Typology of the differences Between Model-Based System Engineering (MBSE) and Safety Assessment (MBSA) models: Analysis of a Reference System

Julien Vidalie, Michel Batteux, Jean-Yves Choley, Faïda Mhenni, Mohamed-Sami Kendel

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Typology of the differences Between Model-Based System Engineering (MBSE) and Safety Assessment (MBSA) models: Analysis of a Reference System

Typologie des différences entre modèles d'Ingénièrie Système Basée sur les Modèles (MBSE) et de Sûreté de fonctionnement (MBSA) : Analyse d'un système de Référence

Julien VIDALIE
Quartz Laboratoire
IRT SystemX
Palaiseau, France
julien.vidalie@irt-systemx.fr

Michel BATTEUX
IRT SystemX
Palaiseau, France

Jean-Yves CHOLEY
Quartz Laboratoire
Supméca
Saint-Ouen, France

Faïda MHENNI
Quartz Laboratoire
Supméca
Saint-Ouen, France

Mohamed-Sami KENDEL
Quartz Laboratoire
Supméca
Saint-Ouen, France

***Abstract*—With the increasing complexity of systems, engineers have to design an increasing number of models to perform simulation of the product. In this work we intend to compare the system engineering and safety models of a system, and establish a typology of the differences between those models.**

***Résumé*—Avec l'explosion de la complexité des systèmes, les ingénieurs doivent concevoir un grand nombre de modèles afin de les représenter et simuler. Dans cet article, nous comparons les modèles d'ingénierie système et de sûreté de fonctionnement d'un système et établissons une typologie des différences entre ces modèles.**

Keywords—MBSE, MBSA, Model consistency, differences typology, AltaRica 3.0, SysML

I. INTRODUCTION

Developing complex systems is a multidisciplinary process that requires many different models to be used to represent and simulate one same system for different purposes. As those different models are made by different actors, it is obvious that there is a high risk that they present some inconsistencies. In this context there is a need for verification of consistency of the MBSA model with the MBSE model it is derived from. System Engineering and Safety Assessment are two disciplines that are deeply correlated. The system engineer and safety analyst need to work together to prove that the system is safe, especially in industrial fields that require safety certification.

This paper aims at providing a typology of differences that can occur between a Model Based System Engineering (MBSE) model and a Model Based Safety Assessment (MBSA) model. It is a basis for further work centered on MBSE/MBSA synchronization. This typology does sort differences between models based on their causes and allows engineers to get a better understanding over what should or shouldn't be corrected in the models. This is the key for a future formal definition of what is an inconsistency.

To establish this typology we modeled a reference system, a landing gear study case [5], from both the MBSE and MBSA point of view, and analyzed the differences that occurred between those models.

The remainder of the paper is organized as follows. Section II describes the landing gear study case that we used for this study. Section III presents the modeling of the system in both MBSE and MBSA points of view. Section IV presents some previous work about model synchronization, comparison of the models and foundations of our typology of differences. Finally the conclusion is given in the last section, along with some perspectives for future work.

II. THE LANDING GEAR STUDY CASE

A. Study case

The landing gear study case was described in [5] and served as a benchmark for techniques and tools for the assertion of system behavior. This system is a standard aircraft landing gear composed of three gears (front, rear-left and rear-right). It describes the system pilot interface, its mechanical and hydraulic parts, and its digital control part.

This system is also relevant to a MBSA study, since aeronautic systems are required to be compliant with CS25 regulations, an aeronautic recommended practice describing

safety analyses that are authorized to be completed on aircraft equipment for certification is the ARP4761a [7].

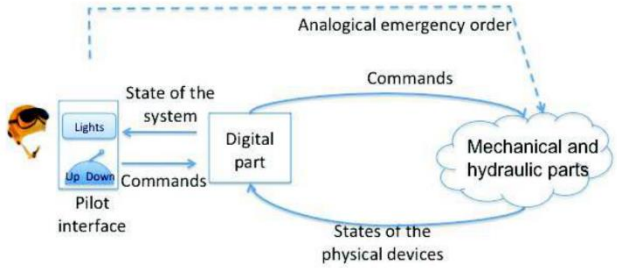

Fig. 1 : Global Architecture of the system

This study case will also be used in further work for the definition of a mathematical framework around the S2ML language [1], which will be referred to in the IVth section of this paper, and for MBSE/MBSA synchronization.

