Towards Model Synchronization in Model Driven Engineering of Mechatronic Systems

Aroua Berriche, Faïda Mhenni, Abdelfattah Mlika, Jean-Yves Choley

Towards Model Synchronization in Model Driven Engineering of Mechatronic Systems

Aroua Berriche
Quartz laboratory
Supmeca
3 rue Fernaut Hainaut, 93407 Saint-Ouen,France
aroua.berriche@supmeca.fr

Faïda Mhenni
Quartz laboratory
Supmeca
3 rue Fernaut Hainaut, 93407 Saint-Ouen,France
faida.mhenni@supmeca.fr

Abdelfattah Mlika
LMS laboratory
ENISo
Technopole de Sousse, 4054 Sousse, Tunisia
abdelfattah.mlika@gmail.com

Jean-Yves Choley
Quartz laboratory
Supmeca
3 rue Fernaut Hainaut, 93407 Saint-Ouen,France
jean-yves.choley@supmeca.fr

***Abstract*—Because of its multi-disciplinary nature, the development of a mechatronic system requires a collaborative effort of a variety of design teams from different disciplines. As a consequence, a variety of modeling languages, formalisms and tools are needed during the design process.**

On the other hand, the communication issues and poor interdisciplinary understanding limit the exchange of information among these teams. The main consequence is a high risk of inconsistency between the different views (or models) of the same system.

In this paper, we propose a model synchronization methodology to detect inconsistencies between the different models of a mechatronic system. The proposed method is composed of three phases: first, the entry models are abstracted into a common representation; second a comparison process of the abstracted models allows to identify potential inconsistencies, and finally the concretization phase acts to solve the detected inconsistencies.

This approach is illustrated on a case study from the automotive industry to validate its adequacy in enhancing collaboration among design teams during the design process.

***Index Terms*—Multi-view modeling, SysML, Modelica, Altarica, Consistency management.**

I. Introduction

Developing mechatronic systems requires the consolidation of models from a variety of disciplines such as mechanics, electronics, software engineering among others. Various designers having different perspectives on the overall system usually create these models using diverse formalisms. As they are performed by different actors these models may present some inconsistencies. To ensure consistency between different models of a mechatronic system, this work proposes a methodology to detect the potential differences between these models and then check whether they are inconsistencies, i.e. one or more designers omitted or added some parts, or just differences due to the specificities of the used modeling formalisms or to the purposes of the considered models.

The contribution of this paper is to provide a first step towards consistency management of heterogeneous models involved in the development of mechatronic systems, using a cooperative process to support the exchange between different designers.

The methodology allows multi-disciplinary interactions between multiple designers at an early stage of the development process that improve competitiveness and reduce development time and cost.

Although this work was driven by the interaction issue between three specific domains (i.e. system engineering, dynamic simulation and safety engineering), the proposed methodology is independent of the studied disciplines.Therefore, it could be used for further multi-disciplinary interactions.

Also, the methodology is constructed in such a way that experts do not need to understand other expertise's formalism. Since, they will manipulate more abstract representations based on graphs that add more flexibility and easier understanding of the overall design system without complete knowledge of the underlining formal systems.

The remainder of the paper is organized as follows. Section II depicts similar works that deal with assuring consistency between models. Section III gives a presentation of the proposed methodology. Section IV presents the case study of the Electronic Throttle Body (ETB). In section V, we discuss the outcomes of applying our methodology. Finally, the conclusion is given in the last section.

II. RELATED WORK

The use of models is a key part of developing and managing complex systems, relying on Model Driven Engineering (MDE) approach. This approach allows to

involve a multitude of engineers as well as heterogeneous formalisms, modeling languages and tools.

To improve the collaboration between different discipline engineers during complex systems development process, three types of approaches are typically used:

(1) Integration approaches that incorporate the different disciplines-specific views in a single model. For example, to integrate safety analysis in system engineering process, Mauborgne et al. in [1], [2] proposed to incorporate safety properties on system architectures viewpoints.

