

Technical Note

Complements and recommendations of the MBSA modelling guide

DATE: 17/02/2022

Summary

One of the objectives of the S2C project is to develop and validate a comprehensive MBSA methodology adapted to aeronautical developments. This document provides complements and recommendations to the MBSA modelling guide "MBSA Modelling guide and validation report" [1]. It highlights questions, comments and recommendations made by the people in charge of modelling the "AIDA" safety case [2] in the S2C project. Our intention is to provide answers and frequently asked questions during MBSA modelling.

<i>Author(s)</i>	<i>Function(s) & name(s)</i>	<i>Research engineers</i>	<i>A. Legendre</i>
------------------	----------------------------------	---------------------------	--------------------

<i>Checker(s)</i>	<i>Function(s) & name(s)</i>	<i>WP leader</i>	<i>S. Delavault</i>
-------------------	----------------------------------	------------------	---------------------

<i>Approver</i>	<i>Function & name</i>	<i>Project leader IRT Saint Exupéry</i>	<i>J. Perrin</i>
-----------------	----------------------------	---	------------------

Evolutions

Version	Date	Modified §	Modification summary	Modified by
1	17/02/2023		Daft version	A.Legendre
2	13/03/2023		Complete version	A.Legendre

IRT Saint Exupéry

Fondation de Coopération Scientifique

www.irt-saintexupery.com
+33 (0) 5 61 00 67 50
contact@irt-saintexupery.com

SIREN 793 007 048
TVA Intracom : FR29 793 007 048

TOULOUSE (Siège)

Bâtiment B612
3 Rue Tarfaya • CS 34 436
31405 Toulouse Cedex 4
France

BORDEAUX

Arts et Métiers ParisTech
Campus of Bordeaux-Talence
Esplanade des Arts et Métiers,
33405 Talence Cedex
France

SOPHIA ANTIPOLIS

Bâtiment B
240 Rue Evariste Galois
06410 Biot
France

Contents

Summary	1
Evolutions	2
Table of figures	4
Table of tables	4
1. Introduction	5
1.1. Context	5
1.2. Subject of the document.....	5
1.3. Documentary reference	6
2. Modelling Activities: Remarks and recommendations.....	7
2.1. Assertion et Composition.....	7
2.2. CAN Bus Modelling	8
2.3. Modelling of external events.....	9
2.4. Power supply modelling.....	10
2.5. Lambda and Gamma modelling	12
2.6. Type of variables: Boolean or enumerated?	12
2.7. Boolean logic modelling.....	13
2.8. Conditions expressions	14
3. Modelling strategy: Remarks and recommendations	16
3.1. Top down or Bottom up modelling approach.....	16
3.2. Separation of Hardware and Software components	17
3.3. Loop opening position	18
3.4. Failure data and model granularity.....	19
3.5. Block validation and block integration validation	20
4. Conclusion	22
ANNEXE 1 Overview of model specification AIDA for AltaRica 3.0.....	23

Table of figures

Figure 1 Example of assertions from block A.....	8
Figure 2 CAN Bus Modelling with failure mode inside	9
Figure 3 CAN Bus Modelling without failure mode inside	9
Figure 4 Illustration of external event in a block A – Case n°1.....	10
Figure 5 Illustration of external event associated to block A and B – Case n°2.....	10
Figure 6 Illustration of back loop in propulsion unit in AIDA safety case	19

Table of tables

Table 1 Advice to model power supply according several situations.....	11
Table 2 <i>Modelling description of assertion</i>	15

1. Introduction

1.1. Context

The S2C project was born thanks to a industrial need to improve current development processes and to ensure consistency between safety analyses and system design, which today requires a significant effort.

Indeed, the development of systems like aircrafts is made difficult by two major factors: the inherent complexity of such system and the complexity of the organization required to build and maintain them. A large number of stakeholders are involved throughout the lifecycle. They have various concerns that are not necessarily consistent with each other. Each stakeholder has its own viewpoints and descriptions, focused on its concerns and based on the use of dedicated languages, formalisms, and tools. These viewpoints are highly interrelated, which produce overlaps. Clearly, these overlaps and all elements mentioned above may possibly introduce inconsistencies.

In the S2C project, we will focus on the interactions between two particular disciplines: systems engineering and safety analyses. The challenge is to develop methodologies to manage inconsistencies and to maintain a coherent state over the development cycle. These methodologies should be validated through a representative case study supporting by tools demonstrator in order to define tooling specification.

The main objectives are to:

- Improve confidence in safety analysis by aligning and maintaining consistency with system definition models.
- Provide safety specialists with more efficient means to understand and analyze complex systems.
- Facilitate the use of model-based approaches in compliance with the requirements of certification bodies (for the MBSA approach, there are no requirements issued by certification bodies).
- Reduce the number of iterations between system definition and related safety analyses, in order to limit costs and delays, by enabling efficient MBSE/MBSA interfaces or MBSE/FTA interfaces.
- Control changes during the product development cycle; reduce risks due to redesign.

In order to address these issues, the project was divided into 4 work packages:

- WP1: Definition of a global process for ensuring and maintaining system/safety consistency all along the life cycle
- WP2: Methods and means of implementing and maintaining system/safety consistency for the integrated system models and system levels (horizontal links)
- WP3: Methods and means of setting and maintaining consistency of safety data between different system levels (vertical links)
- WP4: Modelling methodology for safety

1.2. Subject of the document

One of the objectives of the WP4 in S2C project is to provide complements and recommendations to the MBSA modelling guide "MBSA Modelling guide and validation report" [1]. It highlights questions, discussions and recommendations made by people in charge of modelling the "AIDA" safety case [2]. Our intention is to provide answers and questions frequently asked during MBSA modelling.