For this work, two models of this system were created, the first one aiming at modeling the system architecture, made with the Cameo System Modeler tool with the SysML language [6], and the second one is a safety analysis view that was created using the OpenAltaRica tool with the AltaRica 3.0 [2] modeling language. To comply with the aeronautic certification requirement, these models have to be separated and to be made by separated people. This allows safety analysis to independently verify the compliance of the architecture described in the MBSE model. In this work we reproduce a realistic workflow, with models written by two different people and did not eliminate differences before the final review. By this protocol we want our workflow to present realistic differences, and we want not to avoid inconsistencies by not having independency between both models. The creation of the MBSA model was based on the MBSE model, and we aim at detecting differences that occurred in this creation.

B. Introduction to the Landing Gear System

As depicted in Fig. 1, the system is composed of three main parts :

1) The Pilot Interface

This part allows the system to communicate with the pilot. It is composed of a Handle which is used by the pilot to order the system to be up or down. This handle communicates its position to the digital part.

Three lights indicate to the pilot the status of all three gears using the following code:

- Green light: “Gear locked down”
- Orange light: “Gear maneuvering”
- Red light: “Landing gear system failure”
- No light: “Gear locked up”

2) The Mechanical and Hydraulical Parts

The structure of the Hydraulic part is described in Fig. 2. The system is composed of three landing sets (front, rear-left and rear-right). Each set has a box (containing all the components, to be fitted in the aircraft landing gear well), door which is opened and closed by a cylinder and a landing gear that is extended and retracted by another cylinder.

The hydraulic power is provided to the cylinders from the aircraft hydraulic circuit by a set of electro-valves:

- One general electro-valve supplies all the system from the aircraft hydraulic circuit
- One electro-valve provides pressure to the portion of the system related to door opening
- One electro-valve provides pressure to the portion of the system related to door closing
- One electro-valve provides pressure to the portion of the system related to gear extending
- One electro-valve provides pressure to the portion of the system related to gear retracting

Each valve is controlled by electrical order from the digital part.

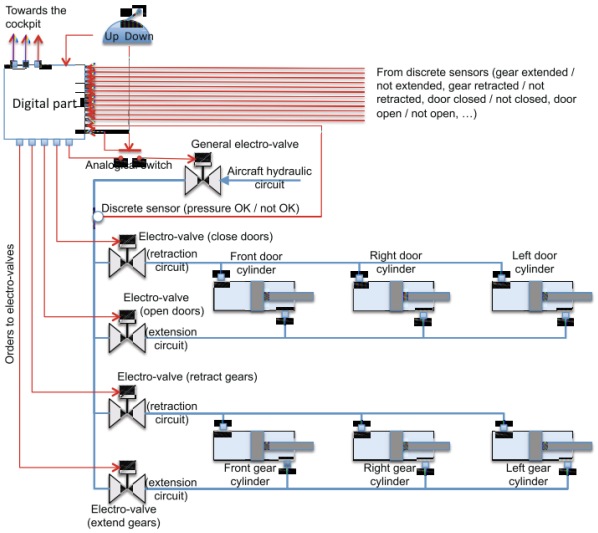

Fig. 2. : Architecture of the hydraulic part [5]

3) The Digital Part

The digital part is composed of two redondant computing modules that run the same control software. It is used to receive and analyze data from the system sensors, and to command the system’s components.

III. System Modeling

A. MBSE modeling

1) Methodology

In our study, the MBSE modeling was realized following the SysML methodology described in [8]. This methodology first focuses on a black box analysis of the system describing requirements, system context, lifecycle and operational scenarios. Then some white box views of the system represent its functional and physical structure in addition to its behavior.

For synchronization of the MBSA model we only focus on the white box views of the system. In fact, our interest is about the structural and behavioral features of our models.

