applied sciences

MDPI

Article

Category Theory Framework for System Engineering and Safety Assessment Model Synchronization Methodologies

Julien Vidalie [1,2,*], Michel Batteux [1], Faïda Mhenni [2] and Jean-Yves Choley [2]

[1] IRT SystemX, 8 Avenue de la Vauve, 91120 Palaiseau, France; michel.batteux@irt-systemx.fr
[2] Laboratoire Quartz—ISAE Supméca, 3 Rue Fernand Hainaut, 93400 Saint-Ouen, France; faida.mhenni@isae-supmeca.fr (F.M.); jean-yves.choley@isae-supmeca.fr (J.-Y.C.)
* Correspondence: julien.vidalie@irt-systemx.fr

Abstract: In recent decades, there has been a significant increase in systems' complexity, leading to a rise in the need for more and more models. Models created with different intents are written using different formalisms and give diverse system representations. This work focuses on the system engineering domain and its models. It is crucial to assert a critical system's compliance with its requirements. Thus, multiple models dedicated to these assertions are designed, such as safety or multi-physics models. As those models are independent of the architecture model, we need to provide means to assert and maintain consistency between them if we want the analyses to be relevant. The model synchronization methodologies give means to work on the consistency between the models through steps of abstraction to a common formalism, comparison, and concretization of the comparison results in the original models. This paper proposes a mathematical framework that allows for a formal definition of such a consistency relation and a mathematical description of the models. We use the context of category theory, as this is a mathematical theory providing great tools for taking into account different abstraction levels and composition of relations. Finally, we show how this mathematical framework can be applied to a specific synchronization methodology with a realistic study case.

Keywords: system engineering; safety assessment; multi-physics; consistency

Citation: Vidalie, J.; Batteux, M.; Mhenni, F.; Choley, J.-Y. Category Theory Framework for System Engineering and Safety Assessment Model Synchronization Methodologies. *Appl. Sci.* **2022**, *12*, 5880. https://doi.org/10.3390/app12125880

Academic Editor: Agostino Forestiero

Received: 29 April 2022
Accepted: 6 June 2022
Published: 9 June 2022

1. Introduction

During recent decades, systems have become more complex as they evolved towards having multiple features and due to the rise of mechatronic design. This complexity leads to a need to create numerous models in system design. We use these models to represent the product towards diverse modeling intents, thus allowing the analysis of specific properties.

Among these models, we find system engineering models used to apprehend complexity in the system. System engineering encompasses many models, including requirement and architecture models, safety models, and multi-physics models. These models are written by different people, using various tools and formalisms. This independence can lead to problems with consistency between the models. As diverse models represent one system, we fear that some models may contain errors, making them incompatible with the designed system. Such inconsistencies can be very concerning in the case of safety-critical systems and their safety models, and manual review of the models is time expensive. This is especially true of complex systems that can interact with humans, such as autonomous vehicles or cobots. Autonomous systems are complex and critical systems with highly complex control loops [1] that require advanced models when engineers simulate them. An undetected error in the safety model could lead to not considering some of the system's potential failure events. Such errors would then result in the system being dangerous for the user. When systems have to be certified in order to be accepted into the market, e.g., for planes, the safety assessment methodology needs to be accepted by safety authorities.

This requires great trust in the safety assessment process and, therefore, in the consistency of the models.

The system engineering community proposes different methodologies such as model federation [2] or proof theory approach [3] to assert and maintain consistency between these models to face this challenge. Amongst them, we find synchronization methodologies [4]. In this work, we address the mathematical aspect of the synchronization methodologies. The synchronization methodologies consist of three steps: first, we translate the models to a common formalism; then, we compare them in this common formalism; and finally, we carry the comparison results back to the original models.

Although those methodologies exist and provide means to minimize the number of contradictions between the models, there is no formal definition of what consistency the community would widely accept. We primarily understand consistency by the absence of contradiction between the models, but this is not a criterion that single-handedly allows a methodology's efficiency to be demonstrated. Thus, we propose a formal definition of a binary consistency relation.

In this work, we propose to use category theory to interpret the model synchronization through a mathematical framework. We aim to provide a tool that helps design and validate model synchronization methodologies mathematically. To achieve this goal, we give a mathematical representation of the comparison models that the synchronization methodologies use, and we propose a definition of a general consistency relation between models. We apply this mathematical framework to the SmartSync methodology [5] through the study of model consistency in the design workflow of an autonomous critical system, a fixed-wing blood delivery drone. In the SmartSync methodology, the comparison models are mathematically depicted through quintuples that contain their characteristics. The mathematical framework described in this work proposes an equivalent definition of the comparison models, representing the models through categories rather than quintuples. This new definition allows our mathematical framework to give a new mathematical perspective of the comparison, which allows for mathematical proofs of the properties compared between the models. In the study case, through the categorical framework, we detect some limitations of the SmartSync methodology and propose some leads to overcome them.

The remainder of this paper is structured as follows:

Section 2 establishes the state of the art for the models, synchronization methodologies, and category theory tools that we use in this work. Section 3 defines the mathematical framework we propose for synchronization methodologies. Section 4 gives an example of applying the mathematical framework with the SmartSync methodology over a study case. Section 5 discusses some points of interest and the results of this study.

2. State of the Art

2.1. Model-Based System Engineering and Safety Assessment

During system design, engineers create representations that they use for different intents. We study the consistency of system engineering and safety assessment representation in this work, especially model-based ones. System engineering is an interdisciplinary approach for designing and integrating complex systems throughout their life cycle, from their design to their end of life. It is achieved by representing the system's requirements, environment, life-cycle, functions, and architecture. Safety assessment aims at asserting that the system complies with its safety requirements. Whereas traditional approaches have been document-centered, industrial practices shift towards model-based approaches. Model-based approaches provide new tools that better consider complexity in system design. This change in modeling paradigm induces new engineering disciplines.

2.1.1. Model-Based System Engineering (MBSE)

The usual approaches for system engineering have been using documents to describe the requirements, environment, architecture, and other views of the system. Although

those documents are sometimes created in tools such as Excel or Doors, they are textual and rely on natural language.

The model-based approach used in system engineering finds its origin in model-based software development [6]. Software developers use such approaches to represent software architecture, with tools such as the UML notation [7]. This language was derived into the SysML notation [8], which system engineers use to describe complex systems.

The model-based system engineering tools such as SysML or Capella use diagrams to represent different system views. Structural and behavioral aspects are represented, with diagrams such as requirement diagrams allowing for representation of the requirements, activities diagrams allowing for representation of the functions, and block definition and internal block diagrams allowing for representation of the architecture. The MBSE languages present a lack of semantics [9], which causes ambiguities without the use of a third-party methodology. Therefore architecture grids—such as CESAM [10] or the methodology from [11] for SysML or Arcadia for the Capella tool—provide guidelines to create MBSE models.

Although we often use the name MBSE for such diagrammatic models, system engineering encompasses various disciplines. Therefore, some domain-specific languages (DSL), such as languages for multi-physics or safety assessment, can be interpreted as part of MBSE.

2.1.2. Model-Based Safety Assessment (MBSA)

Similarily to MBSE, the field of safety analysis is faced with the challenge of complexity. The traditional approaches, such as Petri nets, reliability block diagrams, event trees, and fault trees, are models that are very close to mathematical equations and lack structure or lack expressivity power. They also are very abstract compared to the system specification. The model-based safety assessment (MBSA) approaches try to answer these limitations.

The MBSA models have a high expressivity, as they represent systems with a point of view closer to their architecture. The gain on expressive power is obtained by going from Boolean formalisms to state or event formalisms. There are three different kinds of model-based risk and safety assessment formalisms [12], including specialized profiles of MBSE tools such as [13], approaches that extend fault trees or reliability block diagrams such as dynamic fault trees [14] and model-based safety and reliability assessment modeling languages. In this work, we are interested in MBSA modeling languages, such as SAML [15], Figaro [16], and especially AltaRica 3.0 [17].

AltaRica 3.0 is the third version of the AltaRica language; it is based on the mathematical framework of guarded transition systems (GTS) [18], which are state automata.

2.1.3. Structural Models

The MBSE and MBSA models are two different kinds of models that represent the same systems with two different modeling intents. The modeling intent is the primary constraint on the representation of the system in these models.

In [19], the authors state that models are composed of behavior and structure. This means that these models are composed of a mathematical framework and a structural construction. In this work, we are interested in comparing the structure of the models, as we want to show that the architectures of the systems represented by the models are the same.

The article also introduces S2ML (System Structure Modeling Language), a language designed to model systems structure. This language is built around basic elements: blocks, ports, and connections. These three basic concepts, although very restricted, allow for a robust conceptualization of systems. Many languages can be interpreted through them, such as SysML, AltaRica, Modelica, and Lustre.

2.2. Synchronisation Methodologies

Different approaches exist to assert consistency between heterogeneous models. The model federation [2] approach uses a database to store all data from a system's models. Within that database, the user can assert consistency and traceability between the models. Proof theory is used by [3] in their inconsistency management approach. This methodology interprets models as first-order logics, identifying inconsistencies through a computer-aided proof tool. The lack of expressivity of proof theory limits this approach, making the interpretation and proofs difficult.

A third approach to heterogeneous models consistency is model synchronisation [4]. The synchronization consists of a three-step process, which is illustrated in Figure 1:

- Abstraction: The models are translated to a common formalism.
- Comparison: The comparison is operated in the abstracted models. Writing the compared models in a common formalism makes the comparison easier than with heterogeneous models.
- Concretization: The comparison results are carried back to the original models. Concretization can be achieved by correcting the models, annotations, or other means.

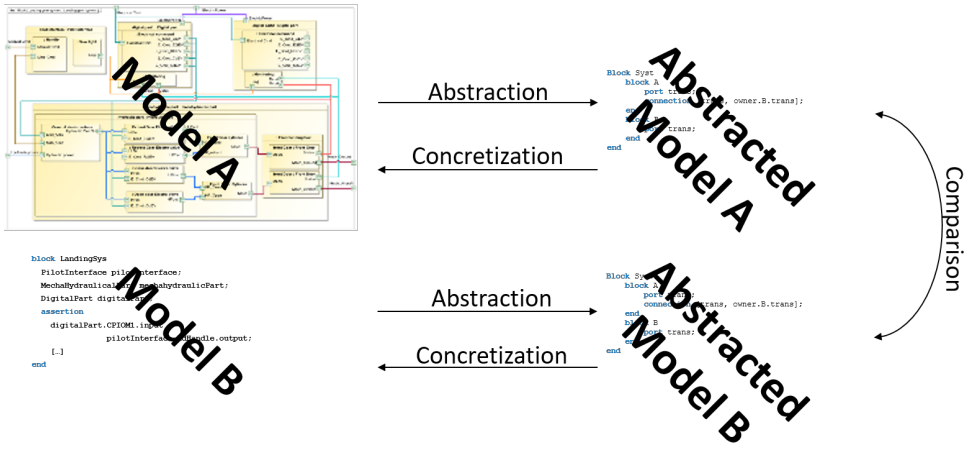


Figure 1. Model synchronisation approach.

The model synchronization approach is the base for different methodologies. In this work, we intend to interpret them in a mathematical framework.

Regarding the SmartSync synchronization framework [20], the authors of the methodology provide a methodology and tools for the synchronization of the MBSE and MBSA models [21]. The comparison is operated on abstracted models written with the S2ML language [22]. The S2ML comparison models used in SmartSync are described with a set-theoretical point of view as a quintuple $< P, C, B, \alpha, r$ [5], where:

- P and B are sets of symbols called ports and blocks, respectively;
- C is a multiset of subsets of P called connections;
- α is a subset of $B \times (P \cup C \cup B)$ such as any element of $(P \cup C \cup B)$ that is associated with, at most, a unique element of B called its parent, and there exists a unique block $r \in B$ with no parent.