The remarks and practical recommendations are illustrated using the AltaRica language supported by the tools Cecilia Workshop (Dassault Aviation), SimfiaNeo (Airbus Protect) and Open AltaRica.

The (Exupéry, “MBSA Modelling guide and validation report” 2023) report presents many abbreviations, acronyms and definitions around the MBSA approach that can be used in this document.

1.3. Documentary reference

[1] MBSA Modelling guide and validation report, IRT Saint Exupéry LIV-S085L01-001, 2023.

[2] AIDA study case, Architecture description, System version : V4.5, IRT Saint Exupéry, 2021.

[3] Get Started - This starter kit is proposed by the IRT teams, IRT Saint Exupéry LIV-S085L01-001, 2022

[4] Prosvirnova Tatiana, Seguin Christel, Frazza Christophe, and al. Strategies for modelling failure propagation in dynamic systems with AltaRica.” LambdaMu23, IMdR, 2022.

2. Modelling Activities: Remarks and recommendations

Chapter 2 of this report addresses questions and remarks on the modelling activities of certain mechanisms that it may be useful to model in an MBSA model. We propose to present the issue and provide recommendations where possible. The different topics covered are as follows:

- How to reduce the number of assertions when a system contains many levels of decomposition?
- Is it relevant to model CAN BUS in MBSA?
- When should a block be created to introduce external events?
- Do you have any recommendations for modelling the power supply?
- How can we view or edit failure rates in a model without browsing the whole model?
- Is it better to use boolean or enumerated logic variables?
- Is it relevant to model AND / OR logic gates with blocks?
- What are the most efficient logical conditions for the calculation?

2.1. Assertion et Composition

Question :

When in a model, a block A contains other blocks (B, C, D for example), the modeler try to define interfaces from block A to link the interfaces of the sub-blocks in his model. This type of exercise leads to the creation of many assertions. We are wondering here about the interest of linking the higher-level blocks with the lower-level blocks.

Argumentations:

It is not always relevant to do this step because it requires defining many assertions. In particular, two situations can be simplified:

- The input interfaces of block A are one by one connected to the interfaces of the sub-blocks (an input port with an input port)
- The output interfaces of block A are one by one connected to the interfaces of the sub-blocks (an output port with an output port).

In this situation, it is not necessarily relevant to define interfaces at the level of block A. This complicates the model by adding assertions which does not bring added value for MBSA analyses.

However, it can make it easier to read for model validation (example of the Fight Controller block in the AIDA safety case [2]). It also can allows you to define higher level observers calculated by the output values of the sub-components.

On the other hand, it is important to keep the assertions in the following situations:

- The interfaces of block A are logical combinations of several interfaces of the sub-blocks.
- Output interfaces linked to input interfaces between two sub-blocks.

Illustration by example :

Figure 1 describes a block A composed of blocks B, C and D and the assertions that link these blocks together.

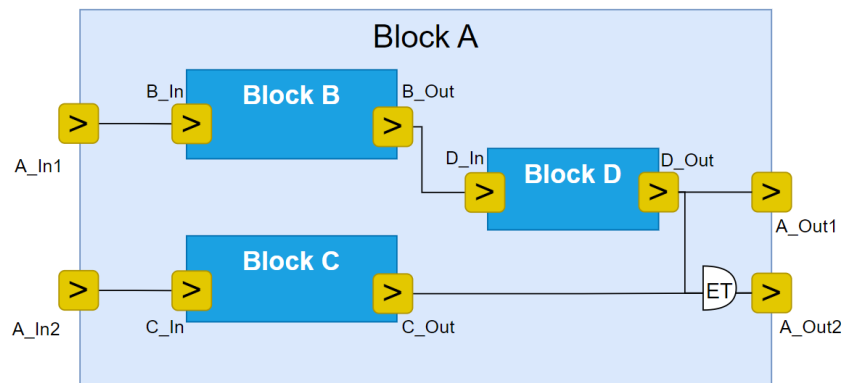


FIGURE 1 EXAMPLE OF ASSERTIONS FROM BLOCK A

The **A_In1**, **A_In2** and **A_Out2** interfaces do not need to be modeled because they are directly connected to the **B_In**, **C_in** and **D_Out** interfaces. When Block A is connected with another block of the same level, we can define an assertion that directly targets the **B_In**, **C_in** and **D_Out** interfaces.

In addition, the **D_In** and **A_Out2** interfaces are interfaces that provide added value:

- **D_In** is necessary because it connects an output interface with an input interface. We must therefore define an assertion.
- **A_Out2** is necessary because it deals with several interfaces (**D_Out** and **C_Out**). We must therefore define an assertion.

Recommendations:

This recommendation makes it possible to limit the number of superfluous assertions in the MBSA model, however it does not facilitate proofreading or validation. It is advisable not to apply this practice for inexperienced modelers.

2.2. CAN Bus Modelling

Question:

When modelling the AIDA safety case model [2] on the different tools (SimfiaNeo, Cecilia Whorkshop and AltaRica Wizard), several modelers wondered about how to represent CAN Bus (for sensors and motors). Shall we represent the level of performance (Quality of service) ? Are there any modelling patterns to represent CAN Bus?