Moreover, CATIA V6 is presented as a commercial application which proposes a single tool with multiple views in [3]. The RFLP process corresponds to the different steps of the conceptual design (Requirement, Functional, Logical, Physical). This approach allows different disciplines to be integrated and be managed in a collaborative manner.

(2) Model transformation approaches that provide a mapping from one discipline to another. In this category, we distinguish two technologies:
First, the use of profiles or SysML (System Modeling Language) extensions to enrich SysML, transforming system models into another language as Modelica Language, the two well-known profiles are SysML4Modelica [4] and ModelicaML [5] link SysML to Modelica.

SafeSysE profile [6] which extend system model with some safety properties such as adding failure modes to functions and components. The system models are then used to generate some safety artifacts that will be used by safety experts.

Second, based on language transformation such as Triple Graph Grammar, Adourian et al. in [7] had built a meta-model of the relation between geometric (CAD) models of a mechanical system and the corresponding dynamic simulation models to assure consistency between the two views.

(3) Federative approaches [8] that aim at defining relationships between model elements with different concerns. In [9] a framework proposed to implement the federative approach using the powerful and rich semantics of the SysML language.

While these approaches allow consistency management between domain-specific models, they have several practical limitations. For one, the integration approaches require that designers must be adaptative in order to design their model with a single tool, i.e. designers are dependent on a single tool.

Although, model transformation approaches allow a diversity of expert tools for design. These approaches consider oriented relations encoded in the transformation rules. As a consequence, certain model transformation do not guarantee consistency between each model developed (e.g. a SysML model can be transformed into a Modelica model, while the modification made to the Modelica model may not be automatically transferred back to the SysML model).

Also, federative approaches can be criticized by the fact that the development of such technology must be capable of managing vast quantitites of data since it manipulates a database that contains information on each version of each component of a complex system.

Finally, some works proposed approaches focusing exclusively on managing inconsistencies.

(4) Inconsistency management approaches
Gausemeier et al. in [10] synchronized domain-specific models with a cross-domain system specification based on model transformation. By that, domain-specific models could be derived initially and changes in one model could be propagated via the cross-domain specification.

This approach is challenging due to the big number of modeling languages that can be used for multi-disciplinary systems. It requires huge effort of encoding large amounts of knowledge and information in the transformation.

The contribution in this paper is to provide a first step towards effectively managing inconsistencies in heterogeneous models, using a cooperative application to support dialog between engineering actors.

In this paper, we have selected three particular but representative modeling languages for illustrating our cooperative approach: SysML for systems engineering, Modelica for dynamic simulation and Altarica for safety engineering.

III. METHODOLOGY

To provide a first step towards model synchronization of mechatronic systems, a conceptual approach is proposed in this section.

The suggested approach allows to manipulate models, builds abstracted models and interacts with domain experts (i.e. designers) using interfaces. It enables to read models, to extract information from experts viewpoints (i.e. source viewpoints). It distinguishes and identifies differences in the information extracted from the source viewpoints. It enables an "interface" expert to propose solutions to the detected inconsistencies and to restore these proposals to manage inconsistencies between different experts viewpoints.

This approach, illustrated in Fig. 1, consists in identifying, classifying and managing differences and inconsistencies in the MDE process. In this work, we consider three different domain-specific models ($v_{0.1}$, $v_{0.2}$ and $v_{0.3}$) to evaluate the consistency between them. The model synchronization approach is based on three phases:

Model synchronization
=
Abstraction+ Comparison+ Concretization.

These phases will be described in the following.

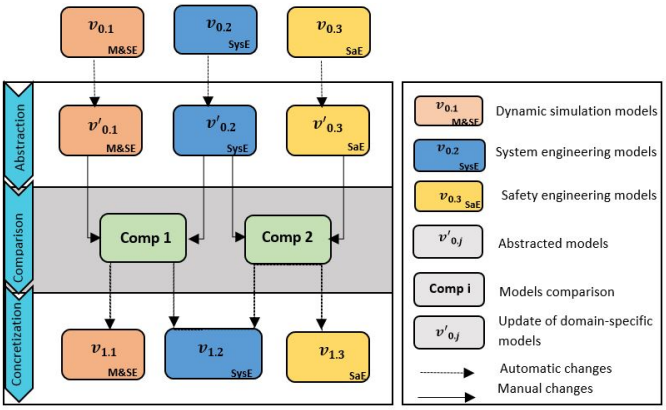


Fig. 1: Model Synchronization approach.