SmartSync relies on the user to manually align model elements, as described in Section 4.2.1.

The directed graph methodology from [23] is also a synchronization methodology. It does rely on the abstraction of the models towards graphs. In the graphs, vertices represent parts and ports from SysML and blocks and classes from AltaRica; edges represent connectors from SysML and assertions from AltaRica. This framework allows for the use of graph search algorithms to traverse the graphs, and the use of concepts such as graph and sub-graph isomorphisms to compare the models.

The consistency links methodology [24] is another synchronization approach that creates links between MBSE and MBSA models. This methodology mainly focuses on the functional aspect of the system.

The three latest methodologies provide a structural comparison between system engineering and safety models. They rely on comparison models that follow the block, ports, and connection structure described in Section 2.1.3, with different ways to represent them. Category theory offers a point of view that allows us to use both the strengths of the graph and set theories, along with other topological tools, which is why we use it to provide a mathematical framework for synchronization methodologies. We do not find a clear formalisation of the consistency notion that seems to be widely accepted by the community. We provide a mathematical framework within which we define an axiomatic for the representation of models and propose a formal definition of a consistency relation.

2.3. Category Theory

Category theory is a branch of mathematics that finds its origins in the work of Samuel Eilenberg and Saunders Mac Lane in the early 1940s. This theory aims at unifying mathematics, especially by creating a bridge between topology and algebra. After category theory was used for many years to redefine existing concepts, Alexander Grothendieck did build completely new mathematical objects using it [25]. In this work, we use category theory to allow us to describe objects through their interactions. We detail in this section some basic knowledge of category theory that is required to understand this paper. A more complete yet well-written and easily understandable introduction to category theory can be found in [26].

2.3.1. Category

Category theory studies categories, which are mathematical entities that contain objects and relations between objects.

In this context, mathematicians use relations to study the properties of objects rather than looking at the objects themselves. It is also possible to define relations between categories called functors; this leads to defining more extensive categories within which the objects are the categories and the relations are the functors. Although this can sound complicated when first encountered, it provides a way to represent abstraction levels; this concept also leads to the definition of some fundamental concepts for logic that we do not use here.

We choose to use category theory in this work because of its properties that encompass both set theory and graphs. At the same time, category theory also has essential composition properties that allow propagating relations in the models we study and abstraction properties that are interesting in model comparisons.

Definition 1. *Category*
A category C is composed of:

- *A class Ob(C) of objects;*
- *For $x, y \in Ob(C)$, the set $Hom_C(x, y)$ of morphisms from x to y;*
  Morphisms are often called arrows;
  This set is called the Homset of x and y.

A category respects the following rules:

- *Composition: If $f \in Hom_C(x, y)$ and $g \in Hom_C(y, z)$, then $g \circ f \in Hom_C(x, z)$.*
- *Associativity: If f,g,h are morphisms such that $g \circ f$ and $h \circ g$ exist, then $(h \circ g) \circ f = h \circ (g \circ f)$.*
- *Identity: for each $x \in Ob(C)$, there exists an identity morphism $Id_x : x. \longrightarrow x$*

We can give a graphic representation of a category through an oriented graph. This representation is called a diagram over the category. Vertices represent the objects of the category, and edges represent the morphisms. A diagram does not have to be exhaustive

over the category. Thus, we can draw a finite diagram over a category with an infinity of objects (such as the $S2ML + Cat$ category that we define in Section 3). We often do not show identities and morphisms resulting from diagrams' composition, which allows better readability.

Just like other mathematical objects, we can link categories through applications. The right concept of application between categories is functors, a categorical pendant of morphisms.

Definition 2. *Functor*
For two categories C, D, a functor F is an application $F : C \longrightarrow D$ composed of:

- *A function $F_{Ob} : Ob(C) \longrightarrow Ob(D)$;*
- *For each $x, y \in Ob(C)$, a function $F_{x,y} : Hom_C(x, y) \longrightarrow Hom_D(F_{Ob}(x), F_{Ob}(y))$.*
  We can note F(x) or F(f) if x is an object and f a morphism.

A functor $F : C \longrightarrow D$ satisfies the following requirements:

- *Identities are preserved by F, i.e., for $x \in Ob(C)$, $F(Id_x) = Id_{F(x)}$;*
- *Composition is preserved by F, i.e., for $x, y, z \in Ob(C)$ and $f \in Hom_C(x, y)$, $g \in Hom_C(y, z)$, we have $F(g \circ f) = F(g) \circ F(f)$.*

There exist applications between functors called natural transformations. Natural transformations are mappings for each object and morphism of a category between the images of their images by two functors. In this aspect, they allow transforming a functor into another functor. They are essential in defining the equivalences of categories.

Definition 3. *Natural transformation*
Let C and D be two categories.
Let $F : C \longrightarrow D$ and $G : C \longrightarrow D$ be two functors.
A natural transformation $\alpha : F \longrightarrow G$ is composed for each $x \in Ob(C)$ a morphism $\alpha_x : F(x) \longrightarrow G(x)$ such that for each $y \in Ob(C)$, for each $f \in Hom_C(x, y)$, we have:
$G(f) \circ \alpha_x = \alpha_y \circ F(f)$
This corresponds to saying that the diagram found in Figure 2 commutes.

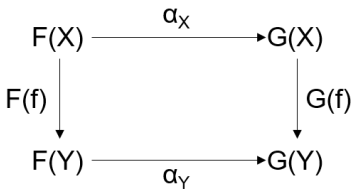


Figure 2. Illustration of a natural transformation.

2.3.2. Some Interesting Concepts

We have defined the basic concepts of category theory; we will now focus on some properties of interest to this work.

In the same way as functions can be injective, surjective, or both (bijective), there are similar concepts for functors between categories.

Definition 4. *Full and Faithful functors*
Let $F : C \longrightarrow D$ be a functor.
We say that F is Full if for any $x, y \in Ob(C)$, $F_{x,y} : Hom_C(x, y) \longrightarrow Hom_D(F(x), F(y))$ is a surjection. We say that F is Faithful if for any $x, y \in Ob(C)$, $F_{x,y} : Hom_C(x, y) \longrightarrow Hom_D(F(x), F(y))$ is an injection.

We can also define the concept of equivalence between two categories.

Definition 5. *Equivalence of categories*
Let C and D be categories.
An equivalence of categories is a functor $F : C \longrightarrow D$ such that there exists:

- *A functor $G : D \longrightarrow C$;*
- *Natural transformations $\alpha : Id_C \longrightarrow G \circ F$ and $\alpha' : Id_D \longrightarrow F \circ G$.*

The notion of pullback allows defining an object that is the biggest object that respects a constraint towards two objects. Prerequisites for this notion are cones and limits, so we shall define them first.

Definition 6. *Cone*
Let C be a category. Let there be a diagram over C. A cone over this diagram is an object Z with morphisms from Z to every object of the diagram, such that any newly formed triangle commutes.

A limit is a universal cone, i.e., a cone such that, for any other cone, a unique path exists from the limit to the other objects in the diagram through the cone.

Definition 7. *Limit*
Let C be a category. Consider a diagram over C. A limit on this diagram is a cone Q such that for any other cone Z we have a unique morphism $\psi : Z \longrightarrow Q$ such that all triangles including ψ commute.

Cones and limits are illustrated in Figure 3. The diagram considered is constituted of the A, B, C, and D objects and morphisms between them. Z and Q, assuming that compositions of morphisms of the diagram with the dotted arrows commute, are cones. Furthermore, Q is a limit of the diagram if for any cone Z, there is a unique morphism $\Psi : Q \longrightarrow Z$ that makes everything commute. We say that a limit is a universal cone because whichever cone we choose, there is a unique way to obtain this cone from the limit, and this way is the ψ morphism.

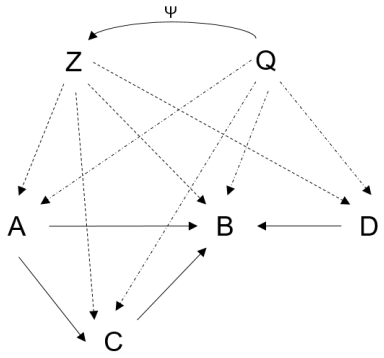


Figure 3. Illustration of the cone and limit.

Definition 8. *Pullback*
Let A, B, and C be three objects of a category. Let f and g be two morphisms from A and B to C. A pullback in a category is the limit Ax_CB of the diagram from Figure 4.

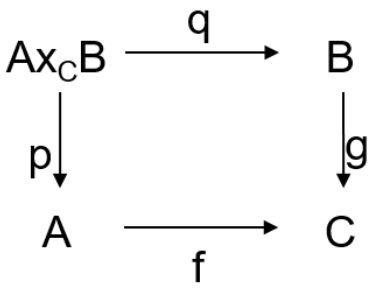


Figure 4. Structure of the pullback.

The pullback is an object with morphisms towards two objects that respects constraints with a third object. This can be interpreted as the biggest object that respects the constraint.

2.3.3. Use of Category Theory in System Engineering

Because category theory is a tool that encompasses abstraction levels and interactions between objects, some approaches have given representation of complex systems through this mathematical frame.

Fundamental mathematical approaches give highly theoretical representations of these systems. Dynamic systems can be represented using the concept of sheaves [27], which are functors from a category to the category of sets with compelling topological properties. This allows for the representation of interactions between multiple dynamic systems. Category theory can also be used to represent systems in a way that allows for straightforward representation of hierarchy levels with evolutive memory systems [28], through the evolution of the system, and interactions between multiple systems.

Some approaches are closer to the system engineering point of view. Ontologies and category theory are used to represent MBSE models in the MB2SE framework [29]. A category-based meta-model of system engineering and safety assessment was also designed [30].

3. Mathematical Framework for Consistency

3.1. Mathematical Representation of a Structural Model

In this section, we assume that a model that carries structural information can be described with the following elements:

- Blocks;
- Ports;
- Connections.

This is at least the case with the comparison formalisms used in the synchronization methodologies from Section 2.2, although they are named differently in the consistency links methodology. In [5], the S2ML models are formalized with a quintuple composed of sets of ports and blocks, a multiset of connections, a composition relation, and a unique block with no parents regarding the relation.

Although this description is rigorous, we propose another definition inspired by category theory which allows for the definition of functors between the models. To give a categorical definition of S2ML models, we need to define four concepts: ports, connections, blocks, and then S2ML models.

Definition 9. *Port*
A port is a singleton containing a symbol p.

The ports are atomic elements of the model. They can be used to express variables in the model, states of a component, or other properties. We use symbols to represent this property; the symbol could be interpreted as the port's name.

Definition 10. *Connection*
A connection is a non-empty, finite set of ports.

This definition means that a connection is a category that contains the ports that it connects.

We can now define the blocks with ports and connections, which are containers in the model. A block is a container that holds other blocks, ports, and connections. Therefore the definition of a block is recursive.