Argumentations:

A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer. The modelling of a CAN consists in grouping to the same CAN Bus several communications (interactions). This technique makes it possible to significantly reduce the number of cables to be connected (in real system) between components. However, from a safety point of view, it generates common failure modes on all interactions included in the CAN Bus.

Several ways of modelling the CAN are possible:

- Case n°1: Modelling of the CAN as a component containing the failure modes associated with each signal.

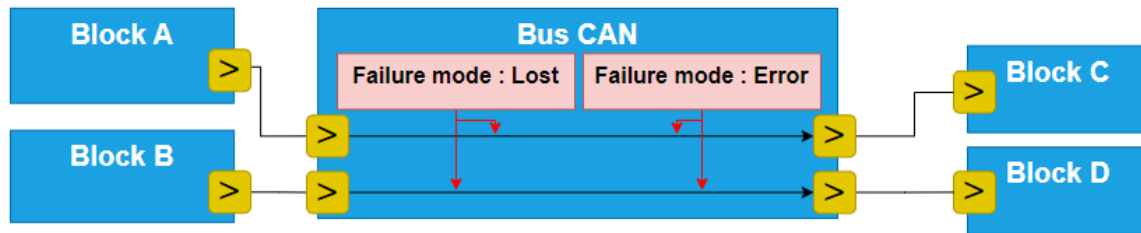


FIGURE 2 CAN BUS MODELLING WITH FAILURE MODE INSIDE

- Case 2: Incorporate CAN failures into signal transmitter and receiver components

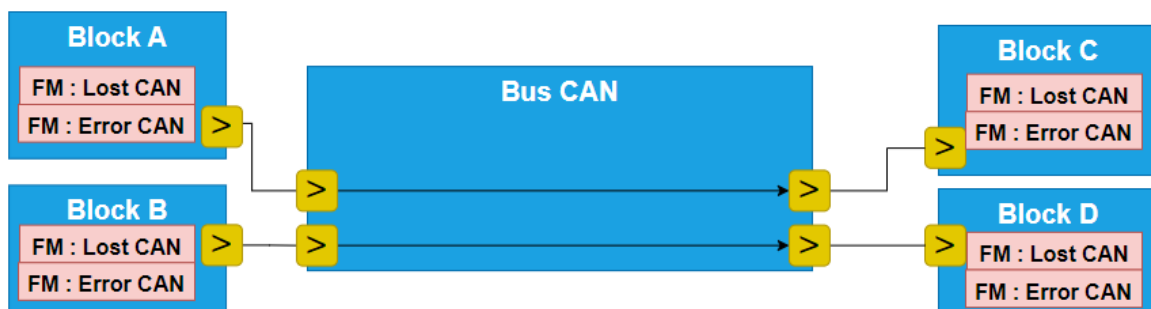


FIGURE 3 CAN BUS MODELLING WITHOUT FAILURE MODE INSIDE

The second pattern have been used to the safety case AIDA [2].

Moreover, it is not required in a safety analysis for aeronautical certification to model the quality of service of the components. Indeed, the requested model must be static and must therefore not consider failure rates or performance changes over time. However, it is possible to make performance analyzes taking into account the quality of service to possibly model latency times but it cannot be used in the context of certification.

Recommendation:

We recommend adopting 1 of the two formalisms presented below to represent the CAN Buses and the failure modes.

2.3. Modelling of external events

Question:

An AIDA safety case modeler wondered when should a block dedicated to outdoor events be modelled? Rather than representing the event directly in the component concerned?

Argumentations:

The modelling of external events can be done in two main ways:

- Case n°1: Modelling the external event inside the component as a new failure mode.
- Case n°2: Modelling the external event in a new dedicated block containing a failure mode. It supposes to add an assertion to propagate the external event to the component that can be impacted.

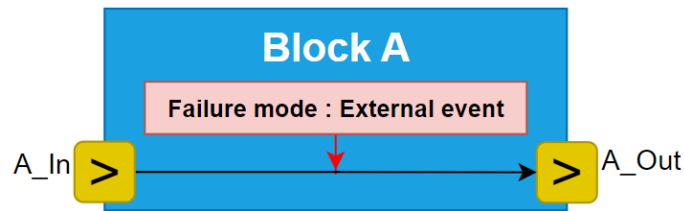


FIGURE 4 ILLUSTRATION OF EXTERNAL EVENT IN A BLOCK A – CASE N°1

Case 1 is simpler in terms of modelling because it simply requires the addition of an event, a transition and the update of the assertion. However, in this situation, the external event impacts only one component, it does not allow to consider a common cause failure.

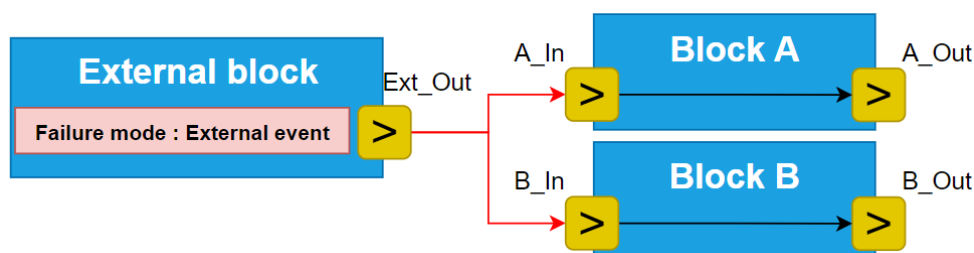


FIGURE 5 ILLUSTRATION OF EXTERNAL EVENT ASSOCIATED TO BLOCK A AND B – CASE N°2

Case 2 therefore makes it possible to represent common causes of failure and allows reuse. It is thus possible to define several external events (by duplicating them or by creating new instances and connecting them to the components targeted by the external event).

