SONNX – KOM

ONNX "safety-related profile" Workgroup

September 25th 2024

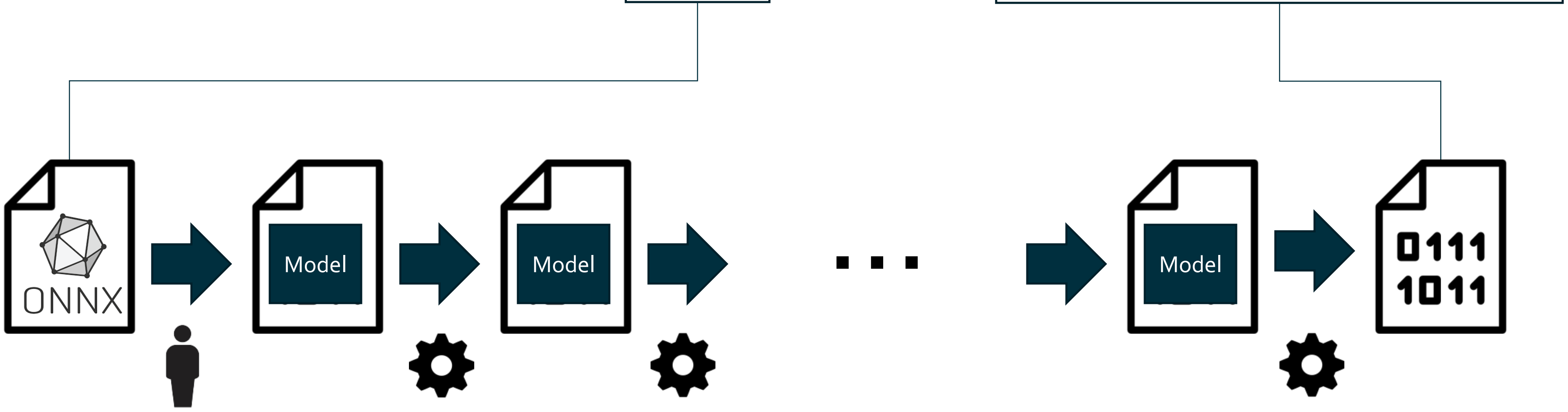
Eric JENN[1], Jean SOUYRIS[2] co-chairs
Christophe GABREAU[2], Nicolas VALOT[3]
[1] IRT Saint Exupery, Toulouse, France, [2] Airbus, Toulouse, France, [3] Airbus Helicopters, Marignane, France

Agenda

- Introduction: main objectives of the workgroup
- Why?
  - An example of regulatory requirements: the ARP-6983
  - A few examples of issues posed by the current standard
- What?
  - (An example of) requirements
  - (An example of) results
- How?
  - The work plan and the organization of the working group
  - Some of the "hard points" that we have already identified
- What about you?
- And now?

Our main objective at a glance

1. We need to demonstrate that the ML model is correctly implemented on the target



2. To be able to ensure / verify correctness, we need a non-ambiguous description of the model (see later in the presentation)

3. ONNX specification is formally insufficient (see later in the presentation), we need to improve it

Where do we come from? Who are we?

- Initially (<10 persons)
  - A group of people from (aeronautical) industry and academia dealing with ML for Safety related embedded systems. Context: confiance.ai programme.
  - Workgroup presented during ONNX Meetup (2024/06/28) and accepted by ONNX steering committee (2024/07/18)
- Today
  - Labs :
    - CEA, INRIA, IRT Saint-Exupery, ISAE SupAero, ONERA, TUM
  - Industry:
    - Aeronautics : Airbus Helicopter, Airbus Operations, Airbus Protect, Embraer, Safran Electronics and Defense, THALES AVS, TAHELS Research and Technologies, DGA-TA
    - Space : Airbus Defence and Space
    - Automotive : Bosch, Ampere
    - Naval: Naval Group
    - Industry : Trumpf, Crosscontrol
    - Other: SopraSteria, Mathworks, Infineon

References (sample)

- Christophe Gabreau *et al*, A study of an ACAS-Xu exact implementation using ED-324/ARP6983, ERTS 2024, Toulouse, France, https://hal.science/hal-04584782
- Gauffriau *et al*, Formal Description of ML models for unambiguous implementation, ERTS 2024, Toulouse, France, https://sciencespo.hal.science/ERTS2024/hal-04167435v2, https://hal.science/hal-04588599
- Vincent Mussot *et al*, Assurance Cases to face the complexity of ML-based systems verification, ERTS 2024, Toulouse, France
- Dumitru Potop Butucaru *et al*, "Bidirectional Reactive Programming for Machine Learning", https://arxiv.org/abs/2311.16977
- Delseny *et al*, White paper "Machine Learning in Certified Systems", see https://arxiv.org/pdf/2103.10529
- Jenn et al, Identifying Challenges to the Certification of Machine Learning for Safety Critical Systems, ERTS 2020, Toulouse, France
- Michele Alberti *et al*, CAISAR: A platform for Characterizing Artificial Intelligence Safety and Robustness", https://arxiv.org/abs/2206.03044
- Iryna De Albuquerque Silva *et al*, ACETONE: Predictable Programming Framework for ML Applications in Safety-Critical Systems, 24th Euromicro Conference on Real-Time Systems (ECRTS 2022), Jun 2022, Modena, Italy.