Definition 11. *Block*
A block is a category B such that:

- $Ob(B)$ *is a finite set of blocks, ports and connections;*
- *For each connection C, if a port* $P \in Ob(B)$ *is such that* $P \in Ob(C)$*, then there exists two functors* $r_{P,C} : P \longrightarrow C$ *and* $r'_{P,C} : C \longrightarrow P$ *in B such that* $r_{P,C}$ *maps* $p \in P$ *to* $P \in Ob(C)$ *and* $r'_{P,C}$ *maps every element of C to* $p \in Ob(P)$ *and every morphism to* Id_p*.*
- *For each block* $B_1 \in Ob(B)$*:*
  - For each connection $C \in Ob(B_1)$*, we have* $C \in Ob(B)$ *and there is a morphism* $\alpha_{C,B_1} : C \longrightarrow B_1$ *in B that maps each element of C to* $C \in Ob(B_1)$*;*
  - For each port $P \in Ob(B_1)$*, we have* $P \in Ob(B)$ *and there is a morphism* $\alpha_{P,B_1} : P \longrightarrow B_1$ *in B that maps* $p \in P$ *to* $P \in Ob(B_1)$*;*
  - For each block $B_2 \in Ob(B_1)$*, we have* $B_2 \in Ob(B)$ *and there is a morphism* $\alpha_{B_2,B_1} : B_2 \longrightarrow B_1$ *that maps every object of* B_2 *to itself in* B_1 *and does the same with morphisms;*
  For each X in $Ob(B_1)$*, there exist a unique block* $B' \in B$ *(possibly* B_1*), such that there is no block* $B'' \in B'$ *such that* $X \in B''$*; we also have either* $B' = B_1$ *or* $B' \in Ob(B_1)$*;*
- *There are no other morphisms in the block than those described here and the morphisms that derive from category theory axioms, i.e., identities and compositions.*

Although this definition can look very complicated, it simply carries the properties of a block to the world of categories. Thus, each point of the definition can be explained straightforwardly.

We define two morphisms between a connection and the ports that are in the block and the connection. These morphisms represent the relation between the ports and the connection. The ports are both parts of the connection, and the connection refers to them. The typical structure of a connection in a block is depicted in Figure 5; it corresponds to the connection from Figure 6. We do not assume that all ports of the connection are contained in the block. In a model, we will eventually find an ancestor block containing all the connection's ports. However, connections usually go from one block to another. Thus, there is no reason for all ports to be contained in any block that contains the connection.

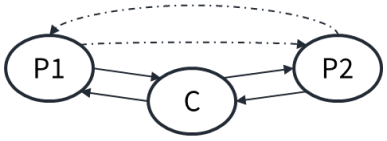


Figure 5. Structure of a connection with two ports in a block/model.

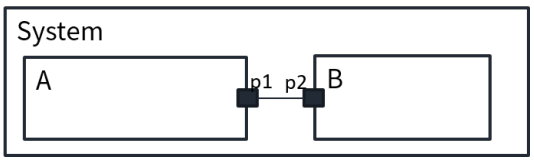


Figure 6. Example of a system model.

In Figure 5, the plain arrows represent the $r_{P1,C}, r'_{P1,C}, r_{P2,C}$ and $r'_{P2,C}$ morphisms, and the dotted arrows represent their compositions. We have morphisms both from the

ports to the connection and from the connection to the ports. Therefore, we also have morphisms between ports that participate in the same connection because of composition. In the same way, assume we have three ports, P1, P2, and P3, and two connections, C1 between P1 and P2 and C2 between P2 and P3. We will have morphisms between P1 and P3 because of composition, indicating an indirect connection between these ports.

The third point of the definition does translate the belonging relation in S2ML models. We define a morphism between a block, port, or connection and a block that contains it. In the case of blocks, this morphism resembles the identity; it maps every object of the port to its parent or ancestor. This means that everything in the block is also contained in its ancestors. For connections, this relation will absorb the content of the connection to itself since the connection conceptually only refers to the ports rather than being a container that they are a part of. This also helps to disambiguate the difference between a port being in a connection that belongs to a different block and a direct belonging morphism from a port to a block. The last property of that point expresses that a model element has a unique parent, either the block itself or a block within it. All other blocks that contain it are its ancestors and will contain this parent.

An example of the category depicting the system block from Figure 6 is depicted in Figure 7. This category indeed contains everything that is contained in the system block. In this case, this includes the A and B blocks, the P1 and P2 ports, and the C connection.

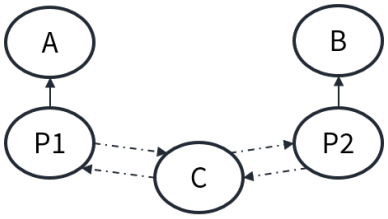


Figure 7. Diagram of the system block from Figure 6.

We have defined all elements of the models: blocks, ports, and connections; we can now use these concepts to define the model itself. The model contains the blocks, ports, and connections and is structured as its root (main) block, with the difference that it also contains the root block and morphisms between elements of the root block and the root block.

Definition 12. *S2ML Model*
An S2ML model is a category M such that:

- *Ob(M) is a finite set of blocks, ports, and connections;*
- *There exists a block R called the root of M such that* $Ob(R) = Ob(M) \setminus R$*, and for each* $x, y \in Ob(R)$, $Hom_M(x, y) = Hom_R(x, y)$*;*
- *The morphisms described between each object and a block in the definition of a block also hold between the objects of M and the block R;*
- *There are no other morphisms in the model than those described here and the morphisms derived from category theory axioms, i.e., identities and compositions.*

The root object in the model does contain all other objects; there will always be a belonging morphism from an object to the root object.

We can draw diagrams over S2ML models like diagrams are drawn over any other categories. In order to make these diagrams easy to read and faithful to the system, we only show the direct belonging morphisms and morphisms between ports and connections. Any morphism that is a composition or identity still exists in the category, but we do not show them in the diagrams.

We have defined morphisms between the objects of the models. Although these morphisms can always be composed, they, in a way, are typed, as their properties can be

used to explain the relationship they express between the objects. The belonging morphisms express that an object is contained in a block.

Definition 13. *Belonging Morphism, Direct Belonging Morphism, Ancestor, Parent*
Let M be a model, let $A, B \in Ob(M)$, and let $f : A \longrightarrow B$ a morphism in M.
We call f a belonging morphism if and only if B is a block, and:

- *A is a block, and f is the identity over this block;*
- *A is a port, and f maps $p \in A$ to $A \in Ob(B)$;*
- *A is a connection, and f maps any $x \in A$ to $A \in Ob(B)$.*

If f is a belonging morphism, we say that B is an ancestor of A.
We call f a direct belonging morphism if it is a belonging morphism and there is no block $B' \in Ob(B)$ such that $A \in Ob(B')$.
If f is a direct belonging morphism, we say that B is the parent of A.

We call a morphism a belonging morphism if its target is a block, and it either is similar to the identity when its source is a block or a port, or it maps everything to the connection itself if the source is a connection. Examples of direct belonging morphisms in the diagram for a model can be found in Figure 8; the plain black arrows are the direct belongings. Paths in the model formed by the composition of one or more direct belonging morphisms are the belonging morphisms; they correspond to the relations between a model element and its ancestors.

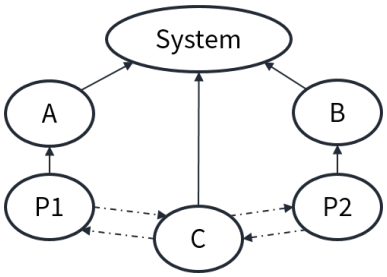


Figure 8. Diagram of the system model from Figure 6.

The other significant kind of morphism is reference morphisms. The reference morphisms show the relation between a port and a connection that the port participates in.

Definition 14. *Reference morphism*
Let M be a model, let $P, C \in Ob(M)$ such that P is a port and C a connection, and let $f : A \longrightarrow B$ a morphism in M.
We call f a reference morphism if and only if f maps $p \in P$ to $P \in Ob(C)$.

Examples of reference morphisms can be found in Figure 8; they are the dotted arrows from P1 and P2 to C. The reference morphisms represent the encapsulation of the ports in the connections, as they are elements of the connection.

We call injections a certain kind of functors between models. These functors represent the fact that a model is part of another model.

Definition 15. *Injection*
Let A, B be two S2ML models, and let $F : A \longrightarrow B$ be a functor. We call F an injection if and only if F is injective on objects and faithful, and F lets an object's and morphism's properties remain unchanged, i.e., ports, blocks, and connections are, respectively, mapped to ports, blocks, and connections, and reference and belonging morphisms are, respectively, mapped to reference and belonging morphisms.

A graphical illustration of injection between two models can be found in Figure 9. Injections are functors that associate each object and morphism of its source with a unique object or morphism of the target. The existence of an injection between two models means that the structure of the first model is a part of the structure of the second one.

Interesting properties of the objects can also be defined, which we use in the demonstrations of theorems.

The order of an object is the number of depth levels in the system breakdown separating this object from the root object of the model. We also call the order of the model the number of depth levels of system breakdown of the model.

Definition 16. *Order of a model object, order of a model*
Let M be a model, and let $X \in Ob(M)$.
We call the order of X its distance to the root object, i.e., the number of direct belonging morphisms that have to be composed to obtain a morphism, $f : X \longrightarrow R$.
We call the order of M the upper bound of the orders of its objects.

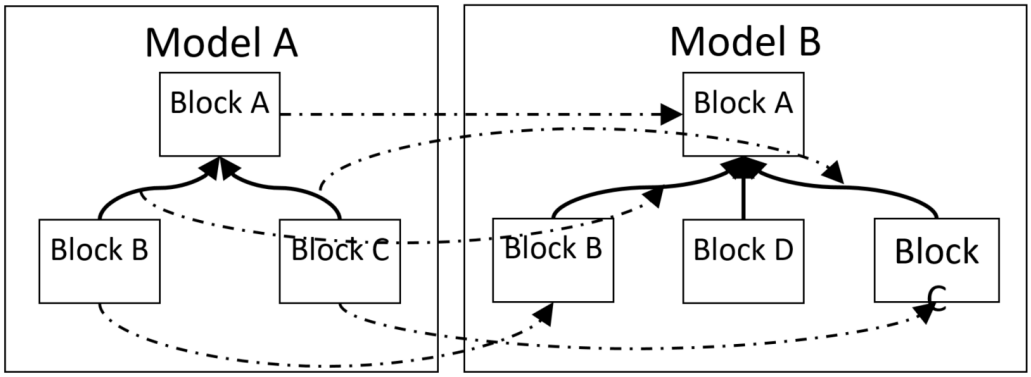


Figure 9. An injection between two models.

The root of the model is an object of order zero, and objects with a direct belonging towards the root are of the first order. As an example, in Figure 8, the object of order zero is the block System, objects of the first order are the blocks A and B and the connection C, and objects of the second order are the ports P1 and P2. The model itself is of the second order since the ports have the maximal order.

We also define elementary blocks, which are blocks that do not contain other blocks. They are useful in inductive proofs, as they are the smaller blocks in the models, in the sense that they may be contained in other blocks that are subsequently "bigger" than them.

Definition 17. *Elementary block*
We call an elementary block any block that contains no other block.

The S2ML models with the injections are a category that we call $S2ML + Cat$. $S2ML + Cat$ is the frame within which we will study the characteristics of the models.

Definition 18. $S2ML + Cat$
We define $S2ML + Cat$ *as being the category composed of:*

- $Ob(S2ML + Cat)$*, the set of all S2ML models. We call them the objects of* $S2ML + Cat$*;*
- *For each* $X, Y \in Ob(S2ML + Cat)$*, Hom(X, Y) is the set of the injections from X to Y.*

Proposition 1. *S2ML+Cat is a category.*

Proof.

- Identity:
  Let $X \in S2ML + Cat$.

The morphisms of S2ML+Cat are the injections between S2ML models.
Due to the identity being trivially injective on objects and morphisms, it is an injection.
Therefore, $Id_X \in Hom_{S2ML+Cat}(X, X)$.

- Composition:
  Let $X, Y, Z \in S2ML + Cat$, let $F \in Hom_{S2ML+Cat}(X, Y)$ and let $G \in Hom_{S2ML+Cat}(Y, Z)$.
  $G \circ F$ is an injection, because each of its components (over objects and homsets) is a composition of injective applications and thus an injective application.
  Therefore $H = G \circ F \in Hom_{S2ML+Cat}(X, Z)$.
- Associativity: Injections are functors between categories; therefore, they are associative.