Recommendation:

Both proposals for modelling external events are possible. Case 1 is relevant only if the external event impacts only one component (which is rarely the case). We rather recommend the second case to allow you to consider all the impacting components and to facilitate the reuse of the concept of external events.

2.4. Power supply modelling

Question

A modeler wondered about the best way to represent power supplies in models. The question addressed is : Do you have any recommendations for modelling the power supply ?

Argumentations:

To model an electrical power supply, you must first ask yourself a few questions:

- is the power supply part of my system?
- do I consider power loss in the modelling?
- is the power supply common or distributed to several subsystems of the system of interest?

Depending on the answer, the model may be different.

Internal/ External	Failure mode	Energy Distribution or not	Advice
Internal	Yes	Distributed	In this case, the power supply can be represented as a sub-block of the system with several outputs that provide energy. The block must contain failure event and associated transition. In the system block, one assertion shall be added to relate the power supply to each targeted component.
Internal	Yes	Local	In this case, the power supply can be represented as a sub-block of the system with one output that provide energy. The block must contain failure event and associated transition. In the system block, one assertion shall be added to relate the power supply to the targeted component.
Internal	No	Distributed	In this case, the power supply can be represented as a sub-block of the system with several outputs that provide energy. The block doesn't contain any event. In the system block, one assertion shall be added to relate the power supply to each targeted component. Warning, the power supply may have input that influence the energy provided in outputs.
Internal	No	Local	In this case, the power supply don't have any impact on the component because we don't need to consider loss of power supply. It is not necessary to consider it in the model.
External	Yes	Distributed	In this case, the power supply can be represented as a block outside of the system with several outputs that provide energy. The block must contain failure event and associated transition. In the main block, one assertion shall be added to relate the power supply to each targeted component.
External	Yes	Local	In this case, the power supply can be represented as a block outside of the system with one output that provide energy. The block must contain failure event and associated transition. In the main block, one assertion shall be added to relate the power supply to the targeted component.
External	No	Distributed	In this case, the power supply don't have any impact on the component because we don't need to consider loss of power supply. It is not necessary to consider it in the model.
External	No	Local	In this case, the power supply don't have any impact on the component because we don't need to consider lost of power supply. It is not necessary to consider it in the model.

TABLE 1 ADVICE TO MODEL POWER SUPPLY ACCORDING SEVERAL SITUATIONS

In the [1] §4.2 Overview of modelling units, an example of modelling unit is shown. It can be used to model a power supply.

Recommendation:

The power supply can therefore be modeled in different ways depending on the configuration of the system:

- Sometimes, it's not relevant to model Power Supply when it is not in failure and input which conditions the outputs variables.
- If the energy supply only one component, it is not useful to consider it as a separate block, but only as an internal event of the targeted component.
- If the power supply is distributed, it is recommended to create a dedicated block and assertions to link the power supply to each targeted block.

2.5. Lambda and Gamma modelling




Question:

A question was raised on how to effectively check all the failures parameters of component such as lambda or gamma for example. This can be useful for modifying, updating or verifying failure rates in the MBSA model.

Argumentations:

Failure rates in operation or on demand are defined in the models by parameters used by events of the type failure in operation or on demand. Depending on the versions of the AltaRica language and the tools, the method will be different. To my knowledge, there is no API to view and edit all parameters of a model in a threshold input table.

With Cecilia Workshop tools, there is no way to modify all failure parameters of a model in one. To consult and modify the parameters, there are several ways to do it. Solution 1: In the "component" (or "model explorer") tab on the left of the tool, you can access all the components used in the model(s). By selecting the components one by one, you will be able to consult and modify the value of the parameters. Solution 2: In the graphical model (in the main window), it is possible to double click on each component to modify the failure parameters in property view.

With SimfiaNeo, it is possible to modify or check all parameters (failure rate and probability law) of the components in a dedicated table of the model. To do this, you must select the brick of the system of interest in the "model explorer" then select the table of equipment  in the task bar (on top of the tool) and unfold all the system composition ( and ) to see or modify all events and parameters (failure modes, probability laws and lambda).

With AltaRica Wizard, it is possible to modify or check the parameters of the components one by one in the model. However, it is possible to overload a class when creating its instance to update its parameters in the instance. In this case, it is possible to modify only the failure parameters in the block which contains the instance. But it is not possible with the AltaRica Wizard tool to check and modify all the parameters of a model in a single place of the model. It would be possible by developing a script of import and export of model's parameters.

Recommendation:

The tools currently do not offer an effective solution for consulting all the parameters entered in an MBSA model. It is however possible to display and modify them by browsing the components of the model.

2.6. Type of variables: Boolean or enumerated?

Question:

When developing models for various applications such as aeronautics, it is important to choose the appropriate data types for the model variables. This is especially important for safety-critical systems where errors or ambiguities in the model can lead to catastrophic consequences. In the context of modelling languages such as AltaRica, a common question that arises is whether it is better to use Boolean or enumerated logic variables.

Argumentations:

Whether to use Boolean or enumerated logic variables in AltaRica for aeronautics depends on the specific needs and requirements of the system being modeled. Both types of variables can be used effectively in modelling aeronautical systems, but they have different strengths and weaknesses.