ONNX

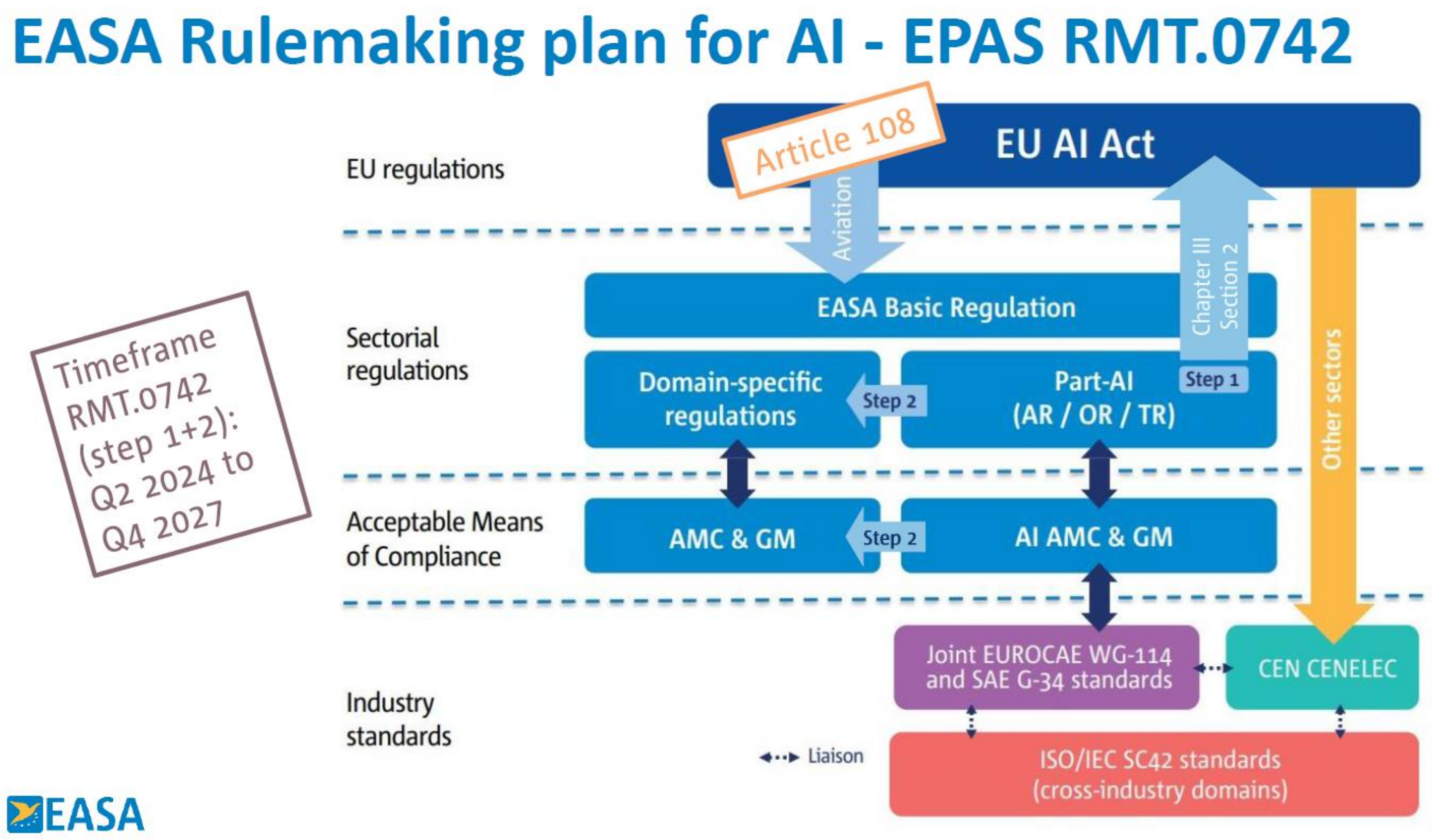
Why?

- An example of regulatory requirements: the ARP-6983
- Some problems of the ONNX standard

What is at stake?

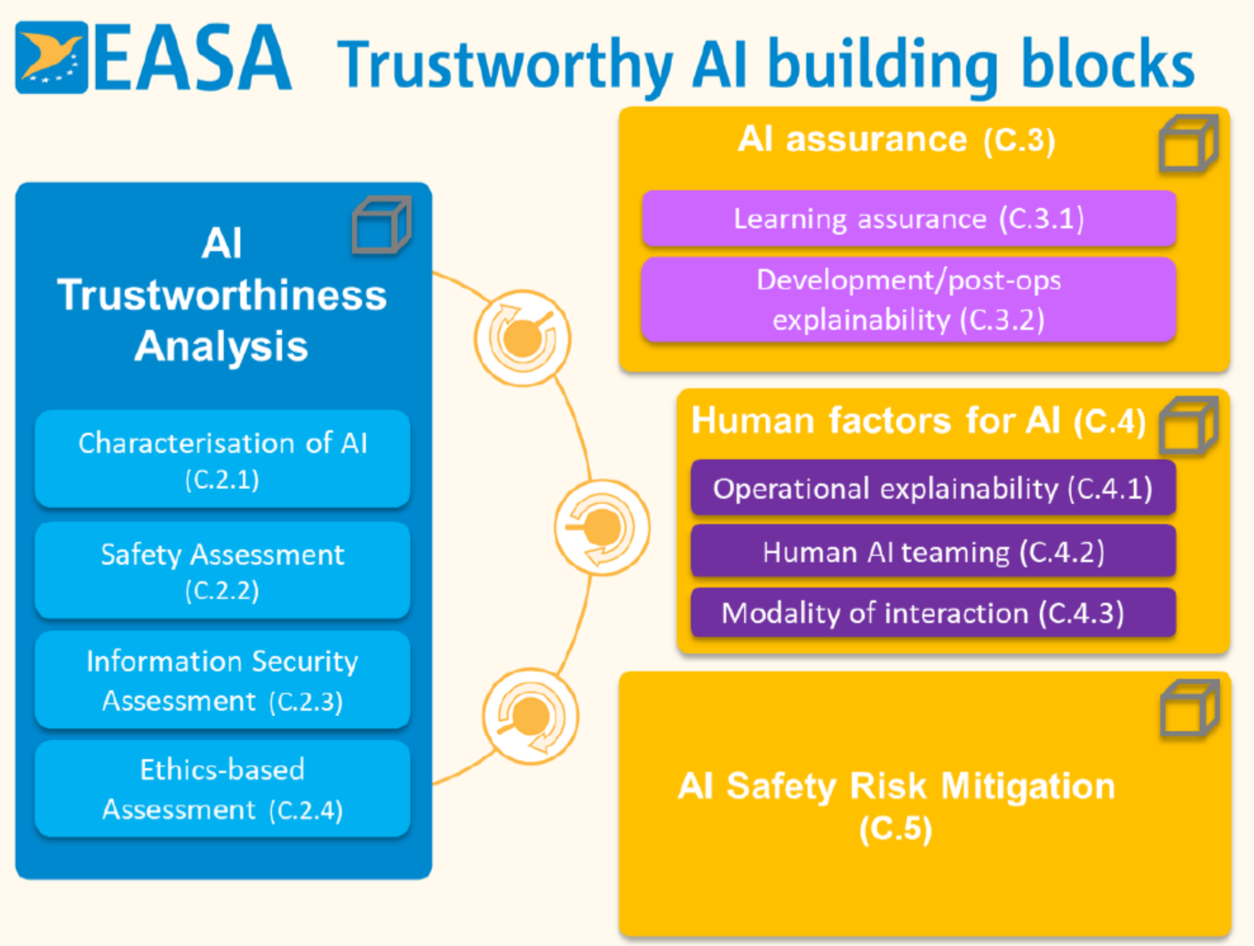
1. From EU AI act to regulation
2. Certification of aeronautical products (e.g. aircraft, drones) using ML techniques
3. Compliance with standardized process (ARP6983/ED-324)
4. Specification (without ambiguity) of the ML algorithm for implementation purposes