S2ML+Cat presents identity morphisms. Its morphisms can be composed, and their composition is associative. Therefore, S2ML + Cat is a category. □

We have defined a category that allows for the mathematical representation of S2ML models. Under the hypothesis that any structural model can be represented with blocks, ports, and connections, this mathematical representation is readily applicable to any such model. This is the basis for our mathematical framework.

The definition we have given here allows a unique categorical representation of any S2ML that was defined as a quintuplet.

Theorem 1. *Ob(S2ML + Cat) is isomorphic to the set of the quintuples from [5], up to the choice of symbols for blocks.*

Proof. Let S2ML be the set of all quintuplets from [5].

$$\text{Let } \begin{cases} F: & S2ML \longrightarrow Ob(S2ML + Cat) \\ & X =< P, C, B, \alpha, r > \rightarrow X_{cat} \end{cases}$$

where X_{cat} is such that:

- Ob(X) contains:
  - For each $p \in P$, the port $P' = \{p\}$;
  - For each $c = \{P_1, \ldots, P_n\} \in C$, the connection c' such as $c' = \{P'_1, \ldots, P'_n\}$ with $P'_k = \{P_k\}$;
  - For each block $b \in B$, the block b' that contains all blocks, ports, and connections that are under b for the α relation;
- The morphisms in X are trivially obtained from Definition 12.

The block b' that we define is unique because it can be built by induction by building first the blocks that contain no blocks (the connection and ports being trivially unique, these blocks also are), then blocks of the next level of abstraction for the α relation, until we have built all blocks that are contained in b'.

F allows us to convert any model from S2ML to a model in $Ob(S2ML + Cat)$.

$$\text{Let } \begin{cases} G: & Ob(S2ML + Cat) \longrightarrow S2ML \\ & X \rightarrow X_{quint} \end{cases}$$

where X_{quint} is the set of models $< P, C, B, \alpha, r >$ from S2ML, such that:

- P = { $p \in P_{Cat} \mid P_{Cat} \in Ob(X)$ is a port };
- B = { b' | b' is a symbol associated to $b \in Ob(X)$, which is a block };
- C = { c' = {$p \in P_cat$ for $P_cat \in c \mid c \in Ob(X)$ is a connection};
- α = { $(x, y) \in Ob(X)^2 \mid Hom_X(x, y)$ contains a direct belonging morphism };
- r is the root of X.

The image of X_{quint} by F is X, by construction. Therefore, we have a way to map each element of S2ML to an element of $Ob(S2ML + Cat)$. We also have a way to map each element of $Ob(S2ML + Cat)$ to a set of elements of S2ML. Finally, both these mappings are each-other inverse up to the choice of symbols for the blocks. □

A pleasing property is that when two models can be injected into one another, they have the same structure; this can be expressed as follows.

Theorem 2. *Let $A, B \in Ob(S2ML + Cat)$, such that there exists injections $F : A \longrightarrow B$ and $G : B \longrightarrow A$.*

Then there exists an injection $G' : B \longrightarrow A$ such that $G \circ F = Id_A$ and $F \circ G = Id_B$.

Remark 1. *Theorem 2 is a generalisation of the Cantor–Schröder–Bernstein theorem to S2ML models.*

Proof. Let $M_1, M_2 \in Ob(S2ML + Cat)$ and $F : M_1 \longrightarrow M_2, G : M_2 \longrightarrow M_1$ be two injections.

F and G are injective on objects; therefore, we know that $card(Ob(M_1)) = card(Ob(M_2))$. Let R be the root object of M_1 and R' the root object of M_2.

The belonging morphism between the object and its parent will also be mapped to a belonging morphism. Therefore, as F is injective, any object of M_1 which has a parent is mapped to an object of M_2, which also has a parent.

Therefore all objects of M_2 that have a parent are the images of objects of M_1 that have a parent, since $card(Ob(M_1)) = card(Ob(M_2))$.

Thus, since F is injective on objects, $R' = F(R)$.

Let $X \in Ob(M1)$ be of order k.

By definition, there exists $X_1, \ldots, X_{k-1} \in Ob(M1)$ and $r_1 : X \longrightarrow X_1, \ldots, r_k : X_{k-1} \longrightarrow R$ such that the r_i are direct belonging morphisms.

Therefore, there exists $X', \ldots, X'_{k-1} = F(X), \ldots, F(X_{k-1})$ and $r'_1 : X' \longrightarrow X'_1, \ldots, r'_{k-1} : X'_{k-1} \longrightarrow R'$, where the r'_i are belonging morphisms, which are the images by F of the r_i.

This means that $Order(X') \geq Order(X)$.

If we take the objects of maximum order in M_1 and M_2, we obtain $Order(M_1) \geq Order(M_2)$ inversely.

Therefore, $Order(M_1) = Order(M_2)$.

Let $k = Order(M_1) = Order(M_2)$. Let n be the number of objects of order k in M_1; then, M_2 has at least n objects of order k, since $Order(F(X)) \geq Order(X)$.

Since the same holds for G, we have that M_2 also has exactly n objects of order k.

Assume this property (M_1 and M_2 have the same number of objects of a specific order and objects are mapped to objects of the same order) to be proven for any order above $i \in [|0; k|]$.

Let $X \in Ob(M_1)$ be an object of order i; then, $Order(F(X)) \geq Order(X)$.

Any object of M_2 of an order superior to i is already the image of an object of M_1 of an order superior to i.

Therefore, $Order(F(X)) = Order(X) = i$.

We know that the property holds for objects of order k. For $i \in [|0; k|]$, if it holds for any order of $[\,|\, i + 1, k \,|\,]$, it holds for order i; thus, by complete induction, it holds for any $i \in [|0; k|]$.

Let C be a connection of M_1.

Then $card(F(C)) \geq card(C)$; indeed, if $C = \{P_1, \ldots, P_n\}$ and $F(C) = \{P'_1, \ldots, P'_m\}$, then we have, for each $i \in [|1, n|]$ a l such that $F(P_i) = P'_l$, because F preserves the reference morphisms, thus $m \geq n$.

This means that M_2 has more connections than M_1 for a certain order.

In the same way as for objects' order, we obtain that the image of a connection by F and G is a connection with the same number of ports.

We can now prove the theorem. Let us show by complete induction that we can build an inverse of F.

Let n be the order of M_1 and M_2.

For $n = 0$, M_1 and M_2 only contain their root objects, which are elementary blocks that have no port or connection.

Therefore, F and G are entirely defined by $F(R) = R'$ and $G(R') = R$; they are trivially inverse of eachother.

Let $n > 0$, and assume that the property holds for any $k < n$.

Then:

- F maps R to R' and vice versa.
- F maps each block $B \in Ob(M_1)$ of order 1 to a block $B' \in Ob(M_2)$ of order 1. The component of F on B and its descendants corresponds to an injection F_B between the models (of order $< n$) that have B and B′ as their root objects. Thus, we can associate an inverse G'_B to this injection.
- M_1 and M_2 have the same numbers of ports of order 1 and connections of order 1.
  - Let $C \in Ob(M_1)$ be a connection of order 1. We know that F(C) has the same number of ports as C and that the images of the ports of C are the ports of F(C). Therefore, we can define an inverse to the component of F on C, and its ports, and this inverse is compatible with the ones defined for blocks of order 1.
  - We can associate two by two the remaining ports. When we add the compositions to these components, we obtain a functor $G' : M_2 \longrightarrow M_1$, which is the inverse of F.

Therefore, we have shown Theorem 2 by strong induction. □

A direct corollary of Theorem 2 is the following one.

Corollary 1. *Let $A, B \in Ob(S2ML + Cat)$, such that there exists injections $F : A \longrightarrow B$ and $G : B \longrightarrow A$.*
Then, F is an equivalence of categories.

We can therefore give a simple definition of the equivalence of S2ML models.

Definition 19. *Equivalence of S2ML models*
Let $A, B \in S2ML + Cat$.
We say that A and B are equivalent if there exists for $F : A \longrightarrow B$ and $G : B \longrightarrow A$ two injections.

Theorem 3. *The equivalence of S2ML models is an equivalence relation over Ob(S2ML+Cat).*

Proof. We need to show that this relation is reflexive, symmetric, and transitive.

- Reflexivity:
  Let A be a S2ML model.
  A is equivalent to A because the identity over A is an injection. Therefore, $F = G = Id_A$ and α composed of for each x, and $\alpha_x : F(x) \longrightarrow G(x)$ with $\alpha = id_x$ suits.
- Symmetry:
  A is equivalent to B, so the injections $F : A \longrightarrow B$ and $G : B \longrightarrow A$ exist. Therefore, we have two injections between B and A, and as such, B is equivalent to A.
  The relation is symmetric.
- Transitivity:
  Let A, B, C be three S2ML, with A equivalent to B and B equivalent to C.
  Let $F_{A,B}$ and $G_{B,A}$, be the equivalence injections for A,B and $F_{B,C}$, $G_{C,B}$ be the equivalence injections for B,C.
  Then $F : A \longrightarrow C = F_{B,C} \circ F_{A,B}$ and $G : C \longrightarrow A = G_{B,A} \circ G_{C,B}$ are two injections between A and C. Therefore, A and C are equivalent.

□

Another property of injection is that it, in a way, represents the fact that one model is bigger than another since it is a part of it.

Theorem 4. *The injection defines a partial order relation over the set of S2ML models, given equivalences between models.*

Proof.

- Reflexivity :
  Let X be a S2ML model, and let Id_x be the identity over X, i.e., the image of $x \in Ob(X)$ is x and for $g \in Hom_X(x, y)$ with $x, y \in Ob(X)$ the image of g is g. Then Id_X is trivially an injection. Therefore, X is injected into X.
- Antisymmetry:
  Assume X and Y are two S2ML models such that X is injected in Y and Y is injected in X, with $F : X \longrightarrow Y$ and $G : Y \longrightarrow X$ such injections.
  Then, we have two injections between X and Y, and therefore, X and Y are equivalent.
- Transitivity
  Let X, Y, and Z be S2ML models.
  Let $F : X \longrightarrow Y$ and $G : Y \longrightarrow Z$ be injections.
  Let $H : X \longrightarrow Z$ be the composition of F and G on objects and morphisms, i.e., composed of:
  $\begin{cases} H_{Ob} : & Ob(X) \longrightarrow Ob(Z) \\ & x \rightarrow G \circ F(x) \end{cases}$
  and for each $x, y \in Ob(X)$,
  $\begin{cases} H_{x,y} : & Hom_X(x, y) \longrightarrow Hom_Z(H_{Ob}(x), H_{Ob}(y)) \\ & f \rightarrow G_{F_{Ob}(x),F_{Ob}(y)} \circ F_{x,y}(x) \end{cases}$
  Because each of the applications defined here are compositions of injective applications, they are injections. Therefore, H is an injection, and thus, X is injected in Y.

Therefore, the existence of these injections between S2ML models defines a partial order relation. □

3.2. Consistency Relation

Now that we have explained the category in which we are working, we can define a consistency relation between two S2ML models.

Definition 20. *Binary consistency relation*

Let $\sim$ be a binary relation over $Ob(S2ML + Cat)$. For $A, B \in Ob(S2ML + Cat)$, we note $A \sim B$ if $(A, B) \in \sim$, i.e., if A is in relation with B.

$\sim$ is a Binary consistency relation if and only if for $A, B \in Ob(S2ML + Cat)$, there exist A', B' such that these injections exist:

- $f_A : A' \longrightarrow A$;
- $f_B : B' \longrightarrow B$;
- $F : A' \longrightarrow B'$;
- $G : B' \longrightarrow A'$.

F and G respect the following properties:

- $F \circ G = Id_{B'}$;
- $G \circ F = Id_{A'}$.

This definition means that the two models are in a consistency relation if the structure shown in Figure 10 exists with *fa*, *fb*, F, and G being injections. All the morphisms are injections and the compositions of F and G are the identities of A' and B'. Note that Theorem 2 implies that if this structure exists without the compositions of F and G being the identities, then we can find G' such that F and G' compose into the identities.