A. Abstraction

The first phase consists in defining a common representational formalism for models to extract knowledge and information encoded in various models at a same level of abstraction.

Therefore, we consider directed graph as a generic formalism to represent any model consisting in interrelated elements. A directed graph [11] is formed by vertices connected by directed edges.

For each entry model, we transform the elements into vertices and the relationships among elements into edges. "Model-to-Model" (M2M) transformation [12] is used to automatically generate abstracted models.

Table I presents an overview of structural constructs of some modeling languages such as SysML, Modelica and Altarica and shows how these constructs can be transformed into a directed graph.

TABLE I: Mapping between SysML, Modelica, Altarica and directed graph

SysML (IBD)	**Modelica**	**Altarica**	**Directed graph**
Part	Class	Block/Class	Vertex
Port	Connector	Flow variable	Vertex
Connector	Connection equation	Assertion	Edge
item flow	causal connection	Causal assertion	Edge direction
Binding connector	acausal connection	acausal assertion	

- Part is the basic element within an Internal Block Diagram (IBD) that describes blocks in the context of an owning block. A part can be compared respectively to a class in Modelica model and to a block or a class in Altarica model. These elements are associated with vertices in a directed graph.
- Ports provide a way for parts to interact and come in two types: "standard ports" and "flow ports". Ports can be roughly equivalent to connectors in Modelica model and to flow variables in Altarica model. These elements are transformed into vertices in a directed graph. To simplify the abstracted models, only external elements (i.e. external ports, connectors or flow variables) are represented via vertices. However, the internal elements (i.e. ports, connectors or flow variables connected to components) are omitted. The input and output direction of these elements are used to indicate direction edges in a directed graph.
- Connectors in SysML represent connections between parts via its ports through which energy or information is exchanged. They can be compared to connection equations in Modelica model and to assertions in Altarica model. They are mapped to edges in a directed graph to indicate the relationships between elements.
- item flows indicate the flow direction exchanged between parts in an IBD model. They can be compared to causal connections in Modelica model and causal assertions in Altarica model. The interaction between components is formalized in terms of input and outputs variables. These elements are transformed into unidirectional edges.
- Binding connectors in SysML model can be compared to acausal connections in Modelica model and acausal assertions in Altarica model. Modelica and Altarica allow acausal modeling, i.e the model is described by equations and the input-output causality between components is not fixed to promote model reuse. These elements are mapped to bidirectional edges in a directed graph.

Thus, it appears that a comparison can be made between some of SysML, Modelica and Altarica constructs.

As represented in Fig. 1, the abstraction phase permits obtaining three abstracted models ($v'_{0.1}$, $v'_{0.2}$ and $v'_{0.3}$) from entry models, represented in a common formalism using directed graphs to allow the comparison between models.

B. Comparison

Once the entry models are abstracted in a unique formalism, they can be compared.

Graph and subgraph isomorphisms [13] are usually applied to perform a comparison between two different graphs. Graph isomorphism is used to check whether two structures are similar and subgraph isomorphism is used to find whether an input graph is contained in a main graph.

In our study, we are convinced that the structure of different models cannot be the same due to the specification of different modeling languages. For that reason, we will use subgraph isomorphism in order to identify differences and inconsistencies between abstracted models.

Finding subgraph isomorphisms is an important problem in many applications like bioinformatics, chemistry and software engineering. Consequently, many algorithms have been proposed to solve this problem such as Ullmann

[14], Messemer and Bunke algorithm [15], VF2 [16], BB-Graph [17] and more.

For our comparison, we use NetworkX [18], "a python package for the creation, manipulation, and study of structure, dynamics, and functions of complex networks, to implement the comparison process". Since a number of graph algorithms are provided with NetworkX as VF2 for (sub)graph isomorphism.