Boolean variables are simple and easy to understand, and they can represent the true/false states of a system or a component. They can be useful in modelling systems where the behavior of the component is binary or can be simplified to a binary state. For example, they can be used to model the on/off state of a switch or the open/closed state of a valve.

On the other hand, enumerated logic variables are more complex than Boolean variables and can represent a range of states. They can be used to model systems where the behavior of the component can take on multiple states or where the states are more complex than simple on/off or open/closed. For example, they can be used to model the different modes of operation of a control system or the different levels of severity of a fault.

By convention, most MBSA models in aeronautics use enumerated type state and flow variables with at least three values: "OK", "LOST" and "ERRONEOUS".

In the Get Starded [3], the component contactor is a bit special. Its thruth table is unusual and it embeds both enumerates and boolean variables. Indeed, it receives two inputs **i_cmd** (enumerate) and **i_control** (boolean) and has 3 operating modes (**State_OK**, **State_stuck_Open**, **State_stuck_closed**).

If the contactor is in **State_OK** state:

- If **i_cmd** is equal to OK and **i_control** is false, then the contactor returns OK
- If **i_cmd** is equal to ERR and **i_control** is false, then the contactor returns ERR
- In all other cases, the contactor returns LOST.

If the contactor is in **State_stuck_Open** state:

- In all other cases, the contactor returns LOST.

If the contactor is in **State_stuck_closed** state:

- the contactor returns the value of **i_cmd**.

Recommendation:

In general, it is recommended to use enumerated logic variables in AltaRica for aeronautical systems when the behavior of the component is more complex than a simple binary state. This can help to provide a more detailed and accurate representation of the system and its behavior. However, for simpler systems or components, Boolean variables may be more appropriate and easier to understand. Ultimately, the choice of variable type will depend on the specific needs and requirements of the system being modeled.

2.7. Boolean logic modelling

Question:

A modeler thought of modelling the AND and OR logic gates in blocks in AltaRica to put them to reuse several times in the model (when necessary). The questions addressed are therefore Is it relevant

to model AND / OR logic gates with blocks? Does this step create complexity and increase model construction time and simulation time?

Argumentations:

Modelling AND and OR logic gates in blocks might be a good idea to simplify modelling. In this case, the input and state variables of the component would be connected to coasters of the "AND/OR gate" type (using assertions) and the output of the logic gates would make it possible to determine the output values of the making up.

However, this proposal has many drawbacks, for the following reasons:

- This logic is already implemented in the AltaRica language to express assertions. In addition, nothing prevents you from creating intermediate variables in the component block to avoid having an overly complex Boolean expression for each component output variable.
- The "logic gate" type blocks will only allow you to express the logic of the assertions. They cannot be used in transitions to define the conditions and effects of an event.
- This practice will necessarily create many assertions to link the component variables to the inputs and outputs of the "logic gate" sub-blocks.

Moreover, if you model the logic gates with blocks, this would have some consequences on the execution time of the algorithms, in particular during the following steps:

- during the flattening of the model before launching the calculations.
- during the generation of the fault tree.
- during the stochastic simulation.

Recommandation:

We do not recommend modelling AND and OR logic gates in blocks for the above reasons.

When the assertion logic is complicated to express, we recommend defining intermediate variables in the block and using assertions to determine the values of the intermediate variables and then of the component's output variables.

2.8. Conditions expressions

Question:

A modeler wonders about the different ways to write conditional expressions in AltaRica in an assertion and the impacts of each on the computation time. The question is: What are the most efficient logical conditions for the calculation?

Argumentations:

Conditions used in assertions can be expressed using :

- **direct affectation.** A direct assignment is the most computationally efficient but has limitations. It cannot express a condition to assign or not assign values to a variable.
- **"If – Then – Else" condition.** Note that the "else" branch is mandatory.
- **"Switch – Case" condition.** In AltaRica, the "Switch – Case" operators do not have a "break". The operation stops as soon as the first case whose condition is true occurs.

Direct affectation	"If – Then – Else" condition	"Switch – Case" operators

<pre>output := u and v; // don't consider UNKNOWN value. Only for Boolean variables</pre>	<pre>output := if u == YES and v == YES then YES else if u == NO and v == NO then NO else UNKNOWN;</pre>	<pre>output := switch { case u == YES and v == YES: YES case u == NO and v == NO: NO default: UNKNOWN };</pre>
--	--	--

TABLE 2 MODELLING DESCRIPTION OF ASSERTION

When the first expression is expressed (direct assignment), the two others conditions will affect the performance in the calculation phase in the same way. The switch expression is equivalent to the “if-then-else” cascade in term of calculation.

The Switch expression has the advantage of being easy to re-read and seems to structure better to check for different conditions to affect a particular variable, unlike cascade “if-then-else” condition.

Recommendation:

We don't have a strong recommendation. The three types of expressions are rather dependent on the conditions you wish to express. If the condition can be expressed only with a Boolean expression, we recommend using a direct assignment, if the condition must express 2 possible cases, we recommend using the "If – Then – Else" expression. Finally, if the condition must express more than 2 possible cases, we recommend "Switch – Case", although it is possible to also use the expression "If – Then – Else".

The use of one expression, rather than another, will have no particular impact on the calculation time of the model.