From EU AI act to Regulation



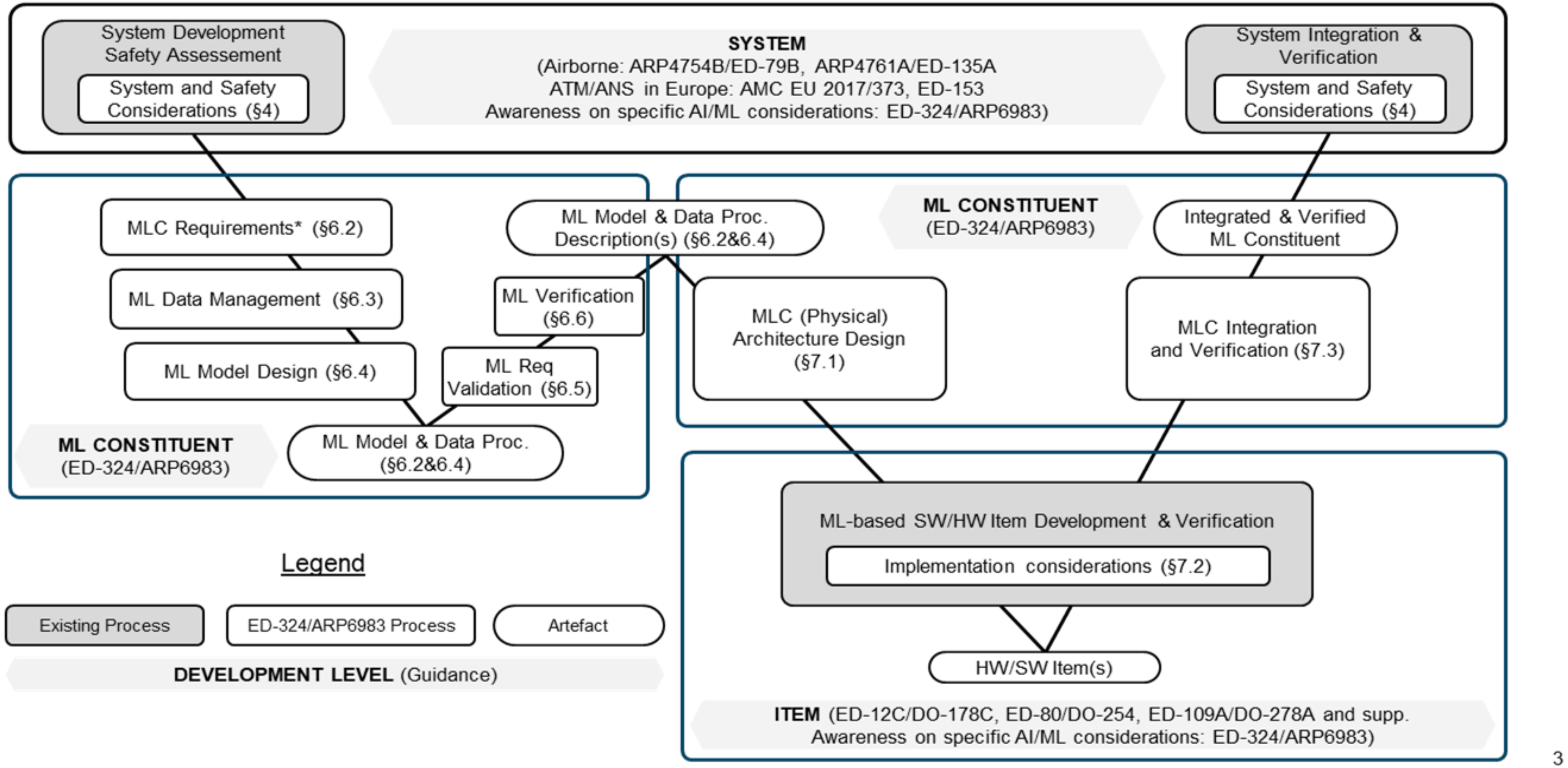
Certification of aeronautical products (e.g. aircraft, drones) using ML techniques

- An aircraft cannot fly without certification approval (Conformance to regulation text)



Compliance with standardized process (e.g. ARP6983/ED-324)

- Being compliant with a standard is a way to demonstrate the conformity



ONNX

Specify (without ambiguity) the ML algorithm for implementation purposes

- Fully describe the model semantics to allow for [exact] implementation

AEROSPACE RECOMMENDED PRACTICE	ARP6983™	REV. DRAFT 6-0
	Issued xxxx-xx Revised xxxx-xx Reaffirmed xxxx-xx Stabilized xxxx-xx Cancelled xxxx-xx Superseding XXXX	

Recommended Practice for Development and Certification/Approval of Aeronautical Safety-Related Products Implementing ML

Issue 1: Non-adaptive Machine Learning in Supervised Mode

6.4.3.6 ML Model Description

When the design of the ML Model is completed, its parameters and hyperparameters are fixed. To enable the correct implementation of this "frozen" ML Model, the ML Model description activity develops sufficient documentation for each ML Model of the ML Constituent. This activity includes the following steps:

d. The analytical/algorithmic syntax and semantics of the ML Model, including all ML Model internal operation that are necessary to compute the output(s) of the ML Model from its inputs, are described in an unambiguous manner in the ML Model description to facilitate their implementation.

- Exact replication: In this first case, the ML Model description should contain sufficient details on the ML Model semantic to fully preserve this semantic in the implemented ML Model. For example, an exact replication criterion may be the direct and faithful implementation of the ML Model description so that the implemented ML Model meets the same performance, generalization, stability, and robustness requirements.

Conclusion: A need for a specific « safety-related » profile

We need to extend the current ONNX format to enable the

- The non ambiguous specification of a designed ML model
- The implementation of a designed model on an embedded target
- The integration of a ML model in safety-related systems
- The certification of aeronautical products using ML techniques

A sample of issues regarding ONNX

So

our objective is to extend/improve ONNX…

but

is there anything to be improved?

Purpose of this part

- Highlight safety related concerns for SONNX profile
- Report some issues related to the legacy ONNX profile
- Make proposals to fix some of them for the SONNX profile.

They are listed unstructured hereafter

Model semantics vs replication criteria (exact vs approximate)

Concern to be addressed as per ARP-6983

- If Operator semantics are mathematical expression in $\mathbb{R}$, it implies approximate replication.
- If Operator semantics is Quantized (float, int…), replication can be exact (bit accurate) or approximate, at the cost of describing each Tensor data_type per Operator semantics

Domain scope

ai.onnx is the base domain of ONNX Operators and Functions.

This domain is unstructured (e.g.: RNN, Max, BlackmanWindow)

Proposal:

We need a methodology to select the Operators for SONNX:

1. industry use cases analysis
2. ai.onnx domain analysis, we can group Operators by …criteria TBD (ex: element wise, math, linalg, quantized…)
3. then select groups which belong to SONNX

SONNX v1 is only for neural networks. Next version should include ML Tree family. In that case, the methodology shall be extended to ai.onnx.ml

Specification of properties on the graph structure & verification means

Proposal:

Example of expected properties:

- No dead node
- DAG structure (no loop)
- Single output Nodes
- Topological Node ordering
- …

Example of verification means:

- Define explicit validity rules for SONNX (similarly to the one implemented in https://onnx.ai/onnx/api/checker.html) ?

Opset resolution, naming ambiguity

Opset resolution problem

- An ONNX Function is a design artefact used to:
  1. define a composition of operators (ex: Relu Function is defined through Max Operator)
  2. define a composition of Nodes in the Graph as a reusable sub-graph (local function)
- Opsets are referenced in the Model element, and in each Function definition.
- Ex : Model import Opset v14, Model local function Relu import Opset v15.

- The Opset resolution is not specified:

  // The (domain, name, overload) tuple must be unique across the function protos in this list.
  // In case of any conflicts the behavior (whether the model local functions are given higher priority,
  // or standard operator sets are given higher priotity or this is treated as error) is defined by
  // the runtimes.