Our program uses graph algorithms proposed by NetworkX to find:

- Possible isomorphism between graphs.
- Common subgraphs between abstracted models.
- Missing nodes in a graph compared to another.
- Missing edges in a graph compared to another.

The result of our program should be analyzed by an "interface" expert and a report should be created where differences and inconsistencies are identified and classified between the abstracted models.

The differences are authorized because they represent the specification of different modeling languages, while the inconsistencies should be analyzed and solved by the "interface" expert and then validated by the designers. Consequently, we need to update our entry models during the concretization phase.

As shown in Fig. 1, we consider the system engineering model as a reference model since system engineering provides an interdisciplinary approach and a framework for translating the stakeholders requirements into the definition of systems that enable the realization of successful systems. Systems engineering applies over the entire life cycle of complex systems, from concept development to final disposal.

Moreover, SysML is defined as a general modeling language for systems engineering applications and it supports the entire design life cycle as specification, analysis, design, verification and validation of complex systems.

This phase allows to carry out a consistency verification between multi-domain models created for the development of a mechatronic system.

C. Concretization

The last phase allows annotating the source models with the necessary corrections proposed by the "interface" expert considering the detected inconsistencies from the previous phase. If the corrective actions proposed by the "interface" expert are validated by designers, a transformation is executed to update entry models. The updated models are represented in their specific languages.

This phase is considered as the opposite process of the abstraction phase, where our goal is to refine the entry models based on the corrections proposed during the comparison phase. It will be implemented using a model-to-model transformation technique in future works.

The methodology presents many merits compared to other approaches described in the literature.

First, the suggested methodology applied the model synchronization approach to propose an automated and continuous strategy for consistency management that adds significant value to verification and validation process during mechatronic systems development, without encoding large amounts of transformation knowledge and information as suggested in the work focused on the inconsistency management in the related work.

Second, our methodology allows consistency management between diverse viewpoints of a complex system, meanwhile preserving the sources models of any undesirable information from other domains and respecting the specifications of each language used in the design process.

Also, This work enables the interaction between several designers at an adapted abstraction level while ensuring the separation of concerns (i.e. each designer use their own tool, modeling language and formalism. Furthermore, it allows the exchange between actors to identify and resolve possible inconsistencies at an early stage of development.

Moreover, the methodology is independent from the studied disciplines, which allow the possibility to apply the methodology in other multi-disciplinary interactions.

Finally, our methodology is better applied for inconsistency management in mechatronic systems, since graphs provide an extensible and flexible common representations to process and interpret heterogeneous models without a complete knowledge of the underlying formal languages.

IV. Case study

The studied system is an Electronic Throttle Body (ETB) from the automotive industry. An ETB is an actuator which controls the air supply to the engine to vary its output torque. This system improves vehicle emissions and drivability.

The system consists of a DC motor, a gearbox, a failsafe system with two springs and a position sensor for the Electronic Controller Unit (ECU) as shown in Fig. 2 [19].

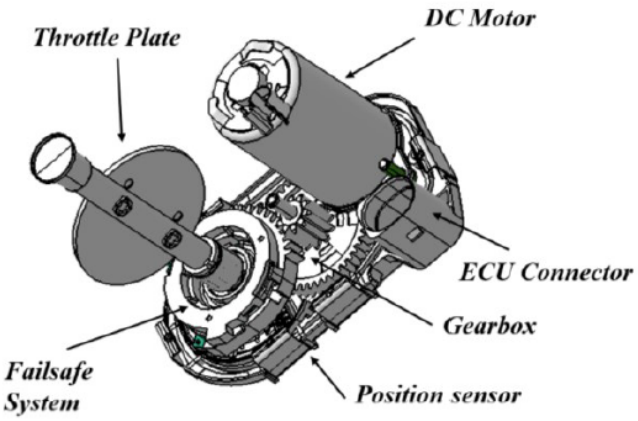


Fig. 2: Electronic Throttle Body architecture.

A. Design models of the ETB

During design process, various views of the system are established. In our scenario, we consider three different domain-specific models of the ETB system created during the engineering process:

A SysML model is developed for outlining the physical architecture design of the case study using an IBD diagram. A Modelica model is used to perform dynamic analyses to predict the system performances.