3. Modelling strategy: Remarks and recommendations

The chapter 3 addresses questions and remarks on method or strategy that may be useful to build an MBSA model. We propose to present the issue and provide recommendations where possible. The different topics covered are as follows:

- What approach to take to build an AltaRica model of a system when the model specification is already prepared?
- Which view is most effective for representing a system? Should we separate the hardware, software, electrical, functional, etc. layer?
- In the performance business, the choice of the position of the loop opening is important. For MBSA models, do you have any recommendations on the position of the buckle openings?
- Should failure parameters be defined on equipment/block a priori in the MBSA model in order to determine the granularity of the model for the simulation?
- How to ensure that locally the block behaves as expected (this implies having defined a specification beforehand)?

3.1. Top down or Bottom up modelling approach

Question:

A modeler questioned which method to adopt to build an AltaRica model of the AIDA safety case [2] when the model specification is already prepared. That is, the components, flows and breakdowns are already defined, all that remains is to build the model.

Method:

There are great approaches to modelling systems:

- **The top-down**, approach consists of starting from the entire perimeter of a system, and breaking down the flow propagation logic into as many detailed elements as there are levels in the system. We thus start from the macro to go towards the micro.
- The **bottom-up**, or ascending, approach is the reverse of the top-down approach. It consists of starting from the 'bottom', i.e. the most detailed level of the echelons, and going up to the highest instances of the system of interest.

However, there is no one way to code an MBSA model of a system. However, we address methodological recommendations to help you make a model of a system. This approach has been used to develop the MBSA model in AltaRica 3.0 language of the safety case AIDA [2].

For modelling in AltaRica, we recommend adopting a hybrid Top-Down and Bottom-Up approach.

- Start with a Top-Down approach
 - Create a system block and its environment. It is simply a matter of creating a block and defining these sub-blocks, adding flow variables and adding assertions to link the blocks together. The system block and its environment is the main block, so it is the perimeter of this block that will then be used in simulation.
 - Then detailed each of the blocks with sub-blocks. Similarly, add flow variables and add assertions to link blocks together.

- Continue with a Bottom-Up approach
 - Create behavioral classes. In general, in these classes, we define the state variables, the failure events and the associated transitions.
 - Create generic behavioral classes that will be used for generic components (which will inherit behavioral classes) to have all the combinations of failure modes that can occur on each component of the system. For example, some components may have a LOSS failure, other components may have LOSS and ERRONEOUS failure modes, etc.
 - Create classes of elementary components (which can be found in several places in the model). In these classes, we inherit from the generic behavioral classes and we add the flow variables as well as the assertions allowing to determine the values of these variables.
- Finish with a hybrid approach
 - Identify the second step of the top-down approach, i.e. build the decomposition of the blocks with sub-blocks (without forgetting to create the flow variables and the assertions). Stop this step when your component contains elementary components.
 - The next step consists in using the classes of the elementary components in the blocks of the higher level. To do this, they just need to create instances of the classes and create the appropriate assertions to link the inputs and outputs of the block to the input and output of the instantiated classes.
 - Add observers in the models on the variables or combination of variables which will be necessary to quantify during the simulation, in general these are feared events.

Throughout the process, we recommend that you do syntax checks and step-by-step simulations to test your model and your classes.

3.2. Separation of Hardware and Software components

Question:

When developing a system, it is important to have a clear and accurate representation of the system and its components in order to ensure that it functions as intended. One question that arises is which view is the most effective for representing a system. There are several views that can be used, such as the functional view, structural view, and behavioral view, each of which emphasizes a different aspect of the system.

Another question that arises is whether we should separate the different layers of a system, such as hardware, software, mechanical, electrical, and functional layers. Separating these layers can provide a more modular and structured design, allowing for easier maintenance and updates. However, it can also increase complexity and make it more difficult to integrate the different layers.

Argumentation:

In MBSA (Model-Based Safety Assessment), the most effective view for representing a system is typically the functional view. This view describes the system's behavior and requirements in terms of inputs, outputs, and state transitions. The functional view is typically used to define the system's safety requirements and identify potential hazards, which can then be further analyzed and mitigated using other views, such as the hardware, software, and physical views.

In MBSA, it is often useful to separate the different layers of the system, such as hardware, software, mechanical, and electrical, in order to facilitate analysis and communication. This allows different teams

and stakeholders to focus on specific aspects of the system and identify potential issues or risks. But, if this separation of layers have not been set in the architecture design, it would be difficult and may add complexity in the MBSA model if you try to add more segregation.

For example, in MBSA, the hardware view may describe the physical components of the system, such as sensors and actuators, and their properties and interfaces. The software view may describe the system's software architecture and algorithms, and how they interact with the hardware components. The mechanical and electrical views may describe the physical connections and interactions between the system's components.

By separating the different layers of the system in this way, MBSA can help identify potential safety issues and ensure that the system is designed and operated in a safe and reliable manner. However, the appropriate separation of layers will depend on the specific context and goals of the MBSA analysis, and may vary from one project to another.

Recommendation:

We recommend following the different layers of the system as much as possible as defined by the design. This will facilitate communication with the design and maintain consistency of the MBSE and MBSA models.

3.3. Loop opening position

Question:

In the performance business, the choice of the position of the loop opening is important. For MBSA models, do you have any recommendations on the position of the buckle openings?

Argumentation:

In the performance analysis, the choice of the position of the opening of the loop is important (because it plays on the transmission times of the information and therefore on the final servo-control).

Because the AltaRica language makes it possible to model and simulate discrete events from a guarded transition system, it is not subject to continuous time. Loops can be opened between any pair of components that are part of the loop. However, a judicious positioning will allow a fairer (more representative) flow propagation. The S2C team wrote an article [4] at the LambdaMu23 congress in October 2022 on the subject. It presents several examples of loop opening and offers recommendations.