Naming ambiguity problem: when using "Exp" in the semantics, does it refer to the mathematical function or the Exp Operator ?

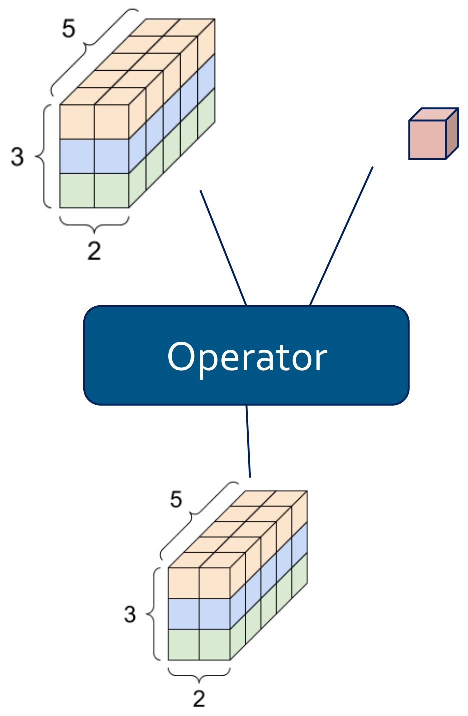
Polymorphism

Problem 1: data types in function parameters

- Function tensor input and output data type and shape are not specified
- To avoid type inference in the implementation, do we need to particularize the semantics for any concrete data type ?

Problem 2: shape broadcasting

- Operator Tensor input shall be of the same element type and shape.
- unless tensor shape broadcasting is enabled.
- Broadcasting logic shall be non ambiguous if supported by safety profile.



19

Overloading , variadic input

Overloading problem:

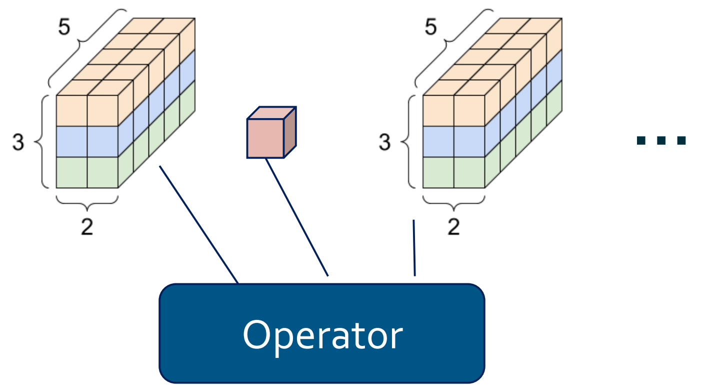
- IR version 10 introduces the overloading capability, i.e. to have several definitions for the same function, and select them using a new ‘overloading’ field.

➔ Proposal: The overloading logic shall be reviewed (non ambiguous ?).

Variadic input problem:

- are used for (Min, Max,…) operators: any number of input can be connected to these operators.

➔ Proposal: We need to define an upper bound



Dynamic (Node input) vs static (Node attribute)

Problem:

The semantics is not clear that Node input is dynamic and Node attribute is static.

As attributes can take Tensor values, these values might come from other Nodes (constant or not)

The ONNX trend follows pytorch : more dynamic capabilities.

E.g.: https://onnx.ai/onnx/operators/onnx__Dropout.html, the ratio attribute in opset 10 was moved to input in recent opsets.

➔ Proposal: Do we follow the trend or do we restrict ? Consequence : compatibility.

Data storage

Proposal:

The data storage might be raw or typed for Tensor and Attributes.

Typed integer is a complex variable length encoding linked to protobuf standard.

Do we specify our own encoding semantics in raw_data field ?

Do we specify another data file format specific to SONNX ?

ONNX failed conversion survey

- See Wenxin Jiang, Arav Tewari, et al, Interoperability in Deep Learning: A User Survey and Failure Analysis of ONNX Model Converters, Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 1466–1478,Vien 2024

What?

- An example of “typical requirements”…
- An example of a partial result…
- Some specific difficulties that we will have to address…

Examples of typical needs and requirements

[REQ-001] For each operator Op, the SONNX standard that specify the necessary and sufficient PRE and POST conditions involving inputs, parameters and outputs such that, *If PRE holds, then POST holds after the execution of any correct implementation of Op.*

[REQ-002] The semantics of each operator shall expressed in a human readable and understandable way

[REQ-003] For a given graph, the SONNX meta-model shall support the description of all necessary data type conversions.

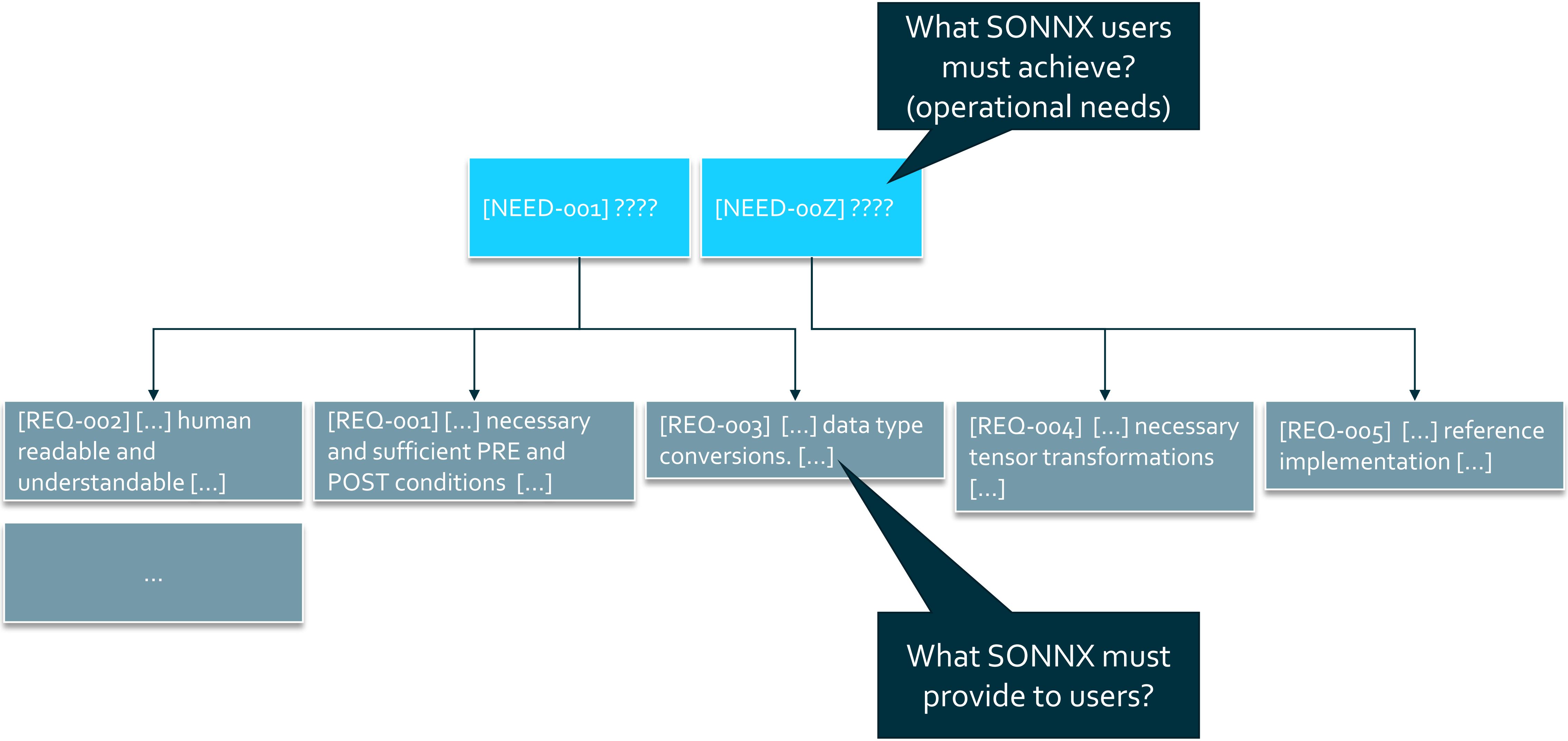
[REQ-004] For a given graph, the SONNX meta-model shall support the description of all necessary tensor transformations.