Finally, an Altarica model that assesses the dysfonctional behavior of the system.

In the following, these models are introduced in more details.

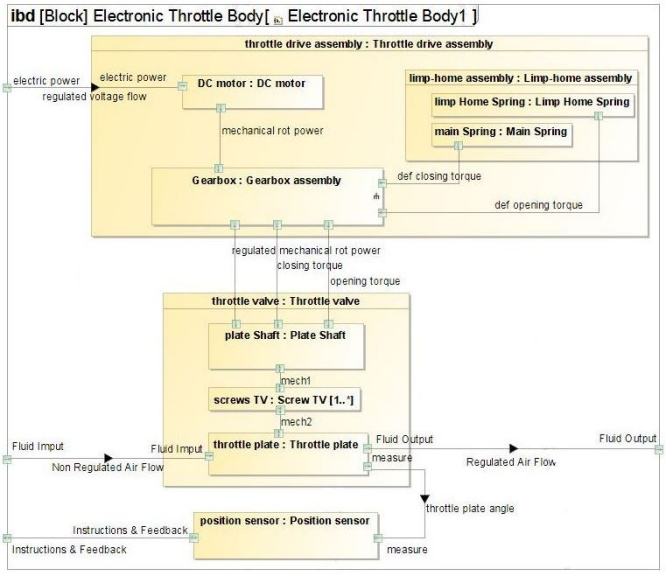


Fig. 3: Decomposition of the ETB (SysML Internal Block Diagram).

1) System Engineering Perspective (SysML):

SysML [20] supports the practice of Model-Based Systems Engineering (MBSE) [21] used to develop system solutions in response to complex and technological constraints.

SysML is a general-purpose graphical modeling language specified by OMG that supports the analysis, specification, design, verification, and validation of complex systems including hardware, software, data, procedure among others.

The SysML model is used for the purpose of formally capturing requirements, specifying the physical decomposition and describing the behavior of the system.

An IBD diagram represents the internal structure decomposition of a system and models the interconnections between components. Fig. 3 gives an overview of the the internal structure of the system via an IBD diagram. The main function of the ETB is to regulate the air of the combustion engine. This function is achieved by modifying the throttle plate angle in a given range $[\theta_{min}, \theta_{max}]$. This angle is controlled by a throttle drive assembly which is composed of a DC motor, a gearbox and a Limp-Home assembly. It is measured by a position sensor.

2) Dynamic simulation analysis (Modelica):

Modelica [22] is an object-oriented equation-based modeling language primarily aimed at physical systems. The language allows defining models in a declarative manner and supports modular and hierarchical modeling.

It also enables modeling large, complex and heterogeneous physical systems containing sub-components from multiple engineering domains (e.g. electrical, hydraulic, thermal, mechanical, etc.).

The Modelica language is standardized by the Modelica Association [22] and is supported by many commercial and open-source modeling tools.

To predict the performance of our case study, we modeled the ETB in an open loop system in Fig. 4.

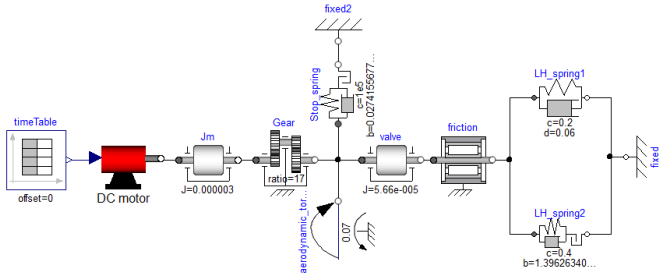


Fig. 4: Overview on the ETB simulation model (Modelica).

The Modelica model of the ETB is composed of essential components as a DC motor, a mechanical inertia, a gearbox, a valve, a failsafe system with two springs and other components depend on the study performance we conduct (e.g. the Stop spring that limits the rotation of the valve between 0° and 90° and the friction that induces the non-linearity of the system). In this way, we can test many architectures in an early stage.