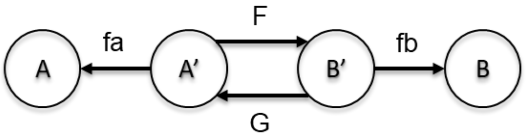


Figure 10. The structure of a binary consistency relation.

Such a relation is too general to describe a relevant interpretation of consistency between two models. To illustrate this idea, we can create a binary consistency relation such that any couple of models would be in the relation:

Proposition 2. *Let $\sim_{all}$ be the binary relation over $Ob(S2ML + Cat)$ with no other restriction than being a binary consistency relation.*
Let $A, B \in Ob(S2ML + Cat)$.
Then $A \sim_{all} B$.

Proof. Let $A, B \in S2ML + Cat$.

We have $A \sim_{all} B$ if there exists A', B' in $Ob(S2ML + Cat)$ such that there exist injections $f_A : A' \longrightarrow A$, $f_B : B' \longrightarrow B$, $F : A' \longrightarrow B'$, and $G : B' \longrightarrow A'$ such that $F \circ G = Id_{B'}$ and $G \circ F = Id_{A'}$.

Because A and B are S2ML models, there exists unique blocks $r_A \in Ob(A)$ and $r_B \in B$ such as there are no morphisms in A (resp B) with domain r_A (resp r_B).

Let A' be the S2ML model with only one object r_A and no morphisms and B' be the S2ML model with only one object r_B and no morphisms.

$\begin{cases} f_A : & A' \longrightarrow A \\ & r_A \rightarrow r_A \end{cases}$ is trivially an injection (we only present the injections through their mappings over $Ob(X)$ here since the models have no morphisms).

We define f_B using the same construction.

Let $\begin{cases} F : & A' \longrightarrow B' \\ & r_A \rightarrow r_B \end{cases}$ and $\begin{cases} G : & B' \longrightarrow A' \\ & r_B \rightarrow r_A \end{cases}$.

F and G are injections, and $F \circ G = \begin{cases} B' \longrightarrow B' \\ r_B \rightarrow r_B \end{cases} = Id_{B'}$ and similarly $G \circ F = Id_{A'}$.

Therefore $A \sim_{all} B$. □

Our definition of a binary consistency relation describes the existence of a similarity between A and B. This similarity is the shared structure of A' and B'.

Because the definition is general, the fact that A and B are models induces a similarity: the existence of the `main` element of the model, namely r_A and r_B, unique elements of the models with no parents. This results in any couple of models being in relation.

This means that not all binary consistency relations are interesting for model comparison.

The main idea behind this definition is that we can complete it with restrictions to assert consistency over specific properties of the models. As an example, we define the dictionary consistency relation between two models:

Definition 21. *Dictionary consistency relation*

We call adDictionary consistency relation a binary consistency relation $\sim_{dic}$ such that for $A, B \in Ob(S2ML + Cat)$, $A \sim_{dic} B$ implies:

A' and B' are two models injected in A_{comp} and B_{comp}, where A_{comp} and B_{comp} are two models with the same sets of objects as A and B but no morphisms.

Such a relation consists of matching objects from one model to corresponding objects on the other model, except for some elements that are kept unmatched. Therefore, this method can be used to detect elements from one model that have no equivalent in the other model; they are the objects of A_{comp} that are not in A' (idem on the B side).

As before, it could be argued that any couple of models is in relation using this definition, as we could also take the models with only the main elements. For example, this could be solved by having a unique "name" attribute in our objects and considering that objects linked by the dictionary should carry the same name. However, that would be hiding the real problem since there is no guarantee that elements are named the same way in the original models. The reality is that such a dictionary would have to be created by the engineers, as the model elements carry a meaning that the computer cannot interpret. In general, the way to express a "good" consistency relation is to have a condition of

maximality on the A' and B' models. One way to achieve this maximality is through the use of a pullback. We discuss these ideas in Section 5.1.

While defining the dictionary relation, we used the A_{comp} and B_{comp} models as an intermediary step between A/B and A'/B'. We use these models to eliminate the elements that we do not intend to compare in our binary consistency relation between A and B. In this way, the objects from A_{comp} and B_{comp} that are not in A' and B' will be the differences detected by the binary consistency relation.

Therefore, one possible structure to build a binary consistency relation over an existing synchronization methodology is the one found in Figure 11. $S2ML + Cat$ could also be replaced by an equivalent construction over another common formalism.

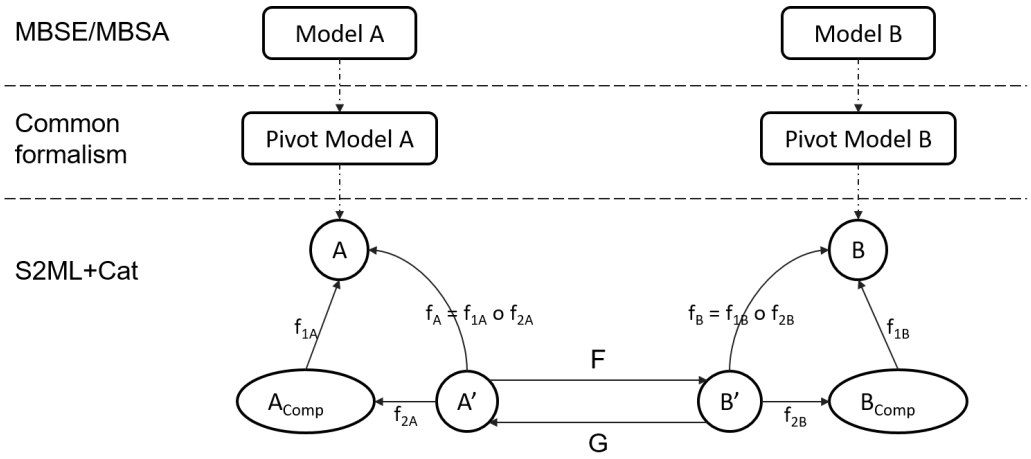


Figure 11. Structure for binary consistency relations.

4. Application Example with SmartSync

4.1. Study Case

The study uses a fixed-wing drone as its applicative example. This drone is inspired by the Zipline company blood delivery drone [31]. The specific use of this drone is the delivery of blood packages to different hospitals and clinics across a large area from a blood storage center. This drone is a fixed-wing drone, meaning a drone similar to a plane. Its powertrain has two coaxial motors, each linked to a propeller. The drone also presents redundancy in its ailerons, used to control direction. A control processing unit is placed in the removable battery of the drone and is given the flight plan before each flight. A QR code allows for identifying the blood package once installed in the drone cargo compartment until it is parachuted down to the delivery site. The drone takes off by being catapulted; then, it flies to the drop area. Cargo can be parachuted up to an 80 km radius, with a maximum time to objective of 45 min. The drone then comes back to the storage site. It lands by being caught by a recovery system, a sling between two 10-m high towers that attaches to a hook on the drone. An illustration of the drone's external architecture and flight scenario can be found in Figure 12.

We consider a SysML architecture model as the specification of the system. From this architecture, we derived three models for different intents:

- Safety assessment: An AltaRica 3.0 model represents the model for safety assessment.
- Scenario: A SCOLA (SCenario Oriented LAnguage) model represents the functional scenarios.
- Multi-physics: A Modelica model represents the multi-physics behavior of the drone's power electronics and aerodynamics.

We asserted consistency over this design using SmartSync to synchronize these three models with the architecture model.

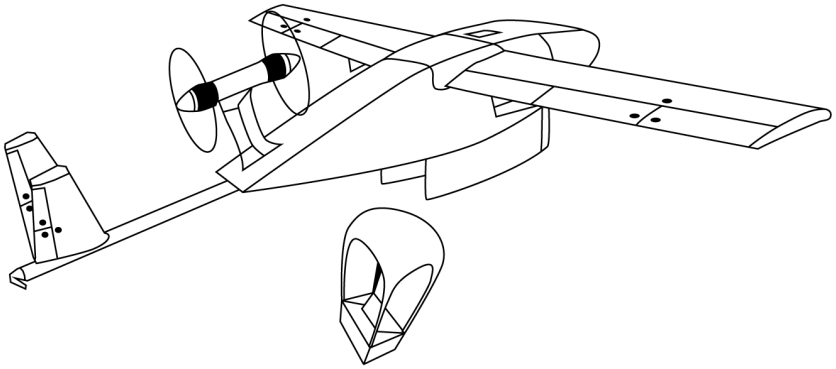


Figure 12. Illustration of the Zipline Flyer drone.

4.2. Applying SmartSync to the Study Case

This section applies the mathematical framework to a study case through the SmartSync approach. We show compatibility potential between SmartSync and our framework, but the mathematical definition of SmartSync within our framework is yet to come. This will be discussed in Section 5.

4.2.1. The Use of SmartSync

The SmartSync approach was operated through the following steps:

- We translated the models to S2ML.
- The SmartSync tool associated the main elements of both models and asked the user to align the children elements of the mains.
- When given the associated children of the main, the SmartSync tool iterated the previous steps for these elements, asking the user to align their children.
- Such iterations were done until the tool had entirely explored the models.

If an element has no equivalent in the other model, the user can give it the attribute `forget` if he can justify why it is not an inconsistency.

The SmartSync tool takes two S2ML compiled models (in the XML format) as an input and outputs a CSV comparison file. The user aligns the elements from both models in the CSV file, then re-iterates the SmartSync comparison with the two S2ML compiled models and the CSV file as an additional input. The SmartSync tool is then able to identify the children elements and provides a new CSV comparison file to the user for the next iteration.

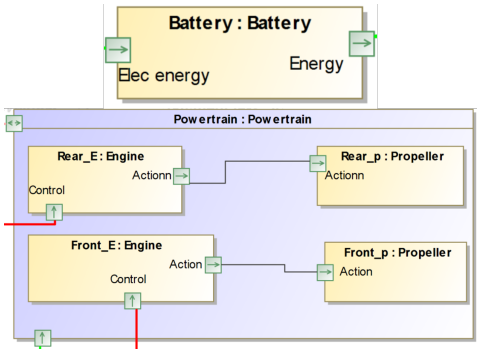
Translation to S2ML

Before conducting each comparison, we first had to abstract the SysML model. The abstraction was made towards the S2ML language, which is used in the SmartSync methodology.

We translated the information from both the BDD and the IBD to conduce the abstraction. The S2ML model followed the hierarchy from the BDD and the structure of the IBD. We represented SysML ‘‘`partProperties`’’, i.e., components of the system in the S2ML model, with blocks. We also used S2ML ports to represent SysML ports and S2ML connections to represent SysML connections.

An example of the transformation over part of the SysML model (namely the battery and drivetrain) is shown in Figure 13.

```
block DRONE
block Battery
port Elec_energy;
port Energy;
end
block PowerTrain
block Rear_E
port Action;
port control;
end
block Rear_p
port Action;
end
block Front_E
port Action;
port control;
end
block Front_p
port Action;
end
port propulsive_effort;
port Energy;
connection [Front_motor.Action,
     Front_propeller.Action];
connection [Rear_motor.Action,
     Rear_propeller.Action];
end
```



(**a**) (**b**)

Figure 13. A part of the SysML model (**b**) and its translation in the S2ML model (**a**).

Once we translated the SysML model, we operated the comparison with the AltaRica model and the Modelica model. This was done by translating those models, then operating the SmartSync comparison.