The AIDA safety case presented in [2] has many loops, including 3 main ones:

- Between the sensors and the flight controller;
- Between the flight controller and the motors;
- Between the monitoring part and control (found in several components).

To illustrate our purpose, we choose to show an example from the ADIA safety case from [2]. It is part of a propulsion unit composed by a motor control and protection block, a motor block and a propeller block. We can notice in the figure a feedback loop between the motor, the motor controller and the motor protection. To model the safety case AIDA [2] in AltaRica 3.0 and opening the loop, we chose to not represent the dotted assertion in the motor block.

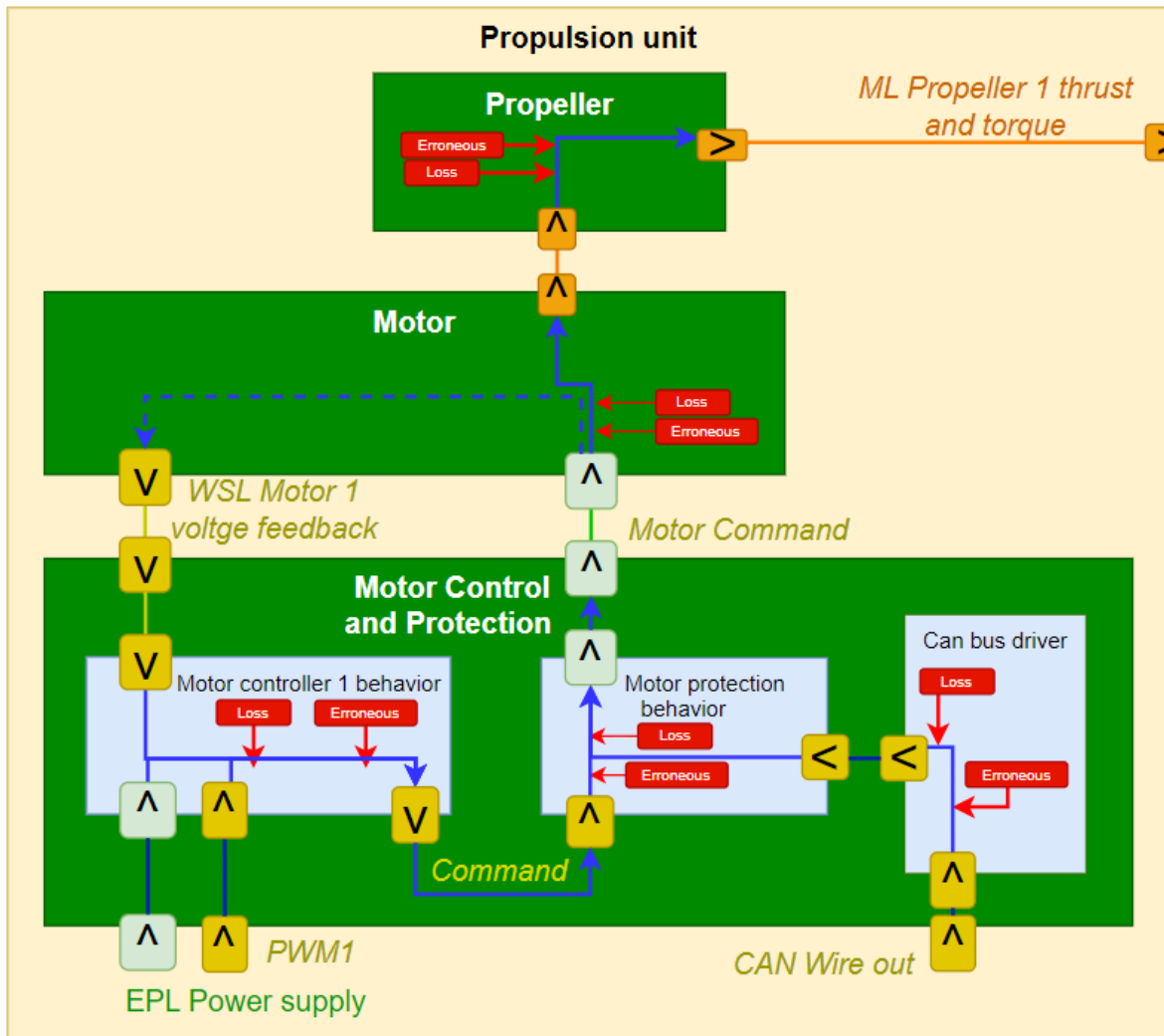


FIGURE 6 ILLUSTRATION OF BACK LOOP IN PROPULSION UNIT IN AIDA SAFETY CASE

For the modelling of the AIDA safety case with AltaRica 3.0, we made the choice to open the loops as far upstream as possible from the system effect (we have cut a propagation link). For more detail and explanation, read the article [4].

3.4. Failure data and model granularity

Question:

When creating an MBSA model, one question that arises is whether failure parameters should be defined on equipment/blocks a priori in order to determine the granularity of the model for the simulation.

Argumentation:

Failure parameters are used to describe the failure behavior of a system or component, such as the probability of a failure occurring, the severity of the failure, and the time to repair the failure. By defining failure parameters on equipment/blocks, the model can be made more granular and provide a more

detailed representation of the system. However, this can also increase the complexity of the model and make it more difficult to analyze.