3) System safety engineering (Altarica):

Altarica [23] is a high-level formal modeling language dedicated to safety analysis. Altarica semantics are defined using Guarded Transition Systems (GTS) [24], and for the structure aspect, Altarica is based on a System Structure Modeling Language (S2ML) [25] as a structural paradigm of models.

Once a system model is specified in the Altarica language, it can be compiled into a lower-level formalism such as fault trees, stochastic Petri Nets and other safety assessments can be performed.

In Fig. 5, we represent the global model of the ETB system in Altarica. It is composed of a DC motor, a gearbox, a Limp-Home Assembly, a sensor and a valve. The assertion represents the connections between components which represent the equalities between flow variables. Observers are defined to calculate the reliability indicators (e.g. the observer “topEvent” failed when the system cannot supply air to the valve).

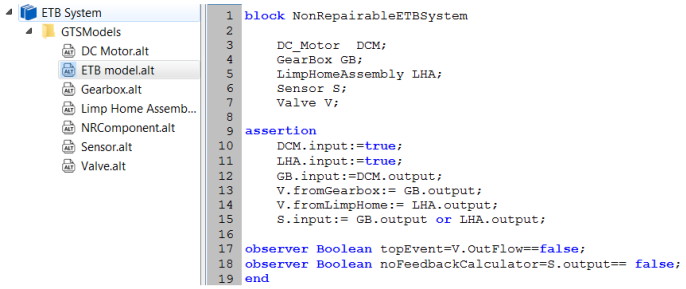


Fig. 5: Safety analysis of the ETB (Altarica).

B. Abstraction

The source viewpoints of system engineering, dynamic simulation and safety engineering of the ETB system are subject to a model transformation to obtain three abstracted views, retracing the structure of different models. As discussed in the previous section, we propose to transform our entry models into directed graph representations, a graphical representation of these abstractions are represented in Fig. 6.

We define by light blue vertices components belonging to the studied system and by gray vertices external elements of the system.

In our case study, the abstracted views of our entry models are composed of 13 vertices and 16 edges to represent the SysML model, 17 vertices and 33 edges to represent the Modelica model and 9 vertices and 8 edges to represent the Altarica model.

According to graph theory, we can prove that our abstracted models are not isomorphic since isomorphic graphs must have the same number of vertices and edges.

Hence, we should execute the comparison phase to evaluate the consistency between entry models.

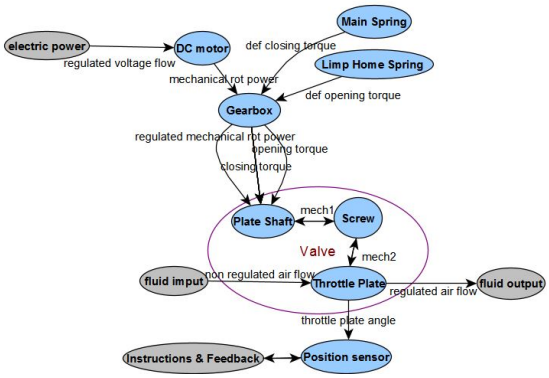
C. Comparison

We compare entry models using their directed graph representations. The comparison is a semi-automatic and an iterative process.

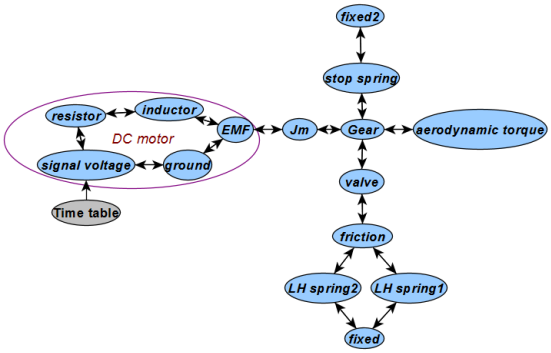
In our case study, five iterations were sufficient to evaluate the consistency between the abstracted models using the comparison process as described in the following:

The first iteration demonstrates just the node "DC motor" is identical between the three directed graph representations. A primary activity that the "interface" expert should handle is to find label correspondences between the abstracted models. This activity allows to obtain the Table II that summarizes the correspondences of component labels and defines common labels between the different abstracted models.