2) Modeling

The system architecture is modeled around its 3 main subsystems which can be observed in the Block Definition Diagram (BDD) shown in Fig. 3. This BDD shows the system’s breakdown structure. Arrows in the diagram represents composition links, meaning that one block (or

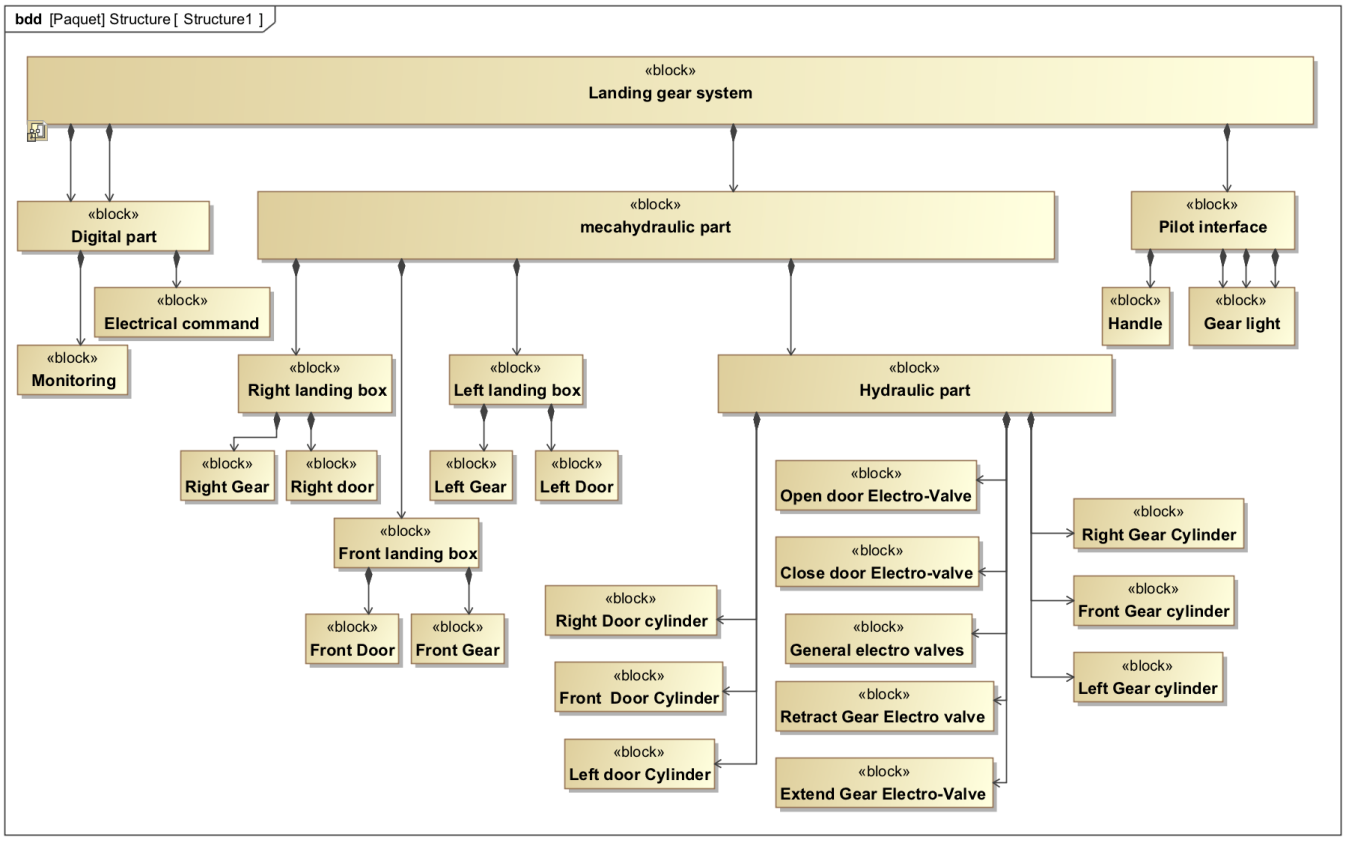

Fig. 3 : Product Breakdown Structure of the Landing Gear System (BDD)

system) is composed of the components that arrows are pointing to. This view of the system shows us its structure but does not define its architecture for we lack connections and flows between the components, which are specified in the Internal Block Diagram (IBD) in Fig. 5. In this IBD, we can observe the relations between components. The names of the connections between them give us information over their type of flows. Finally they can also contain typed variables. It is important to note that this model does not intend to simulate our system but rather to give us a communication tool over it, and means for traceability. Therefore naming on this model is of high importance and carries a lot of information compared to usual simulation models where the content of variables, flows, and other quantitative values are the most important information carried by the model.