MBSE/MBSA

To compare the SysML model and the AltaRica 3.0 model, we first translated the AltaRica 3.0 model to S2ML. This process was not complicated because the structural part of the AltaRica 3.0 languages is S2ML. Blocks were translated to blocks, variables to ports, and assertions to connections. An example of this translation can be found in Figure 14. In this example, it can be seen that some attributes giving more information on the AltaRica model were included in the S2ML model (e.g., "`type = ''Boolean''`"); these attributes are not currently used in the SmartSync comparison.

```
block Avionics
block Comms_module
port is_working (type = ''Boolean'', kind =
     ''init'', value = ''true'');
port lambda (type = ''Real'', value = 10e-6);
port failure (type = ''event'', delay =
     ''exponential(lambda)'');
connection [failure, is_working, is_working]
     (type = ''connection'', expr = ''true ->
     false'');
port in4G (type = ''Boolean'', kind =
     ''reset'', value = true);
port inCalculator (type = ''Boolean'', kind =
     ''reset'', value = ''false'');
port outData (type = ''Boolean'', kind =
     ''reset'', value = true);
port outCalculator (type = ''Boolean'', kind =
     ''reset'', value = true);
connection [outData, is_working, in4G](type =
     ''assertion'');
connection [outCalculateur, is_working,
     in4G](type = ''assertion'');
end
```

```
block Avionics
block Comms_module
extends NRComponent;
Boolean in4G (reset = true);
Boolean inCalculator (reset = false);
Boolean outData (reset = false);
Boolean outCalculator (reset = true);
assertion
outData := if is_working then in4G and
     inCalculator else false;
outCalculator := if is_working then in4G else
     false;
end
```

(**a**) (**b**)

Figure 14. A part of the AltaRica 3.0 model (**b**) and its translation in the S2ML model (**a**).

Once the translation was done, the models were given to the SmartSync tool as inputs. The tool then took an element of each model that it knew to correspond to one another. This was the main element of the models for the first step, i.e., the blocks representing the whole system. The tool then asked the user to align children of these two elements from both models. The user can align elements or give them the `forget` attribute—meaning the element is not supposed to have a counterpart in the other model—or do neither. The tool

then iterated on each couple of aligned elements until all elements were either aligned or forgotten. When no element could be aligned anymore, the remaining elements with no counterpart were considered inconsistent.

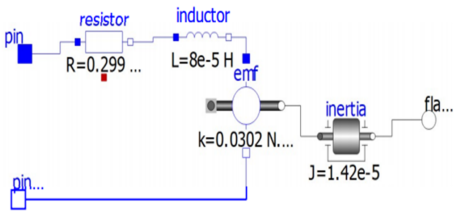
In this case, the comparison between the architecture and AltaRica 3.0 models was done through 4 comparison steps. It shows that the MBSA model lacks some elements of the system. The missing elements are the redundant ailerons and the rudder airfoil. This is due to a communication issue when the safety model was designed based on an outdated version of the architecture model. Correction of these inconsistencies was conducted by adding the missing elements to the AltaRica 3.0 model.

Note that all safety artifacts were ignored in this comparison, such as state variables, events, etc. The comparison of the models also has shown that interactions with the environment such as gravity, 4G network, etc. have not been taken into consideration in the MBSA model. We deemed this normal in this work, but it could be argued that it should be added; therefore, consistency assessment would have resulted in adding these interactions to the MBSA model. Such decisions are non-trivial and should be undertaken by the system engineer and safety analyst; this is why the consistency assessment cannot be fully automatized. Of course, after modification of the model, a new comparison should be operated to assert that the correction did not introduce new inconsistencies.

MBSE/Multi-Physics

We needed to make a second comparison to assert the global consistency of our design with the Modelica/SysML comparison. The translation from Modelica to S2ML was done following the process described in [21], with the difference that we considered Modelica's variables. Part of this translation can be found in Figure 15 with the translation of the motor class. Modelica classes were abstracted to S2ML classes, with model instances being translated to blocks. Modelica variables were translated to S2ML ports, and connect clauses were translated to connections.

```
class motor
port pin, pin_n, flange_b;
block resistor
port _in, _out;
port r (value=''0.299'');
end
block inductor
port _in, _out;
port l (value=''8e-5'');
end
block emf
port _in, _out, flange;
port const (value=''0.0302'');
end
block inertia
port _in, _out;
port J (value=''1.42e-5'');
end
connection [pin, resistor._in];
connection [resistor._out, inductor._in];
connection [inductor._out, emf._in];
connection [emf._out, pin_n];
connection [emf.flange, inertia._in];
connection [inertia._out, flange_b];
end
```



(**a**) (**b**)

Figure 15. A part of the Modelica model (**b**) and its translation in the S2ML model (**a**).

Once this translation was conducted, we compared the models with the SmartSync tool. A first iteration showed that the structure of the models is very different. For ease of comparison, we added `drivetrain` and `cell` blocks to the models, respectively containing the motors and propellers and the fuselage and wings. We then translated again and compared the models.

After four comparison steps, we had three conclusions:

- Most components of the drone are missing in the Modelica model. This can be explained by the fact that we are here only interested in the propulsion and aerodynamics of the drone; therefore, components related to other functions are not represented.

- In some specific cases, the Modelica representation is more detailed than the architecture model. This is the case of the motors since the SysML model only considers them through a black box view, whereas the Modelica view shows the inner parts of the motor for calculations.
- There is an inconsistency between the models. This inconsistency is the presence of the fuselage within the Modelica model, which is modeled for aerodynamics purposes, but unrepresented in the architecture model. This inconsistency is corrected by adding the fuselage to the architecture model.

4.2.2. Categorical Point of View

In this subsection, we analyze the comparisons made in Section 4.2.1 in the context of the mathematical framework we defined in Section 3.

Visualisation of S2ML + Cat Models with NetworkX

In order to achieve this goal, we used oriented graphs that give a visual representation of the categorical S2ML models.

We created this representation at each step during the comparison to visualize which parts of the model the comparison traversed. Because SmartSync makes the comparison in a way that resembles a breadth-first search, this corresponds to taking the model with only its first level of abstraction (the main and its children) and then adding a level of abstraction for each iteration.

To obtain the graph representations, we wrote a python script that converts the CSV files created by the SmartSync tool to directed graphs over the underlying S2ML + Cat models. This script used the NetworkX package [32] as its basis for graph classes and visualization methods. This package is a library for the study of graphs and networks. It features classes for the representation of graphs and digraphs, and methods for graph analysis and visualisation.

The graphs we show here over the S2ML + Cat models only show direct belonging relations, blocks, and ports. The absence of composed morphisms is because we wanted to minimize the number of edges to allow for good readability. The absence of connections is related to the comparison made by SmartSync, which does not consider them yet.

For each iteration, the script also computed a correspondence matrix between objects of both models, which corresponds to giving the F and G functors from Figure 11. This matrix is represented as a table. Its element $A_{n+1,m+1}$ will contain -1 if the nth element of the first model or the mth element of the second has the attribute forget, 1 if they are aligned and 0 otherwise. The first column and line of the table, respectively, contain the first and second model elements' names, meaning $A_{n+1,0}$ contains the name of the nth element of the first model, and $A_{0,m+1}$ the name of the mth element of the second one, with the elements being arbitrarily numbered.

In this aspect, we can say that for each iteration, we have constructed the A_{Comp} and B_{Comp} S2ML + Cat models for the elements that were already aligned and the F and G functors. The A' and B' S2ML + Cat models are obtained by removing the objects with the attribute `forget` and the morphisms with these objects as the source or target.

Comparison with the S2ML + Cat Models

Figure 16 shows said graphs for the comparison of the SysML and AltaRica 3.0 models of the fixed-wing drone. Blocks are represented with green vertices and ports with blue ones, and diverse abstraction levels in the models are shown by different concentric ellipses of vertices.

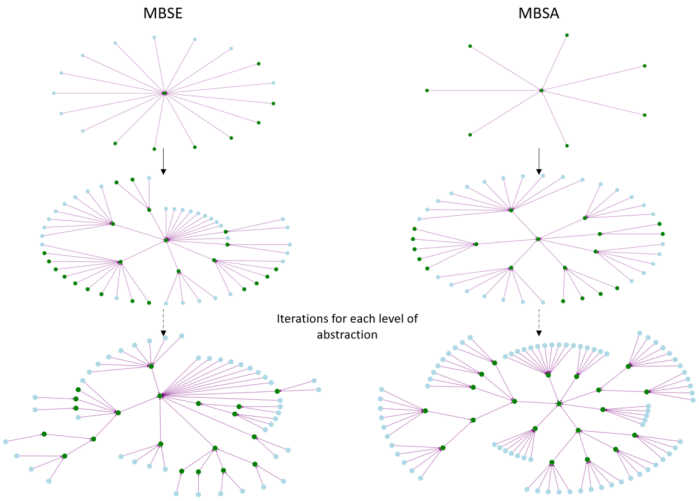


Figure 16. Diagrams for the S2ML models of the fixed-wing drone during iterations of SmartSync over the architecture and safety models.

Table 1 provides the correspondence matrix for the fixed-wing drone between the SysML and AltaRica 3.0 S2ML models. It can be observed that many ports are present in the SysML representation, whereas they are not in the AltaRica 3.0 model. This corresponds to the interfaces with the outside of the system that were discussed in Section 4.2.1.

Table 1. Correspondence table for the first iteration of the comparison between the SysML and AltaRica 3.0 models.

	ZippyFlyer	Avionics	Battery	Calculator	Cell	Inertial_measurement_unit	Power_unit	Radar
Alternative_1	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
AirAction	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.1
Battery	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
Data_in	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.1
Data_out	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.1
Ener_elec	−1.0	−1.0	−1.0	0.0	−1.0	−1.0	−1.0	−1.1
Gravity	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0
Obstacle_image	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0
Orientation	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0
Blood_bag	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0
4G_network	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0
Geolocation_signak	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0	−1.0
Avionics	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
Calculator	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
Cell	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
Inertia_measurement_unit	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
Power_unit	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
radar	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

From the final step of the comparison between the SysML and AltaRica 3.0 models, we can obtain the A' and B' S2ML + Cat models from Figure 11. These models are

shown in Figure 17. We observe that although these graphs have different names for their vertices, they are the same graphs, meaning we indeed found a common skeleton between both models. This property characterizes the way we mathematically define a binary consistency relation.