[REQ-005] The SONNX standard that provide a reference implementation of each construct (operator, graph execution)

[REQ-006] The SONNX standard that provide the capability to specify the exact operator execution ordering

Etc.

Examples of typical needs and requirements



An illustration of a *possible* outcome

- The example of the CONV operator

Signature

`Y = conv(X,W,[B])`

where

- `Y` is the output tensor of the convolution
- `X` is the input tensor to be convoluted with the convolution kernel `W`
- `W` is the convolution kernel
- `B` is the optional bias to be added to the result of the convolution.

Specification for data types: `Y` : real, `X`: real, `W`: real, `B`: real

Informal specification

The `conv` operator computes the convolution of the input tensor `X` with the “filter”, or “kernel”, `W` and adds bias `B` to the result.

A simplified mathematical definition of the operator is given hereafter for the 2D case, without padding and with $group = 1$. The formal specification is given in Section 3.5. When considering padding, the same formula applies, in which `X` is represents the padded version of the actual input `X`.

$$\mathrm{Y}[b,c,m,n] = \sum_{c_i=0}^{W.C_{in}-1} \sum_{k_h=0}^{W.H-1} \sum_{k_w=0}^{W.W-1} (\mathrm{X}[b,c_i,m\cdot \mathrm{strides}[0]+k_h\cdot \mathrm{dilation}[0], n\cdot \mathrm{strides}[1]+k_w\cdot \mathrm{dilations}[1]] \times \mathrm{W}[c,c_i,k_h,k_w]) \; + \mathrm{B}[c_i]$$

See at https://github.com/ericjenn/working-groups/tree/main/safety-related-profile/documents/conv_specification_example

An illustration of a *possible* outcome

- An example of formal specification (using Why3)

There is a bunch of other formal languages (FramaC)...

```
val conv (inp: input_tensor)(kernel: convolution_kernel)(bias: bias_tensor)(attr: attributes)(out: output_tensor): array real
  requires{inp.x_c = out.y_c = kernel.w_c_in = bias.b_c}
  requires{out.y_h = (div (inp.x_h + attr.pads[0] + attr.pads[2] - (attr.dilations[0] * kernel.w_h)) attr.stride[0]) + 1}
  requires{out.y_w = (div (inp.x_w + attr.pads[1] + attr.pads[3] - (attr.dilations[1] * kernel.w_w)) attr.stride[1]) +1}
  requires { inp.x_h > 0 /\ inp.x_w > 0 /\ inp.x_c > 0 /\ inp.x_b > 0}
  requires{kernel.w_h > 0 /\ kernel.w_w > 0 /\ kernel.w_c_in > 0 /\ kernel.w_c_out > 0}
  requires { out.y_b > 0 /\ out.y_c > 0 /\ out.y_h > 0 /\ out.y_w > 0}
  requires { length inp.x = inp.x_h * inp.x_w * inp.x_c * inp.x_b}
  requires{length kernel.w = kernel.w_h * kernel.w_w * kernel.w_c_in * kernel.w_c_out}
  requires{inp.x_h >= kernel.w_h}
  requires{inp.x_w >= kernel.w_w}
  requires{length bias.b = bias.b_c}
  requires{length attr.stride = 2}
  requires{length attr.dilations = 2}
  requires{length attr.pads = 4}
  requires{forall i. 0 <= i < length attr.pads -> attr.pads[i] = 0}
  requires{forall j. 0 <= j < length attr.dilations -> attr.dilations[j]
  requires{forall k. 0 <= k < length attr.stride -> attr.stride[k] >= 1}
  ensures { length result = conv_size out }
  ensures { forall bi ci hi wi ci_in ki_h ki_w: int.
            0 <= bi < out.y_b ->
            0 <= ci < out.y_c ->
            0 <= hi < out.y_h ->
            0 <= wi < out.y_w ->
            0 <= ci_in < kernel.w_c_in ->
            0 <= ki_h < kernel.w_h ->
            0 <= ki_w < kernel.w_w -> conv_result inp kernel bias attr out result bi ci hi wi ci_in ki_h ki_w }
```

[REQ-001] For each operator Op, the SONNX standard that specify the necessary and sufficient PRE and POST conditions involving inputs, parameters and outputs such that, *If PRE holds, then POST holds after the execution of any correct implementation of Op.*

The function...

An illustration of a *possible* outcome

- An example of formal specification (using Why3)

```
predicate conv_result
  (inp: input_tensor)
  (kernel: convolution_kernel)
  (bias: bias_tensor)
  (attr: attributes)
  (out: output_tensor)
  (res: array real)
  (bi ci hi wi: int)
  (ci_in ki_h ki_w: int) =
   let y_idx = bi * (out.y_c * out.y_h * out.y_w) + ci * (out.y_h * out.y_w) + hi * out.y_w + wi in
   let x_h_idx = hi * attr.stride[0] + ki_h * attr.dilations[0] - attr.pads[0] in
   let x_w_idx = wi * attr.stride[1] + ki_w * attr.dilations[1] - attr.pads[1] in

   (0 <= x_h_idx < inp.x_h /\ 0 <= x_w_idx < inp.x_w) ->
      let x_idx = bi * (inp.x_c * inp.x_h * inp.x_w) + ci_in * (inp.x_h * inp.x_w) + x_h_idx * inp.x_w + x_w_idx in
      let w_idx = ci * (kernel.w_c_in * kernel.w_h * kernel.w_w) + ci_in * (kernel.w_h * kernel.w_w) + ki_h * kernel.w_w + ki
      res.elts (y_idx) = bias.b[ci] +. (inp.x[x_idx] *. kernel.w[w_idx])
```

ONNX

Specific questions / difficulties we have to address…

- What level of specification and formalism do we need?
  - Do we need a formal specification ?
    - For what purpose (formal verification? implementation of a formally proved reference implementation?)
    - Based on what formalism?
  - How to provide a flexible specification (covering multiple levels of argumentation)

- How to handle numerical errors?

e. The replication criterion (either exact or approximated) is defined from the ML Constituent requirements and if applicable from the ML Model requirements:

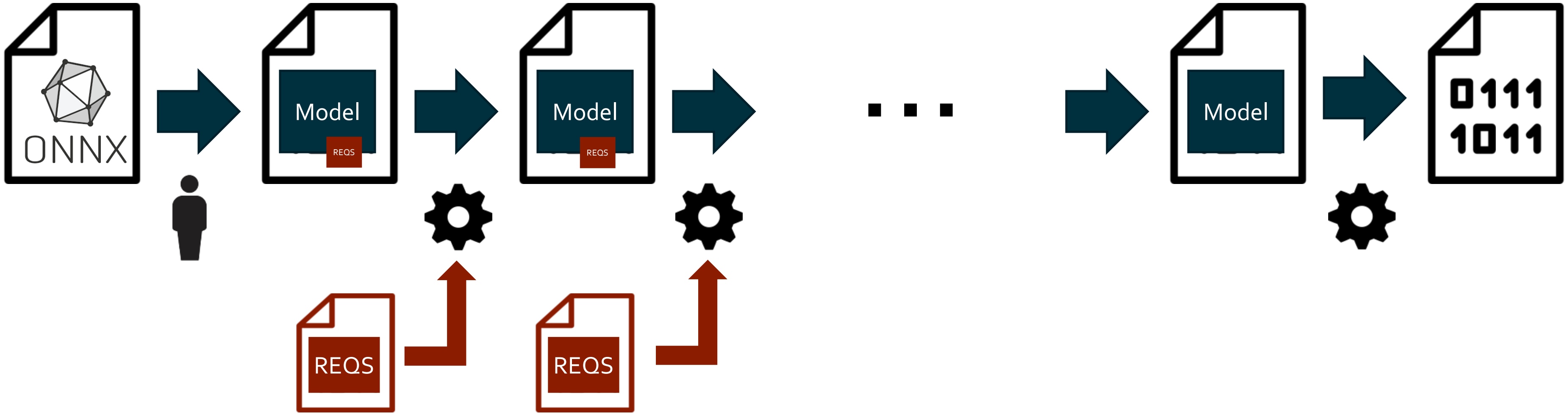
- Exact replication: In this first case, the ML Model description should contain sufficient details on the ML Model semantic to fully preserve this semantic in the implemented ML Model. For example, an exact replication criterion may be the direct and faithful implementation of the ML Model description so that the implemented ML Model meets the same performance, generalization, stability, and robustness requirements.
- Approximated replication: In this second case, the ML Model description should contain sufficient details on the ML Model semantics to approximate this semantic in the implemented ML Model with a specified tolerance. For example, an approximation metric may be expressed for a given dataset by the maximal gap between the trained ML Model outputs and the implemented ML Model outputs. Tthe corresponding approximation replication requirement may be that this maximal gap should not exceed a given value epsilon.

Even the **exact replication** is not formally defined…

Where and how to express and estimate this tolerance?

Specific questions / difficulties we have to address…

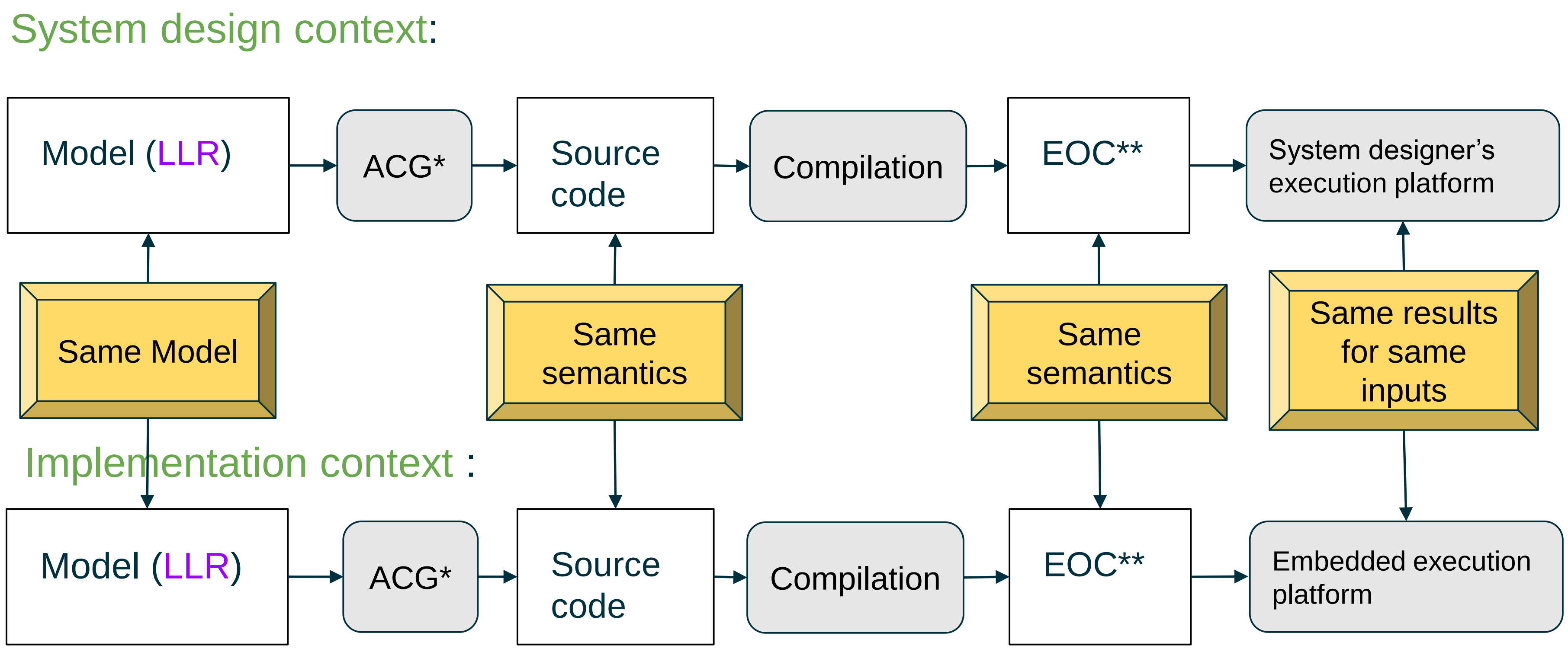
- How to support derived requirements?
  - Do we have to capture derived requirements in the ONNX model or using a dedicated formalism?



An analogy with existing practices

System design context / embedded context Execution representativity

- *Current solution*: The system designer's execution platform is different from the embedded one



* Automatic Code Generator
** Executable Object Code

How?