The two views we previously described are the ones that will be interesting in the context of MBSE/MBSA synchronization for the scope of this work. In further work we will also investigate the state machines diagrams and sequence diagrams that describe the functional behavior of the system.

B. *MBSA Modeling*

1) Methodology

The MBSA modeling of a system can be part of its safety analysis and is a method that is accepted by authorities as a safety analysis method with its integration in the protocol described in ARP 4761a [7] regulations.

Although most safety analysis nowadays is carried through analysis such as Fault Tree Analysis (FTA), MBSA modeling is a good tool for safety analysis thanks to its high expressivity. It is easier to understand and to communicate with this model, making it easier for the safety analyst to shows problems in the architecture to the system engineer.

The MBSA model that we made for this work was created using the OpenAltaRica platform, based on the AltaRica 3.0 modeling language. It represents the system through its structure and dysfunctional behavior. Unlike the MBSE tool SysML, which is a graphical notation, AltaRica is a formal language, meaning a textual syntax and a semantic, even though some AltaRica tools such as Simfia provide graphical interface to design parts of the AltaRica model.

2) Modeling

The MBSA modeling was achieved using the article [5] presenting the system as a reference document and based on the MBSE modeling it aims at verifying. We here considered the MBSE model as a specification document of the system and expect to verify its compliance to safety requirements through MBSA modeling. Except for a few differences which will be talked over in section IV, the model has a very similar structure to the one presented in the SysML IBD and BDD in Fig. 3 and Fig. 5 respectively.

```
domain nrpState {OK, KO}
class NonRepairableComponent
  nrpState s (init = OK);
  parameter Real lambda = 1.0e-5;
  event failure (delay = exponential (lambda));
  transition
    failure: s == OK -> s := KO;
end
```

Fig. 4 : Class NonRepairableComponent in AltaRica 3.0

In AltaRica 3.0, we represent the system by a main “block” which is a container that will be considered and