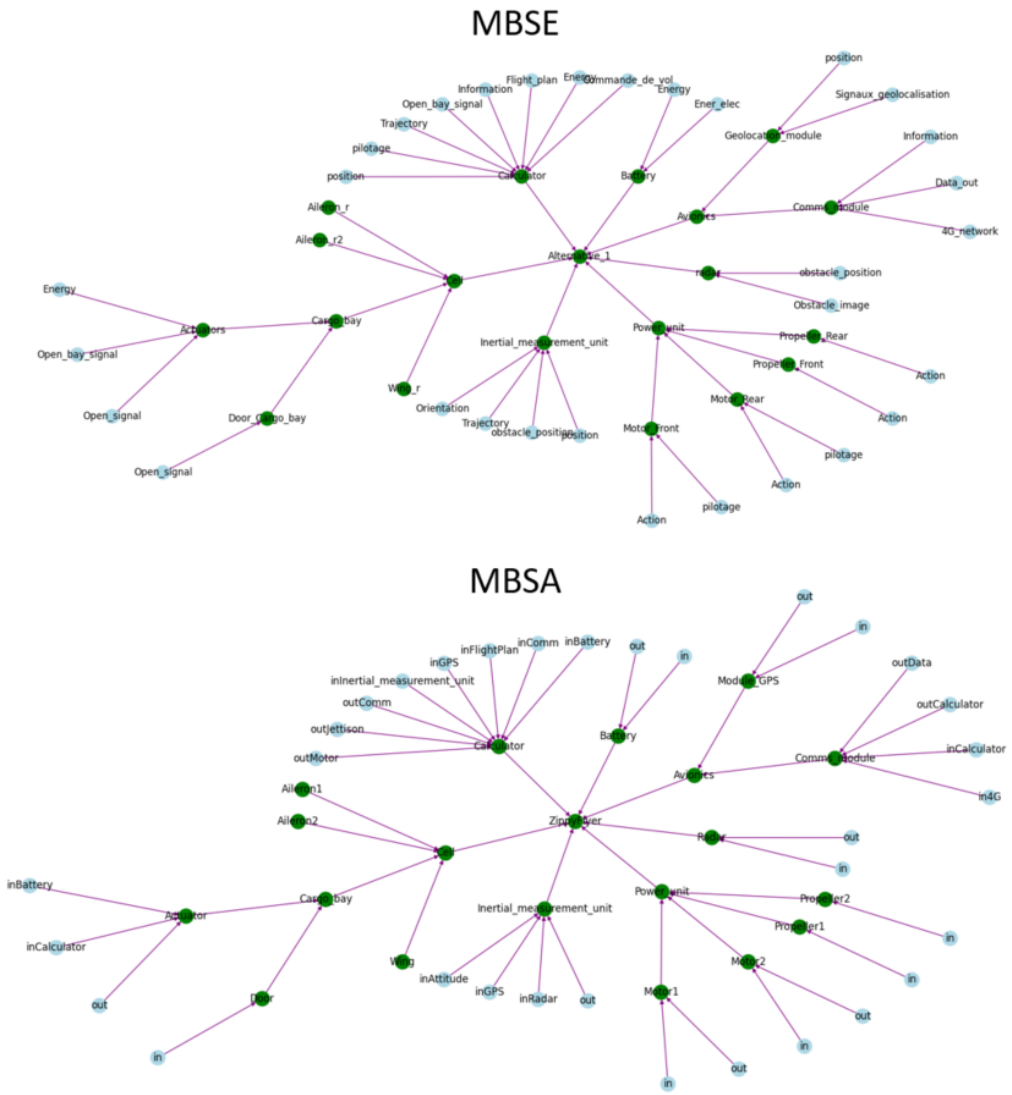


Figure 17. Diagrams for the A' and B' S2ML models of the fixed wing drone at the final comparison step.

5. Discussion

Although this mathematical framework does not provide a new synchronization methodology, it shows existing and future methodologies from a new point of view. In light of this new perspective, we have a new comprehension of some of the characteristics of these methodologies. We also show some current limitations and give ideas for improvement. This framework also allows us to formally show which exact characteristics of the models are compared through the synchronization methodology. In the case of SmartSync, we can tackle the topics of pragmatics, composition, and connections.

5.1. Pragmatics

In SmartSync, and more generally in the methodologies we presented in Section 2.2, the user is required to align the model elements manually. Although this seems like a bothersome and time-expensive step, we believe that it is almost impossible to do otherwise in

general. Even though models have syntax and semantics that allow them to be interpreted by a computer, they also carry a meaning that only makes sense to humans. This meaning is called pragmatics. Pragmatic models, such as the MBSE models, carry a meaning that is lost if model elements are replaced with abstract names, such as described in [33]. The interested reader may find more information on pragmatics in the book [34].

The SmartSync tool deals with the semantic aspect of models by creating a form of a dictionary—the CSV file. In the general definition of consistency that we have given in this paper, we do not force the consistency relation to consider pragmatics. Yet, SmartSync still deals with pragmatics, so we should explain how that makes sense in the S2ML + Cat Framework.

One way to include pragmatics in the S2ML + Cat framework is the idea that we should build the biggest common skeleton to both models that respects the user's dictionary. This can be achieved by constructing a model that represents the dictionary. Injections towards this model from both A_{comp} and B_{comp} and then building the pullback of A_{comp} and B_{comp} will allow representing the alignment of objects by the user. Therefore, as depicted in Figure 18, the A' and B' categories would be the pullback of A_{comp} and B_{comp}.

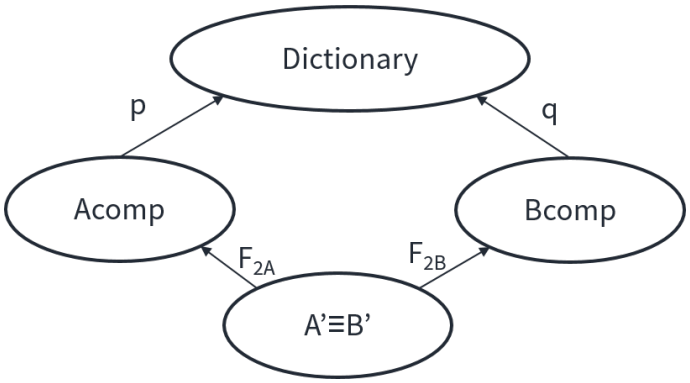


Figure 18. The pullback of A_{comp} and B_{comp} with regards to the dictionary.

We need to precisely define the dictionary if we want to operate the computation of the pullback or even prove that it does always exist. An idea for that would be to build it from A_{comp} and B_{comp} with these constraints:

- $Ob(Dictionary)$ is a set of blocks, ports and connections, such that we have $p : Ob(A_{comp}) \longrightarrow Ob(Dictionary)$, $q : Ob(B_{comp}) \longrightarrow Ob(Dictionary)$ such that for each $x \in Ob(A_{comp})$ and $y \in Ob(B_{comp})$, we have $p(x) = q(x)$ if and only if the elements have been aligned by the user;
- p and q are injective functions;
- We give the dictionary the form of a model by building the morphisms needed to have images of the morphisms of A_{comp} and B_{comp} when we enrich p and q to make them functors.

We can make it so that this category is unique by choosing the symbols for the ports judiciously, but it is not important, as any such category would suffice to represent that we aligned the objects of A_{comp} and B_{comp}.

Although we do not care for the category structure in the dictionary, we can build it. The components over the objects of the p and q functors carry important information, i.e., the alignment of model elements.

5.2. Composition

The aspect of composition in category theory is of high interest in the concepts that we define.

The morphisms in the categories of S2ML + Cat define relations between the model elements to allow us both to apprehend the direct and composed relations. In [5], the definition of S2ML does provide a composition relation that only links an element and its

parent. In S2ML + Cat, however, the belonging morphisms define the Cartesian closure of this relation. This allows us to identify all the ancestors of an object very quickly.

Even though this is not yet taken into account in the methodology that we have tested with the study case, this means that the binary consistency relation that we define is able to encompass cases where a model element is at a certain level of abstraction in one model and another level of abstraction in the second model. A simple example of such a difference is if we represent a gear motor in the system. We can, in one model, have a gear motor block that contains a motor block and a gear block, and, in the other model, directly represent the motor and gear blocks without the gear motor level.

This would result in the categories shown in Figure 19. In these diagrams, we identify the direct belongings to the black arrows and their composition to the dotted arrows. We can easily understand that the dotted arrows in the first case correspond to the black ones in the second; therefore, these models can be deemed consistent with one another.

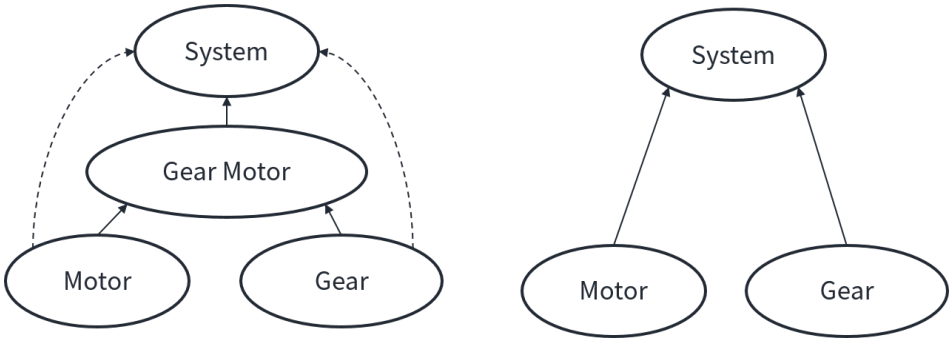


Figure 19. The categories for both possible levels of abstraction of the gear motor.

The second kind of relations that we have in the models are those between the ports and the connections they participate in. Because of the way we defined the morphisms between ports and connections, and thanks to composition, we obtain morphisms between ports that participate in the same connection, as shown in Figure 5.

We also obtain morphisms between ports that are indirectly connected, i.e., in the case where we have three ports, with two connections between two of them. Such a structure will lead to the existence of morphisms between the two ports that are not part of the same connections. This allows us to apprehend indirect connections between ports. Thus, we could consider the case in which one connection on one side corresponds to multiple ones on the other side.

An example of such a case is given in Figure 20. Some system engineers consider the representation on the left to be best practice: a connection should not cross the frontier of a block; therefore, there should be intermediary ports when crossing to a different abstraction level. In a model dedicated to computation, such as the MBSA models, doing this is a problem because it increases the amount of computation needed to execute the system. Thus, the representation on the left also makes sense to represent the system.

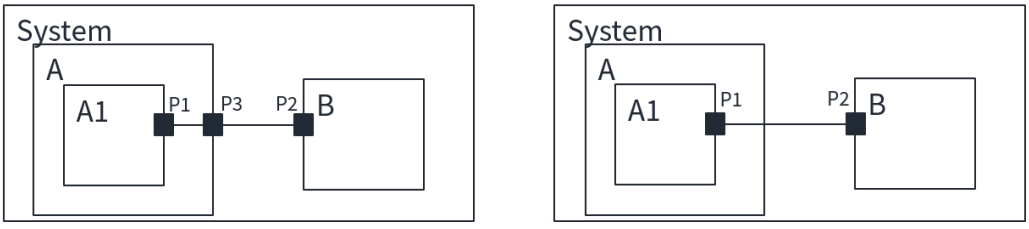


Figure 20. An example of block diagrams for a system with composed connections (**left**) that could be consistent with one connection (**right**).

The categories for the two blocks diagrams from Figure 20 can be found in Figure 21. Thanks to the composition detailed above, we obtain the dotted arrows that can be aligned in the model comparison from the categorical point of view.

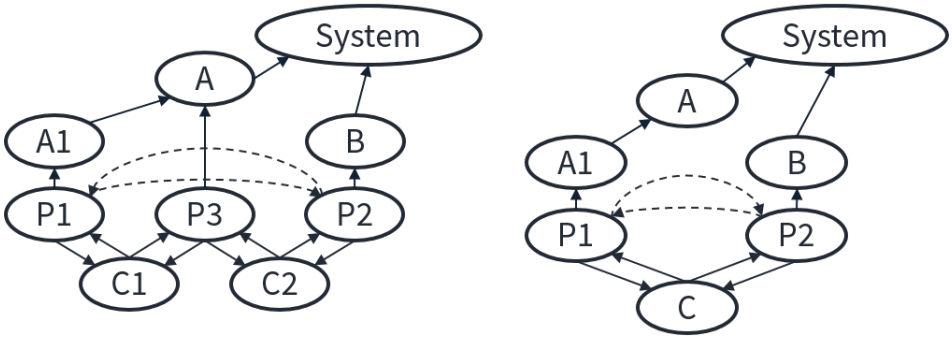


Figure 21. Diagrams of the categories for the block diagrams from Figure 20.

5.3. Connections

We discussed in Section 5.2 the fact that composition allows for better consideration of the connection. As SmartSync does not yet consider connections, we have not demonstrated this in the study case.

It is interesting to note that it should theoretically not be difficult to add support of connections. The ports are already aligned in SmartSync. We could consider the ports of a connection in model A and verify if their counterparts in model B are also connected.

5.4. Connections and Tuples

In Section 3.1, we defined the connections as sets of ports. Although this definition is consistent with the quintuplet definition of S2ML models from [5], the reader could argue that the order of ports in a connection can be significant, for an example, if there is a direction to the connection. This could be considered in the quintuplet definition by using tuples rather than sets to define the connections.

