chapters levels as you wish.*

# Bibliography

ACWG, A. C. W. G. (2021). The goal structuring notation community standard version 3.

AIAct, E. C. (2021). Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts.

Christophe Gabreau (Airbus), Adrien Gauffriau (Airbus), F. D. G. A. J.-B. G. D. C. P. O. (2022). Toward the certification of safety-related systems using ml techniques: the acas-xu experience.

Daedalean, E. a. (2021a). Concepts of design assurance for neural networks (CoDANN).

Daedalean, E. a. (2021b). EASA and daedalean, concepts of design assurance for neural networks (CoDANN) II,.

EASA (2021). EASA concept paper: First usable guidance for level 1 machine learning applications.

RTCA/EUROCAE (2000). DO-254/ED-80 - Design Assurance Guidance For Airborne Electronic Hardware.

RTCA/EUROCAE (2011). DO-178C/ED-12C - Software Considerations in Airborne Systems and Equipment Certification.

RTCA/EUROCAE (2012). DO-248C/ED-94C - supporting information for ed-12c and ed-109a.

SAE/EUROCAE (2010). Aerospace Recommended Practices ARP4754a/ed-79a- development of civil aircraft and systems.

Confiance ai

Title: Title in english

Keywords: Keyword 1, keyword 2

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Our partners:

Air Liquide, AIRBUS, Atos, cea, Inria, NAVAL GROUP, GROUPE RENAULT, SAINT EXUPÉRY, SAFRAN, sopra steria, SystemX, THALES Building a future we can all trust, Valeo, Investir l'Avenir