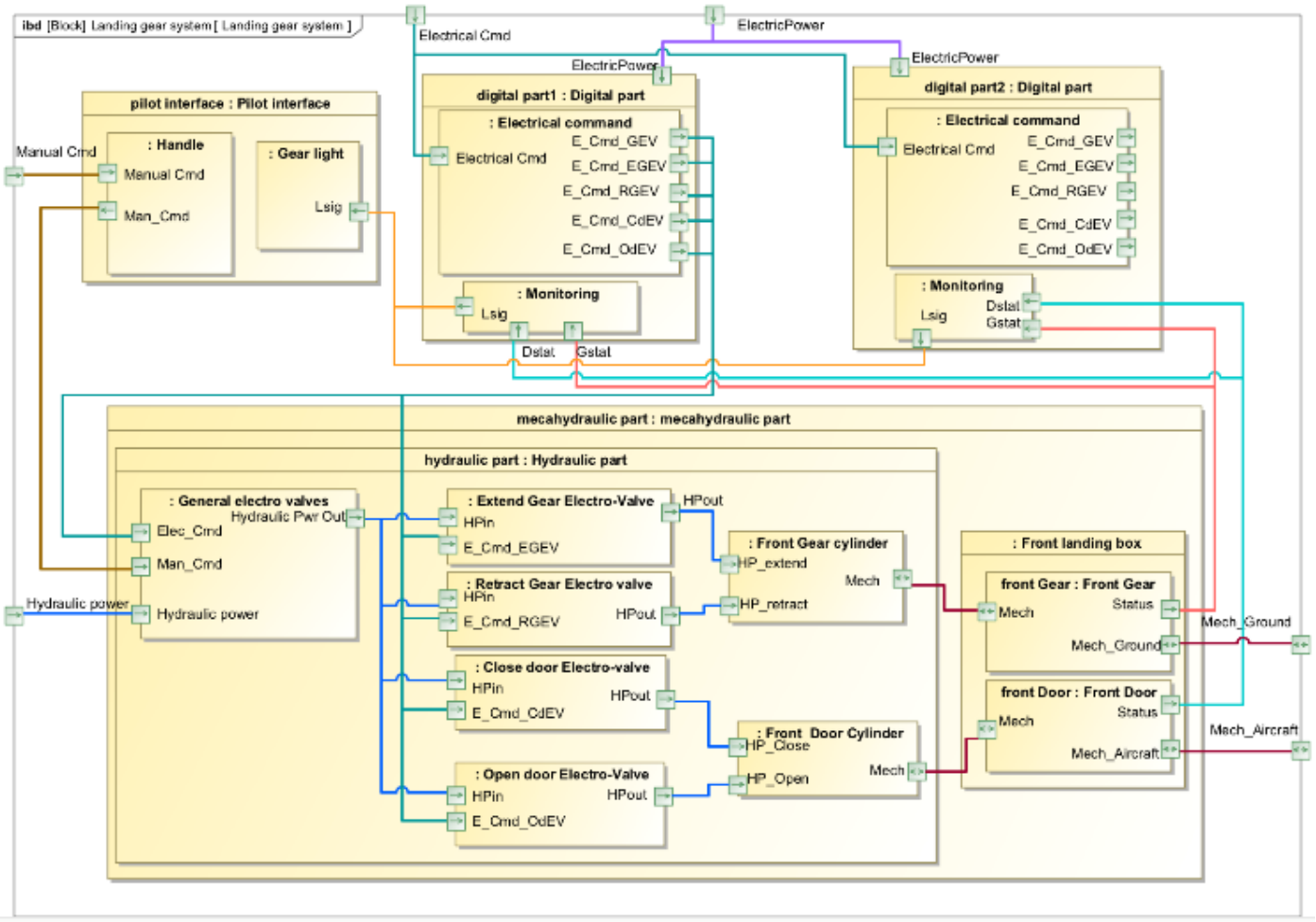

Fig. 5 : System Physical Architecture (IBD)

simulated by our tool. The components of our system are described in classes that are instantiated in the main block. Fig. 7 describes the main block representing the system and Fig. 8 and Fig. 9 describes the class ElectroValve and Cylinder which are components of the landing gear system. Instances of those components are linked in the system to allow for retraction and extension of the landing gear with assertions such as the one presented in Fig. 6, this allows for the input of the cylinder to be at all time equal to the value of the output of the electrovalve.

```
digitalPart.CPIOM1.input:=
                pilotInterface.udHandle.output;
```

Fig. 6 : Example of assertion in AltaRica

All components in our system are extending the NonReparaibleComponent class presented in Fig. 4. This class describes the state machine for failure of a component. Those events are characterized by delays. This allows the execution to compute the time after which the transition shall be fired. Delays are described using probability distributions such as, in our study case, an exponential distribution. They also allow for computation of probability of event happening for the generation of fault trees in the case of static systems.

From this general class we derive all components of the system, specializing this class by adding new variables that are ports of our components and assertions that represents connections between these variables. As an example Fig. 6 represents such a connection. It means that the output value of the udHandle component of the pilot interface is given to the input of the CPIOM1 component of the digital part.

This means that during the execution, the value of the output of the udHandle component, which is the handle used by the pilot to actuate the landing gear, will be given to the input of the first CPIOM unit of the digital part. Assertions can also be used to give values to variables based on component state or other information.

The interest of having this formal representation of the system rather than using a notation such as SysML is that it allows for formal computation over the system safety. Thanks to this model we are able to compute minimal cut sets of the system with their probabilities. We can also execute stochastic simulation of the system with failures, and identify propagation paths of the failures. This wouldn't be possible if there was any ambiguity in the representation of the system. Whereas for human communication with the MBSE model this isn't an issue.

```
block LandingSys
  PilotInterface pilotInterface;
  MechaHydraulicalPart mechahydraulicPart;
  DigitalPart digitalPart;
  assertion
    digitalPart.CPIOM1.input :=
                pilotInterface.udHandle.output;
    […]
end
```

Fig. 7 : Block Landing System in AltaRica 3.0 (some assertion were hidden for clarity)

```
class ElectroValve
  extends NonRepairableComponent
                              (lambda = 1.0e-6);
  Boolean input, output, order (reset = false);
  assertion
    output := if s == OK then input and order
              else false;
end
```