In the case of the categorical definition of S2ML models, we could consider this property by defining connections as categories, with ports being the objects of the category and morphisms from a port A to a port B if port A precedes port B in the connection.

5.5. Version Control

We believe this framework can help consider version control in the consistency assessment of the models. Indeed, when the models evolve, the vast majority of their architecture is generally left unchanged. Detecting which parts are unchanged and, thus, do not need to be reassessed for consistency could be a considerable gain in engineering time. The review process would only need to consider small sections of the models.

Consider the structure denoting a consistency relation as in the S2ML + Cat part of Figure 11 for each version number of the models. Say we compare MBSE model n with MBSA model n, and after modifications, we want to consider the consistency of MBSE model $n + 1$ with MBSA model $n + 1$. We can consider these structures as categories and then construct a functor from one to another that would provide correspondence between the unchanged elements. We hope to explore this idea in future work.

5.6. Perspective about Comparison of This Framework with Other Formal Definitions of Consistency Outside the Scope of MBSE/MBSA

We have not encountered a similar formalization of the concept of consistency between structural models. However, we believe that this work would significantly benefit from being compared with other theoretical frameworks intended to achieve similar goals. We intend in further work to define comparison criteria and operate a comparison through

one or more examples. These criteria could include, but may not be limited to, the capacity to encompass:

- Different kinds of inconsistencies;
- Internal (i.e., consistency of a model within itself) and/or external consistency (i.e., consistency between different models);
- Traceability between successive versions of the models;
- Diagnostics of the inconsistencies;
- Resolution of the inconsistencies.

6. Conclusions

This paper establishes a mathematical framework for consistency between structural models, especially system engineering and safety assessment models. In the context of synchronization methodologies, where heterogeneous models are translated to a common formalism for comparison, we give a categorical representation of the translated models. These categories are linked through injections, which are functors that specify that a category is conceptually contained in another. Using the models as objects and injections as morphisms, we define S2ML + Cat as the category of S2ML models.

In S2ML + Cat, we give a definition of what we call a consistency relation. This definition allows us to consider synchronization methodologies from a mathematical point of view.

We applied this framework to a study case through a fixed-wing drone. We show that we can mathematically interpret a synchronization methodology.

We argue that the use of category theory in this framework allows for the consideration of composition and abstraction levels in the models, and the study case results give clues for improvement in the synchronization methodology. These improvements would include considering the connections and the differences in abstraction level between the models.

Overall, this mathematical framework is not a new synchronization methodology but rather a new point of view on synchronization that allows for a better understanding of the methodologies. It also allows for mathematical proofs of the methodology's effectiveness and better apprehend leads for improvements. In future work, we expect to demonstrate that SmartSync and its encompassing of pragmatics can be modeled through a pullback in S2ML + Cat. We also hope to propose improvements to the current methodology, using the leads we presented to deal with connections and system levels. Finally, we believe that we can improve this framework to consider successive versions of the models and transfer comparison results to unchanged parts of the models.

Author Contributions: Conceptualization, J.V.; methodology, J.V.; software, J.V.; validation, J.V., M.B., F.M. and J.-Y.C.; formal analysis, J.V.; investigation, J.V.; writing—original draft preparation, J.V.; writing—review and editing, M.B., F.M. and J.-Y.C.; visualization, J.V.; supervision, M.B., F.M. and J.-Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the S2C project at IRT SystemX and its partners.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available on request.

Acknowledgments: Part of the models for the study case were developed in collaboration with Imane Bouhali during her master thesis internship at ISAE Supmeca.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
MBSE	Model-Based System Engineering
MBSA	Model-Based Safety Assessment
UML	Unified Modeling Language
SysML	System Modeling Language
DSL	Domain-Specific Language
SAML	Security Assertion Markup Language
GTS	Guarded Transition System
S2ML	System Structure Modeling Language
SCOLA	SCenario Oriented LAnguage
CSV	Comma-Separated Values file format
BDD	Block Definition Diagram
IBD	Internal Block Diagram

References

1. Gul, F.; Mir, I.; Abualigah, L.; Sumari, P.; Forestiero, A. A Consolidated Review of Path Planning and Optimization Techniques: Technical Perspectives and Future Directions. *Electronics* **2021**, *10*, 2250. [CrossRef]
2. Guychard, C.; Guerin, S.; Koudri, A.; Beugnard, A.; Dagnat, F. Conceptual interoperability through Models Federation. In Proceedings of the Semantic Information Federation Community Workshop, Miami, FL, USA, October 2013.
3. Finkelstein, A.; Gabbay, D.; Hunter, A.; Kramer, J.; Nuseibeh, B. Inconsistency handling in multi-perspective specifications. *IEEE Trans. Softw. Eng.* **1994**, *20*, 569–578._7. [CrossRef]
4. Legendre, A. Ingénierie Système et Sûreté de Fonctionnement: Méthodologie de Synchronisation des Modèles d'Architecture et d'Analyse de Risques. Ph.D. Thesis, Université Paris Saclay (COmUE), Gif-sur-Yvette, France, 2017.
5. Batteux, M.; Prosvirnova, T.; Rauzy, A. Model synchronization: A formal framework for the management of heterogeneous models. In Proceedings of the International Symposium on Model Based Safety Assessment, IMBSA 2019, Thessaloniki, Greece, 16–18 October 2019. [CrossRef]
6. Estefan, J. (NASA Jet Propulsion Laboratory, Pasadena, CA, USA). Personal communication, 2008.
7. OMG. *OMG Unified Modeling Language*; Version 2.5.1; OMG: Needham, MA, USA, 2017.
8. OMG. *OMG Systems Modeling Language (OMG SysML^{TM})*; OMG: Needham, MA, USA, 2018.
9. Wach, P.; Salado, A. The need for semantic extension of SysML to model the problem space. In Proceedings of the Systems Engineering Research (CSER), Redondo Beach, CA, USA, 20–22 March 2020._24. [CrossRef]
10. Krob, D. *CESAM: CESAMES Systems Architecting Method—A Pocket Guide*; ESAMES Association: Paris, France, 2017.
11. Mhenni, F.; Choley, J.-Y.; Penas, O.; Plateaux, R.; Hammadi, M. A SysML-based methodology for mechatronic systems architectural design. *Adv. Eng. Inform.* **2014**, *28*, 218–231. [CrossRef]
12. Batteux, M.; Prosvirnova, T.; Rauzy, A. AltaRica 3.0 in 10 Modeling Patterns. *Int. J. Crit. Comput. Based Syst. (IJCCBS)* **2017**, *9*, 133–165. [CrossRef]
13. Mhenni, F.; Choley, J.-Y.; Nguyen, N.; Frazza, C. Flight Control System Modeling with SysML to Support Validation, Qualification and Certification. *IFAC-PapersOnLine* **2016**, *49*, 453–458. [CrossRef]
14. Dugan, J.B.; Bavuso, S.J.; Boyd, M.A. Dynamic Fault-Tree Models for Fault-Tolerant Computer Systems. *IEEE Trans. Reliab.* **1992**, *41*, 363–377. [CrossRef]
15. Gudemann, M.; Ortmeier, F. A framework for qualitative and quantitative model-based safety analysis. In Proceedings of the IEEE 12th High Assurance System Engineering Symposium (HASE 2010), San Jose, CA, USA, 3–4 November 2010; pp. 132–141. [CrossRef]
16. Bouissou, M.; Bouhadana, H.; Bannelier, M.; Villatte, N. Knowledge modelling and reliability processing: Presentation of the FIGARO language and of associated tools. In Proceedings of the SAFECOMP'91—IFAC International Conference on Safety of Computer Control Systems, Trondheim, Norway, 30 October–1 November 1991; pp. 69–75. [CrossRef]
17. Batteux, M.; Prosvirnova, T.; Rauzy, A. AltaRica 3.0 Language Specification. 126p. Available online: https://www.openaltarica.fr/docs/AltaRica3.0LanguageSpecification-v1.1.pdf (accessed on 27 April 2021).
18. Rauzy, A. Guarded transition systems: A new states/events formalism for reliability studies. *Proc. Inst. Mech. Eng. Part J. Risk Reliab.* **2008**, *222*, 495–505. [CrossRef]
19. Batteux, M.; Prosvirnova, T.; Rauzy, A. From Models of Structures to Structures of Models. In Proceedings of the 4th IEEE International Symposium on Systems Engineering, ISSE 2018, Rome, Italy, 1–3 October 2018. [CrossRef]
20. Batteux, M.; Choley, J.-Y.; Mhenni, F.; Prosvirnova, T.; Rauzy, A. Synchronization of System Architecture and Safety Models: A Proof of Concept. In Proceedings of the International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 1–3 October 2019; pp. 1–8. [CrossRef]

21. Batteux, M.; Choley, J.-Y.; Mhenni, F.; Palladino, L.; Prosvirnova, T.; Rauzy, A.; Theobald, M. Synchronization of system architecture, multi-physics and safety models. In Proceedings of the Tenth International Conference on Complex Systems Design and Management, CSDM 2019, Paris, France, 12–13 December 2019. [CrossRef]
22. Batteux, M.; Prosvirnova, T.; Rauzy, A. *System Structure Modeling Language (S2ML)*; 2015. Available online: https://hal.archives-ouvertes.fr/hal-01234903/document (accessed on 29 April 2021).
23. Berriche, A.; Mhenni, F.; Mlika, A.; Choley, J.-Y. Towards Model Synchronization for Consistency Management of Mechatronic Systems. *Appl. Sci.* **2020**, *10*, 3577. [CrossRef]
24. Demachy, R.; Guilmeau, S. Structural consistency of MBSE and MBSA models using Consistency Links. In Proceedings of the Embedded Real Time Systems, ERTS 2022, Toulouse, France, 1–2 June 2022.
25. Grothendieck, A. Sur quelques points d'algèbre homologique, I. *Tohoku Math. J.* **1957**, *2*, 119–221. [CrossRef]
26. Spivak, D.I. *Category Theory for the Sciences*; Massachusetts Institute of Technology, Ed.; The MIT Press: Cambridge, MA, USA, 2014.
27. Schultz, P.; Spivak, D.I.; Vasilakopoulou, C. Dynamical Systems and Sheaves. *Appl. Categ. Struct.* **2020**, *28*, 1–57. [CrossRef]
28. Ehresmann, A.C. MENS, an info-computational model for (Neuro-)Cognitive systems capable of creativity. *Entropy* **2012**, *14*, 1703–1716. [CrossRef]
29. Ernadote, D. MB 2 SE: A Theoretical Foundation for Systems Engineering—Une Fondation Theorique Pour l'Ingenierie Systeme. HDR Thesis, Université Paris-Saclay, Gif-sur-Yvette, France, 2020. [CrossRef]
30. Abdeljabbar, N.; Mhenni, F.; Choley, J.-Y. A Categorical Framework for Collaborative Design of Safety Critical Mechatronic Systems. In Proceedings of the 7th IEEE International Symposium on Systems Engineering, ISSE 2021, Vienna, Austria, 13 September–13 October 2021. [CrossRef]
31. Ackerman, E.; Michael, K. Zipline's Medical Delivery Drones are changing the game in Rwanda, The blood is here. *IEEE Spectr.* **2019**, *56*, 24–31. [CrossRef]
32. Hagberg, A.; Schult, D.; Swart, P. NetworkX Reference (Release 2.7.1). 2011. Available online: https://networkx.org/ (accessed on 29 April 2021).
33. Rauzy, A.; Haskins, C. Foundations for model-based systems engineering and model-based safety assessment. *Syst. Eng.* **2019**, *22*, 146–155. [CrossRef]
34. Rauzy, A.B. Model-Based Reliability Engineering. Available online: http://www.altarica-association.org/members/arauzy/Publications/pdf/Rauzy2022-MBREBook.pdf (accessed on 29 April 2021).