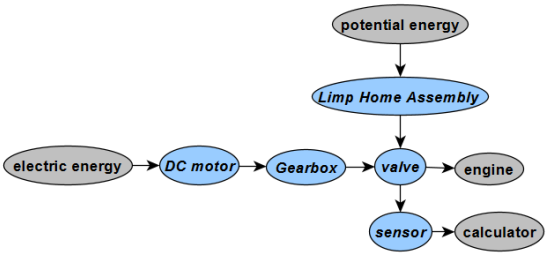
Based on the new labeling, the second iteration is performed. This iteration indicates that in the SysML and Altarica abstracted models the "DC motor" and the "Gearbox" are connected although in the Modelica abstracted model these elements are not connected. The



(a) SysML-IBD directed graph.



(b) Modelica directed graph.



(c) Altarica directed graph.

Fig. 6: Abstraction phase.

TABLE II: Label verification between abstracted models

SysML	Modelica	Altarica	Action
Gearbox	Gear	Gearbox	Gearbox
Position sensor	-	sensor	Position sensor
Instructions	-	calculator	calculator
Main Spring	LH spring1	Limp Home Assembly	Main Spring
Limp Home Spring	LH spring2	Limp Home Assembly	Limp Home Spring
electric power	time table	electric energy	Input
Valve	valve	valve	valve

"interface" expert evaluates this difference as an inconsistency between models due to a wrong decomposition of "DC motor" in the Modelica model. To solve this

inconsistency, the "DC motor" and the mechanical inertia "Jm" should be grouped in the Modelica abstracted model.

Implementing the third iteration, another difference is detected involving the abstracted models. The "Position sensor" is represented in SysML model and Altarica model, but it is not represented in the Modelica model. This difference is considered as an inconsistency because the target of modeling an ETB system is to evaluate the air regulation of the combustion engine. Therefore, The corrective action proposed by the "interface" expert consists on introducing the "Position sensor" in the Modelica abstracted model to simulate the opening degree of the valve.

The fourth iteration emphasizes another difference between models where the "Limp Home Spring" and the "Main Spring" are connected to different elements. In SysML abstacted model, they are connected to the "Gearbox". In Modelica abstracted model, they are connected to the "friction". Whereas, in Altarica abstracted model, a "Limp Home Assembly" is connected to the "valve". This difference is considered as an inconsistency between models. The "interface" expert suggests that the "Limp Home Spring" and the "Main Spring" should be connected to the valve in the different abstracted models and that the "Limp Home Assembly" should be decomposed in Altarica model.

After solving the detected inconsistencies between the abstracted models, the fifth iteration is executed and the "interface" expert evaluates that the remaining differences are due to the specificities of the used modeling formalisms or to the purposes of the considered models.

We represent in Table III and IV the results of the comparison process respectively between SysML and Modelica abstracted models and then between SysML and Altarica abstracted models.

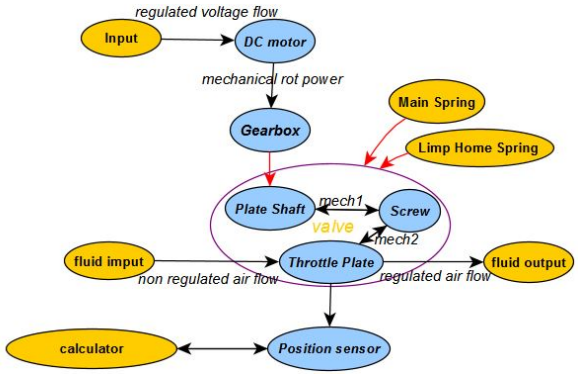
This case study allows detecting architectural differences between the various abstracted models such as: A different decomposition of the system components can be proposed, depending on the specific aim of each view. Specificities of different formalisms and concerns of different designers require to include new elements or to remove others from viewpoints.

Whereas, the inconsistencies can be the result of a wrong decomposition of the system elements or different interactions between components compared to the various models which represent contradictory information among models.