Fig. 8 : Class ElectroValve in AltaRica 3.0

```
class Cylinder
  extends RepairableComponent;
  Integer input (reset = 0);
  Boolean output (reset = 0);
  assertion
    output := if s == OK then input else false;
end
```

Fig. 9 : Class Cylinder in AltaRica 3.0

IV. Typology of differences between MBSE and MBSA

MBSE and MBSA are two important parts of the design of the system, but as we explained it before, they serve two different purposes. Therefore, differences occur between those models, should it be for lack of communication between the teams or for more fundamental reasons linked to the nature of those models. These differences could lead to models presenting two distinct and different systems instead of representing the same model. Synchronization between MBSE and MBSA models is thus necessary to ensure consistency. Work is in progress to create consistency methods that will be described in IV.A. We think that in order to improve upon this work, there is a need for a formal definition of inconsistencies. In order to prepare this definition we established the typology of differences that is described in IV.B.

A. *Existing methods for MBSE/MBSA synchronisation*

As synchronization between heterogeneous models is an important concern for researchers in the modeling field, some tools were already developed to help model synchronization, even in the case of MBSE/MBSA synchronization.

In [9], the authors suggest a synchronization methodology based on three phases: Abstraction, Comparison and Concretization, illustrated in Fig. 10. The different models are first translated to intermediary models written in a same formalism that will allow comparison, this is the abstraction phase. Those intermediary models are then compared to detect the differences that exist between them, this is comparison. Finally the source models are annotated with the differences that are detected, and corrective actions are proposed to the designers, this is concretization.

The SmartSync synchronization framework [3] does provide a methodology and tools for synchronization between MBSE and MBSA derived from this methodology. This is achieved by the abstraction of both models to a common formalism, the S2ML language [1], then computational comparison of the two abstracted models and finally a concretization step, where inconsistencies are taken into account and the models are adjusted to match each other.

The work proposed in [4] has a similar perspective as the ones in [9] proposal. It focuses on the topological aspect of models, and performs the comparison relying on graph theory after abstracting the different models into graphs. This methodology carries a flat view of the model. This methodology is to be applied by an "interface expert" that communicates with specific model designers.

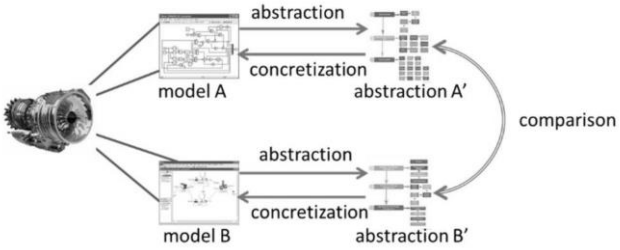

Fig. 10 : Model Synchronization approach [9]

The MOISE project provided the system engineer and safety analyst with a method to create "synchronisation points" that are linked on one side to elements of the MBSE model and to elements of the MBSA model on the other side. This allows the users to review consistency by reviewing each consistency point and attributing it a status and a rationale to justify that elements of MBSE and MBSA models represent the same component.

All those solutions highlight the need for an intermediate representation of the models that enables the comparison. However we feel that there is still a need for a typology and a formal definition of the inconsistencies between MBSE and MBSA models.

B. *Comparison of models*

In order to identify the types of differences, we reviewed both MBSE and MBSA models together and searched for all differences that occurred between them. We will not cite all of them here because some are the same on different elements of the model but we will list all different ones.

The first differences that occur when reviewing the models are the names of the elements of the model. The names can vary between both models, for example the handle from the pilot interface is called "Handle" in the MBSE model and "udHandle" in the MBSA model. Naming differences are linked to the system engineer and the safety analyst calling the same elements in different ways, which can happen because of their different technical backgrounds, or their naming practices (upper/lower cases, shortcuts…). Other naming inconsistencies may come from the naming rules (that can be either restrictions or only recommendations) in both tools. For example the pilot interface subsystem is called "Pilot interface" in the MBSE model and "pilotInterface" in the AltaRica 3.0 model, this difference is caused by the differences of uses of the two models: AltaRica 3.0, as a formal modeling language, is used to compute reliability or safety indicators, thus objects shall be represented in one word; whereas SysML is used to declare and communicate and as a consequence, there are no restrictions on naming objects.