The activities, the planning

The organization of the working group

The collaboration means

Deliverables

(D1.a) Safety-related Profile **Scope** Definition (2024/11/01)

(D1.b.x) End users **needs** for domain x (2024/12/01)

(D1.c) **Consolidated needs** for all industrial domains (2025/01/01)

(D2.a) ONNX safety-related Profile **requirements** (2025/02/01)

(D3.a) ONNX Safety-related profile - proof of concept (2024/12/01)

(D3.b) ONNX Safety-related profile – graph (2025/05/01)

(D3.c) ONNX Safety-related profile – operators (2025/12/31)

(D3.d) ONNX Safety-related profile – format (2025/12/31)

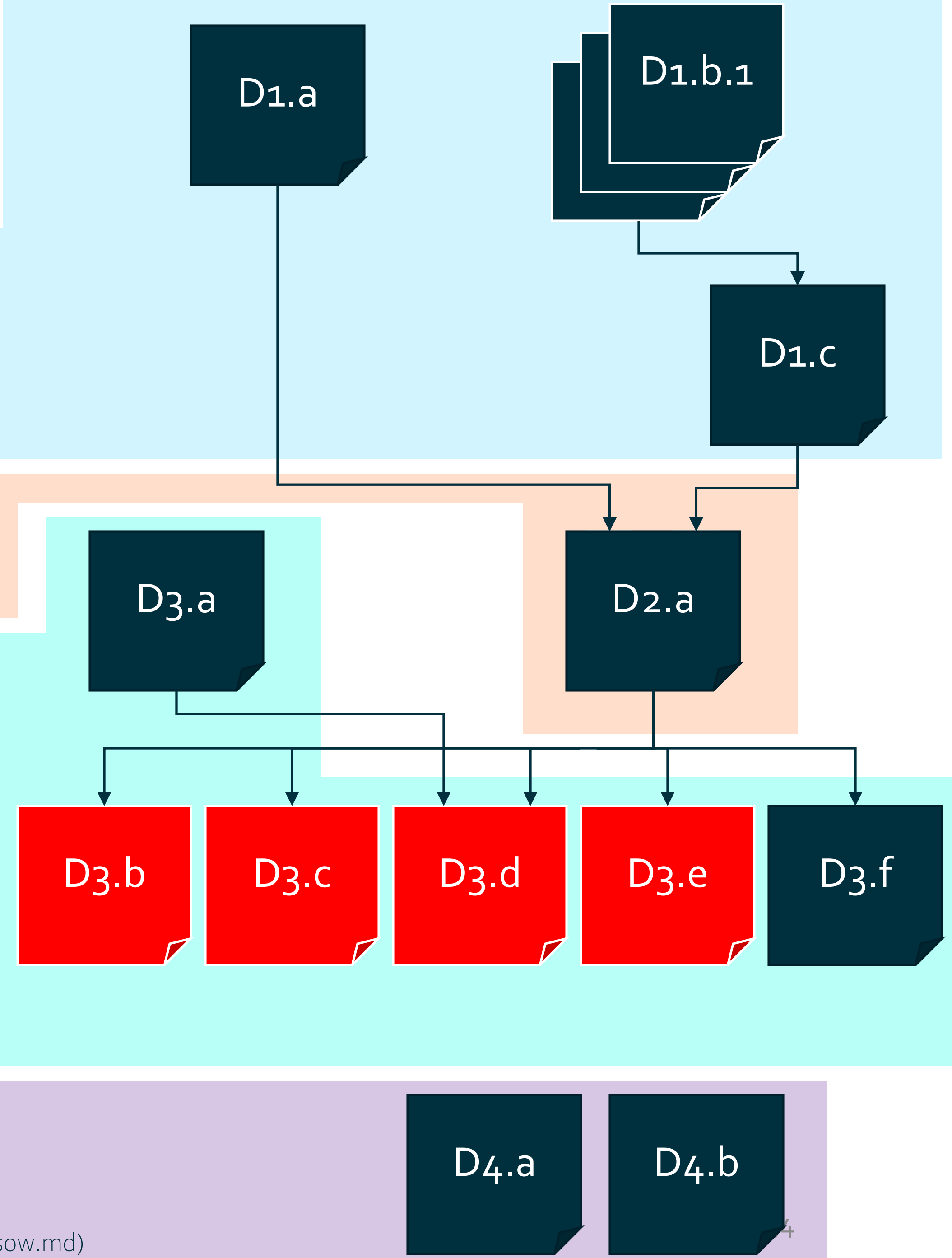
(D3.e) ONNX Safety-related profile reference implementation (2025/12/31)

(D3.f) ONNX Safety-related profile rules (2025/01/31)

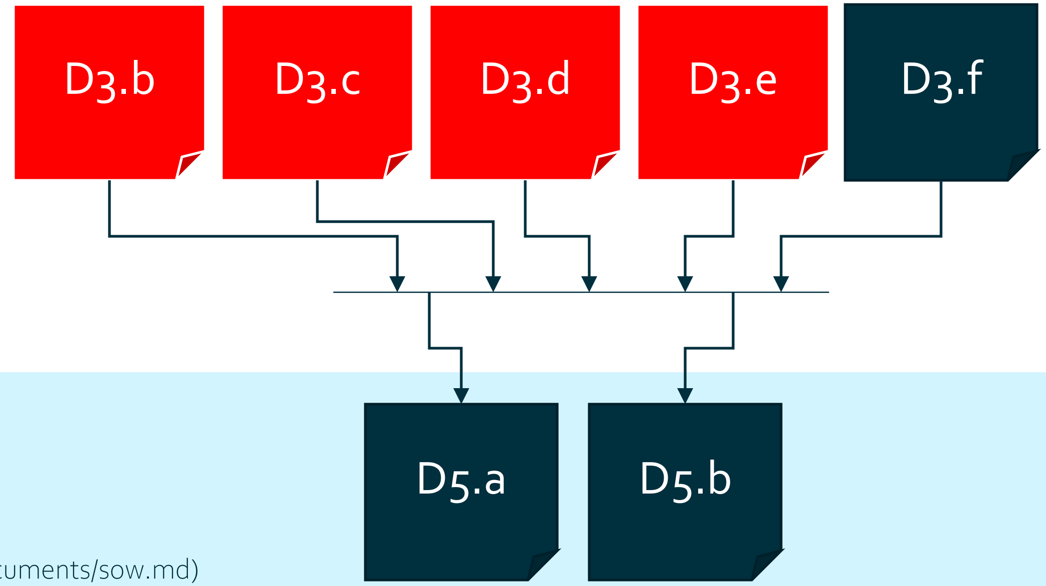
(D4.a) ONNX Safety-related profile **verification** report

(D4.b) ONNX Safety-related profile **validation** report

(detailed WP is available at https://github.com/ericjenn/working-groups/blob/main/safety-related-profile/documents/sow.md)



Deliverables (continued)



(D5.a) Expression of the **needs** / tool list (2025/01/31)

(D5.b) **Requirements** of tool <tool>(2025/12/31)

(detailed WP is available at https://github.com/ericjenn/working-groups/blob/main/safety-related-profile/documents/sow.md)