The decision to define failure parameters a priori on equipment/blocks depends on several factors such as the complexity of the system, the desired level of detail, and the goals of the simulation. In some cases, it may be more appropriate to define failure parameters at a higher level in the model and use a more coarse-grained approach. In other cases, it may be necessary to define failure parameters at a more detailed level in order to accurately capture the behavior of the system.

In general, the specification of an MBSA model contains:

- A structural part including the different components and how they are connected. It describes how the system is broken down and how the components are associated with each other by a flow propagation mechanism.
- A behavioral part. It describes the basic behaviors of the components (their failures, failure laws and parameters and changes of state), as well as possible synchronization of events in the intermediate components/equipment.
- In addition, the specification should outline the safety requirements for the system and the performance criteria that must be met.

When composing the specification of an MBSA model, it is important to work closely with stakeholders, such as designers, operators, and safety experts, to ensure that the model accurately reflects the system and its safety requirements. This may involve conducting interviews, reviewing documentation, and analyzing data to gain a comprehensive understanding of the system and its behavior.

Overall, composing a specification of an MBSA model is an important step in ensuring the safety and effectiveness of a system. By creating a clear and detailed description of the system and its requirements, stakeholders can work together to develop an accurate and effective model that meets the needs of the project.

It is by defining this specification that you will define the level of granularity of your model as well as all the failure parameters to be taken into account (Lambda or Gamma).

For example, ANNEXE 1 Overview of model specification AIDA for AltaRica 3.0 show structural part of a graphical specification of AIDA Safety case [2] for the AltaRica 3.0 model.

Ultimately, the decision of whether to define failure parameters a priori on equipment/blocks in the MBSA model will depend on the specific needs and goals of the project, and may require careful analysis and evaluation to determine the best approach.

Before moving on to modelling the system, i.e. writing the model, it is important to define the structure and the expected behavior of the model, this is the specification of the MBSA model.

3.5. Block validation and block integration validation

Question:

How to ensure that locally the block behaves as expected (this implies having defined a specification beforehand)?

Argumentation:

In model-based safety assessment (MBSA), the validation of block behaviour and the integration of blocks is critical to ensure that the overall system behaves as expected.

Regarding the validation of the behaviour of the blocks, one way to ensure that a block behaves as expected locally is by performing block-level testing. Block-level testing involves testing each individual block in isolation to ensure that it performs as expected.

The following steps can be followed to perform block-level testing in MBSA:

1. Identify the flow variables (input and output) of the block that need to be tested.
2. Create test cases that cover a range of inputs, including normal and abnormal scenarios (Lost signal or erroneous signal for example).
3. Set up the simulation environment and configure the inputs to the block.
4. Run the stepwise simulation and compare the output of the block to the expected output based on the test cases.
5. Analyze the results and identify any deviations between the actual and expected output.
6. Debug any issues and refine the test cases as necessary.

Block-level testing is a crucial step in MBSA as it ensures that each block behaves as expected before it is integrated into the larger system. It is important to thoroughly test each block or class to prevent issues from propagating to the overall system, which can result in safety hazards or other negative consequences.

The Get Started [3] is a good illustration of what to do on simple case to test and validate component behaviour on MBSA model.

Once the components are individually validated, it's become possible de begin the validation of the integration of the blocks at the system level. The following steps can be taken to ensure that the block integration is done correctly:

1. Identify the interfaces between the blocks that need to be integrated by assertion. This includes the flow variables (input and output signals) and states variables and their respective units, data types.
2. Define a set of integration tests that cover all possible scenarios, including normal and abnormal cases. These tests should validate the interactions between the blocks and ensure that they work together as expected.
3. Set up the simulation environment and configure the inputs to the integrated system. This includes providing the inputs to the individual blocks and configuring the outputs of one block as the inputs to another block.
4. Run the stepwise simulation and compare the actual output of the integrated system to the expected output based on the integration test cases.
5. Analyze the results and identify any discrepancies between the actual and expected output.
6. Debug any issues and refine the integration test cases as necessary.

It is important to thoroughly test the integration of blocks in MBSA as it helps to identify potential safety hazards or other issues that may arise when the blocks are combined. Integration testing helps to ensure that the overall system behaves as expected and meets the safety requirements.

IRT Saint Exupéry

Fondation de Coopération Scientifique

www.irt-saintexupery.com
+33 (0) 5 61 00 67 50
contact@irt-saintexupery.com

SIREN 793 007 048
TVA Intracom : FR29 793 007 048

TOULOUSE (Siège)

Bâtiment B612
3 Rue Tarfaya • CS 34 436
31405 Toulouse Cedex 4
France

BORDEAUX

Arts et Métiers ParisTech
Campus of Bordeaux-Talence
Esplanade des Arts et Métiers,
33405 Talence Cedex
France

SOPHIA ANTIPOLIS

Bâtiment B
240 Rue Evariste Galois
06410 Biot
France

4. Conclusion

The MBSA modelling guide and validation report provides a comprehensive framework for modelling and simulation in the context of aeronautics safety assessments. It presents a detailed description of the MBSA approach, modelling process and its components, including data modelling, architecture model, and output analysis.

The AIDA use case modeling highlights that there are topics (such as ...) that have been illustrated in the guide that could be discussed. This is why some questions and recommendations have been provided in this report coming from people that made MBSA model of the [2] in the tools: SimfiaNeo, Cecilia workshop and AltaRica Wizard. The questions and recommendations addressed was focus on modelling concerns and modelling methodology.

This return of experience demonstrated the pertinence of the guide for MBSA modeling in aeronautical context.

ANNEXE 1 Overview of model specification AIDA for AltaRica 3.0