It also happens that some elements of both models can sometimes not be named, whereas their counterpart in the other model is. Assertions in AltaRica 3.0 are unnamed, they serve a pure purpose of simulation, defining rules for the calculation of variables values. On the contrary connections in the SysML model are sometimes named using the type of flow or actions they convey, when the ports they connect are often unnamed by the engineer and automatically named p1, p2, p3… by the modeler. As the previous one, this difference is also caused by the differences of uses of the two models and by the modeling habits of both engineers.

Most variables in the AltaRica model are typed with discrete values, this is explained by the fact that we are mainly interested in knowing whether they are nominal or dysfunctional and not by their accurate values. The system engineer and the safety analyst do not have the same point of view on the system. Therefore they do not represent system variables in the same way. Those variables type are sometimes not affected to the ports in the SysML model but to the connections, depending on the engineer modeling habits. Because the system engineer wants to prove that the system answers requirements that can be linked to those values, so they are usually typed as their physical unit.

We also observe different structural differences between our models.

The first one is that some subsystems are specified at different levels of abstraction. For example, the arborescence for the electro-valve that brings hydraulic pressure to the door extension cylinders is "Landing gear system\mecahydraulic part\Hydraulic part\Open door Electro-Valve" in the MBSE model and "LandingSys\mechahydraulicPart\hydraulicSys\ DoorHydraulicSys\extensionDoorElectroValve". Although the naming differs, subsystem levels between those paths match apart for the "DoorHydraulicSys" level in the MBSA model that does not match any subsystem in the MBSE model. This originates from the Safety analyst regrouping parts in a different way, which may be due to his wish to only consider failure of a group of parts rather than every unique part, and it could also happen that the MBSA model only specified "DoorHydraulicSys" without modeling the electro-valve inside itSome components have also been specified at a different place of the Product Breakdown Structure. This is the case of the cylinders in our comparison. The system engineer considered them to be part of the hydraulic set of parts, whereas the safety analyst did put them in the landing sets along with the gears/doors they are connected to. Such a difference could be caused either by a different point of view over the system as it is here, or by a modeling error.

Finally, in the Internal Block Diagram we observe some connections to the outside of the system that are not considered in the AltaRica model, for example the "Electric Power" input, this is due to the need for the system engineer to represent all interactions within the system and with its environment. However, even though this connection has a real impact on the system, it was not considered relevant for safety analysis of the landing gear system and thus not modeled with AltaRica. Such a difference could be either considered a modeling error or not depending on whether that connection has an impact or not on safety analysis.

We note that although it is not the case in our models, some connections could have been placed between ports that do not necessarily exist in the MBSE or MBSA representations of the system. This could either occur by modeling error, or because those values are not relevant to one or the other modeling intent.

C. Typology of differences

From this comparison we deduce three main types of differences in our model that are in fact related to the cause of the differences between the models. Some of those differences are caused by modeling errors and lead to the models describing different systems, those are inconsistencies, but we also note differences that are due to modeling practices and tools.

The first type of differences we encountered is related to differences that are caused by the different modeling tools and practices. Examples of this are different naming rules or connections between ports or assertions that relate variables carrying names in different ways. This type of differences could be handled by modeling practices or rules in certain cases, for example the implementation of naming rules in SysML similar to the ones that exist in other modeling languages. It could also be taken into account in the comparison by not taking into account names that have no counterpart, or by cleverly comparing them, for example in our case, comparing connections names in SysML to variables names in AltaRica.

Type	Observed differences
Due to Modeling tools and Practices	Different naming rules
	Different name meaning
	No naming counterpart
	Different abstractions of subsystems
Due to Modeling Intent	Different variable types
	Different model arborescence
	Different interactions/connections in and with outside the system
Due to Modeling Errors	Wrong model arborescence
	Wrong Variable Values/Types
	Wrong connections between system components

Fig. 11 : Typology of differences associated to examples from VI.B

The second type of differences that we denote are differences linked to modeling intent. This is the case of the difference in abstraction that could occur between both modeling as we showed it with the electro-valve, or in the case of the different value types that are observed. These differences are necessary for both the system engineer and safety analyst to work correctly. Their existence is the reason for having two separate models rather than modeling all information in one global model.