Activities and planning

T1	Elicitation of industrial needs
Per industrial domain, - Elicit end-users **needs** related to the ONNX format - Elicit **requirements** applicable to the ONNX standard to satisfy the end-users needs. Those requirements shall cover all aspects of the standard, including documentation, graphs and operators semantics, file format, reference implementation, etc.	
Inputs	
	Certification standards (e.g., ARP6983, ISO/DPAS 8800, ECSS-E-HB-40-02A DIR1, etc.)
	Company-specific documents
	End-user use cases
SS	**SR-profile Scope Definition**
SSAct1	Identification/definition of the Safety-related industrial use case reference models
SSAct2	Extraction of the operators and constructs from the Safety-related industrial use cases reference models
SSAct3	Consolidation of the set of operators and constructs to be included in the Safety-related profile (from the reference models possibly augmented with necessary additional operators and constructs).
D1.a	Safety-related Profile Scope Definition
EN	**End-user needs elicitation**
ENAct1.<x>	Description (overview) of the machine learning development process
ENAct2.<x>	Description of the development process objectives and activities
ENAct3.<x>	Definition of the *Trained Model Description (TMD)* artefact (e.g., the Machine Learning Model Description (MLMD) in ARP6983)
ENAct4.<x>	Description of the development process verification objectives and activities that apply to the TMD
ENAct5.<x>	Expression of constraints on the TDM, that come from the Development and verification activities the Industrial context
ENAct6.<x>	Expression of needs
ENAct7	Consolidation of industrial needs from all domains
D1.b.x	End users needs for domain <x>
D1.c	Consolidated needs for all industrial domains

Activities and planning

T2	Specification of the ONNX SR profile
Elaborate the list of requirements applicable to the SR profile in order to cmply with the end users' needs. Consolidate, filter, and prioritize the needs identified for the different industrial domains in D1.a.<x>. Discriminate requirements aimed at the preservation of the model semantics from requirements aimed at facilitating / supporting other development assurance activities.	
Inputs	
D1.b.x	*End users needs for domain <x>*
D1.c	*Consolidated needs for all industrial domains*
OR	**ONNX SR profile requirements specification**
ORAct1	Definition of the list of the aspects (e.g., accuracy, completeness, traceability, etc.) to which the requirements for a safety-related ONNX profile will apply
ORAct2	For each aspect, definition of the requirements applicable to the standard
ORAct3	Grouping and prioritization of requirements
D2.a	ONNX safety-related Profile Requirements

Activities and planning

T3	Development of the ONNX SR profile
Development of the ONNX Safety-related profile (syntax and semantics) in compliance with the ONNX safety-related Profile Requirements (D2.a)	
Inputs	
D2.a	*ONNX safety-related Profile Requirements*
POC	**Proof of concept**
ProAct1	Elaborate a first set of (informal + formal) specification guidelines and apply them on a few operators (e.g., conv) and constructs in order to discussed and reviewed by the workgroup. Will serve as a baseline for other operators
D3.a	ONNX Safety-related profile - proof of concept
Gr	**Graph execution semantics**
GRAct1	Development of the ONNX SR profile graph semantics in compliance with specification D2.a
D3.b	ONNX Safety-related profile - graph
Op	**Operator semantics**
OPAct1	Development of the ONNX SR profile operators semantics in compliance with specification D2.a
D3.c	ONNX Safety-related profile - operators
Fo	**Format**
FOAct1	Development of the ONNX SR profile exchange format in compliance with specification D2.a
D3.d	ONNX Safety-related profile - format
RI	**Reference implementation**
RIAct1	Development of a reference implemation (on the basis of the existing ref imp.). This covers the development of the graph execution engine and operators.
D3.e	ONNX Safety-related profile reference implementation

Activities and planning

T4	V&V of the ONNX Safety-related profile
Verification (validation) of the ONNX Safety-related profile vs the requirements (resp. needs) expressed in D2.a (resp D1.b and D1.c)	
Inputs	
D3.b	ONNX Safety-related profile - graph
D3.c	ONNX Safety-related profile - operators
D3.d	ONNX Safety-related profile - format
VE	**Verification**
VEAct1	Review of the ONNX SR profile against the requirments in D2.a
D4.a	ONNX Safety-related profile verification report
VA	**Validation**
VAAct1	Validation of the ONNX SR profile via its application to one or several industrial use cases
D4.b	ONNX Safety-related profile validation report.

Activities and planning

T5	Tooling
Provision of tooling to support the exploitation of the ONNX SR model. E.g., model inspection and review tool, etc.	
Inputs	
D1.c	Consolidated needs for all industrial domains
D3.a	ONNX Safety-related profile - proof of concept
D3.b	ONNX Safety-related profile - graph
D3.c	ONNX Safety-related profile - operators
D3.d	ONNX Safety-related profile - format
D3.e	ONNX Safety-related profile - reference implementation
D3.f	ONNX Safety-related profile - rules
TN	**Tool needs elicitation**
TNAct1	Expression of the needs for tool
D5.a	Expression of the needs / tool list
TS	**Tool requirements specification**
TSAct2.<tool>	Expression of the requirements of tool <tool>
D5.b	Requirements of tool <tool>

Periodic meetings

- Modalities
  - Every 2 weeks: report of activities, monitoring of progress, distribution of tasks (2 hours)
  - Every 2 months: sub-groups synchronisation and consolidation

- Opportunities (?)
  - Allocate a 30 min slot for presentations (technical topic, use case, etc.)
    Please add propositions of subjects (with possible dates) at https://github.com/ericjenn/working-groups/blob/main/safety-related-profile/meetings/presentations.md

- /!\ Some work has to be done between the periodic meetings /!\

Periodic meeting schedules

- 2024
  - October : 2nd , 16th , 30th
  - November: 12th , 27th
  - December: 11th

- Steering committee?
  - Role? Periodicity? Modality?

Technical means

- Repository
  - ONNX: https://github.com/onnx/working-groups/tree/main/safety-related-profile
  - Forked at https://github.com/ericjenn/working-groups/tree/main/safety-related-profile
- Communication
  - Mail, essentially, due to company policies (no access to Discord for the most of us…)
  - Mailing list to be created (how?)
- Management of tasks
  - Use of github's features : Wiki, discussions, issues,…
  - Policy to be defined

Time for discussion

- Questions? Clarifications?

Roundtable

- What are your expectations (i.e. “why are you here today?”)
- What is the current usage of ONNX in your domain?
- What are your Use Cases for ML and ONNX?

And now?

Identification of contributors

Homeworks

Contributors

- The list of participants is available at https://github.com/ericjenn/working-groups/blob/main/safety-related-profile/documents/people.md
- Add or remove your name, if necessary
- Please indicate if you can
  - Contribute as a writer (writer)
  - Contribute as a reviewer (reviewer)
  - Be simply informed (info)

  If possible, provide any useful information about your subject of interest / contribution

- For the first activity (needs elicitation), we will need one “leader” per industrial domain
- Development activities (tools, reference implementation…) have also to be identified soon
  - Find a way to fund them…

Recall of short term activities (warm-up…)

- Identify target Use Cases
  - What types of models would you like to describe using SONNX
- Define the scope of the profile
  - Collect and consolidate the list of useful operators
- Express your needs / requirements
  - What is the role of the SONNX model in your workflow?
  - What properties to you expect from a SONNX model (req), for what purpose (need)?
- Complete the PoC
  - Provide an example of what we expect as a result
  - *Propose a strategy and formalism…*

Documentation structure to be defined…

- *Other contributions*
  - *Collect the list of "issues" of the current standard that need to be addressed*
  - *Identify tools…*

Contacts

- Eric JENN (eric.jenn@irt-saintexupery.com)
- Jean SOUYRIS (jean.souyris@airbus.com)