Our third type of differences are the ones caused by modeling errors. The aim of model synchronization is to eliminate those differences. They can be different naming, wrong values, incorrect links between components, etc. These inconsistencies will probably be more difficult to recognize from the second type of differences (modeling intent) than the first (modeling practices and tools).

We also raise another interesting way to classify differences. We encountered differences between the models that were either related to a particular element of the system, such as a naming difference or a variable type difference, or differences that were related to the structure of the model, such as abstraction differences, different placement of an element in the Product Breakdown Structure, or wrong connections between component ports/variables. These two kinds of differences are also interesting because we intuit that they should translate very differently in a mathematical framework around the models.

V. Conclusions and perspectives

Since system engineering and safety analysis are serving different purposes, they need to be using two different models and two different formalisms. This results in a risk of inconsistency between those models.

In this paper, we proposed a typology of the differences that may exist between MBSE and MBSA models. This is made possible by MBSA and MBSE modeling of a reference system, and analyzing the differences noticed between the two models.

This typology shows that differences can be sorted by their causes. These causes are the use of different modeling tools and practices, different modeling intents, and finally modeling errors. We think that differences due to modeling intents and standards are important to the models since they are the reason for two models being used instead of one, and they should be preserved. Whereas inconsistencies due to modeling errors should be eliminated from the model, and differences due to different modeling tools and practices should be reduced as much as possible by unifying naming practices for example.

On the basis of this typology, we want to write a formal definition of what is an inconsistency between MBSE and MBSA models. This will be helpful in the formalization of a consistency assessment method and model reconciliation. We think that reconciliation requires to develop strategies to eliminate unacceptable inconsistencies, i.e. ones caused by modeling errors, and reduce other differences to an acceptable threshold.

Moreover, in further work, we believe that the formalization of a mathematical framework supporting those models will help us with consistency assessment, and therefore we intend in formalizing the S2ML language used in the SmartSync methodology with the category theory formalism, to allow better comparison between models, and easier translation from SysML and AltaRica to S2ML.

Acknowledgments

The authors want to thank the S2C project at IRT SystemX and IRT Saint-Exupéry and all its industrial and academic partners for the funding of this research and the trust they give us.

References

[1] M. Batteux, T. Prosvirnova, and A. Rauzy. “From Models of Structures to Structures of Models”. In IEEE International Symposium on Systems Engineering (ISSE 2018). Roma, Italy. October, 2018.

[2] M. Batteux, T. Prosvirnova, and A. Rauzy. “AltaRica 3.0 in 10 Modeling Patterns”. In International Journal of Critical Computer-Based Systems. Inderscience Publishers. Vol. 9, Num. 1–2, pp 133–165, 2019

[3] M. Batteux, J.Y. Choley, F. Mhenni, T. Prosvirnova, and A. Rauzy. “Synchronization of System Architecture and Safety Models : a Proof of Concept.”. ”. In IEEE International Symposium on Systems Engineering (ISSE 2019). Edinburgh, Scotland, UK. October, 2019

[4] A. Berriche, F. Mhenni, A. Mlika and J.Y. Choley. “Towards Model Synchronization in Model Driven Engineering of Mechatronic Systems.”. In IEEE International Symposium on Systems Engineering (ISSE 2019). Edinburgh, Scotland, UK. October, 2019

[5] F. Boniol., and V. Wiels. “The Landing Gear System Case Study”. In ABZ 2014. Communications in Computer and Information Science, vol 433. Springer Cham. 2014

[6] S. Friedenthal, A. Moore, and R. Steiner. “A Practical Guide to SysML”. In A Practical Guide to SysML. 2008

[7] International, S. A. E. “Guidelines and methods for conducting the safety assesment process on civil airborne systems and equipment ARP4761”. 1996

[8] F. Mhenni, J.Y. Choley, O. Penas, R. Plateaux, and M. Hammadi. “A SysML-based methodology for mechatronic systems architectural design”. In Advanced Engineering Informatics, Vol. 28(3), pp 218–231, 2014

[9] A. Rauzy, and C. Haskins “Foundations for model-based systems engineering and model-based safety assessment”. Systems Engineering, Vol. 22(2), pp 146–155. 2018