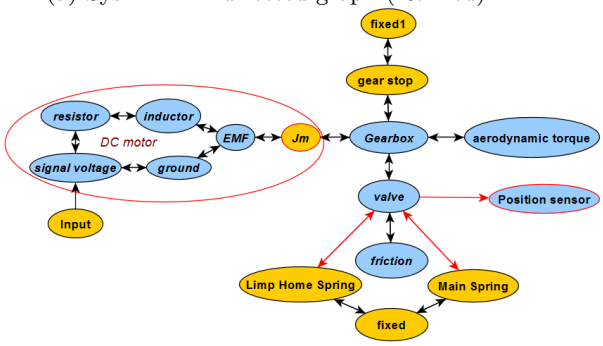
To conclude, during the comparison process the "interface" specialist can operate different actions to evaluate the consistency between abstracted models as follows:

- Add new elements (vertices and (or) edges) in an abstracted model.
- Delete elements (vertices and (or) edges) in an abstracted model.
- Change the element labels (vertices and (or) edges) in an abstracted model.
- move elements (vertices and (or) edges) in an abstracted model.
- Group or decompose elements (vertices and edges) in an abstracted model.

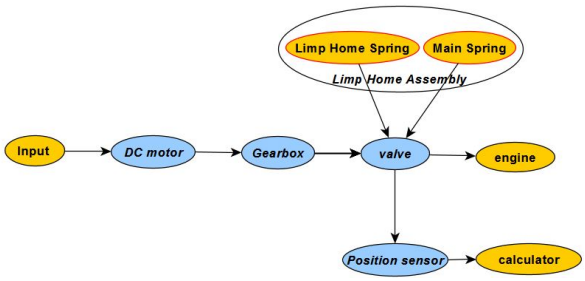
The final step in the comparison phase is annotating the abstracted models with corrections proposed by the "interface" expert in red vertices and (or) edges as represented in Fig. 7.



(a) SysML-IBD directed graph (Refined).



(b) Modelica directed graph (Refined).



(c) Altarica directed graph (Refined).

Fig. 7: Annotate abstracted models.

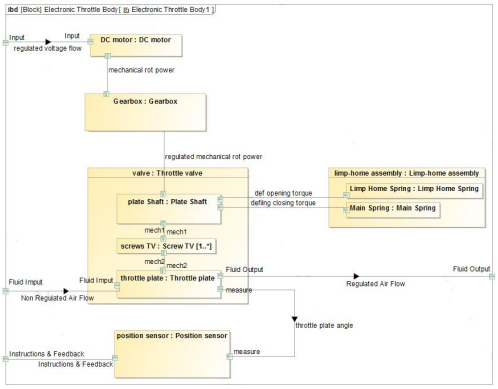
D. Concretization

After verification and validation of corrective actions by designers, the results of the comparison process driven by the "interface" expert can be applied on models in the concretization phase.

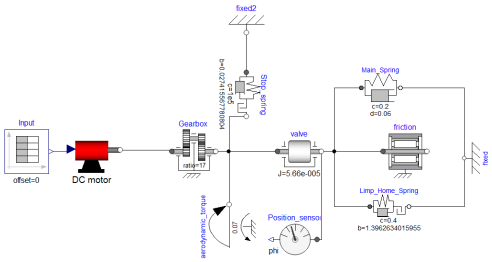
This phase allows updating entry models with chosen compromises between the different designers and the "interface" expert.

We represent in Fig. 8 the entry models refined based on model transformation. We obtained three models (i.e. SysML, Modelica and Altarica models) updated and represented in their specific modeling languages.

These models include consistent information to design the ETB system. Although, the specificities of each modeling language is guaranteed (e.g. in Modelica Model, the Stop spring and the aerodynamic torque are kept since they specify other aspects of the system).



(a) SysML-IBD model (Updated).



(b) Modelica model (Updated).

```
block NonRepairableETBSystem

    DC_Motor  DCM;
    GearBox GB;
    Limp_Home_Spring LHS;
    Main_Spring MS;
    Position_sensor S;
    valve V;

assertion
    DCM.input:=true;
    LHS.input:=true;
    MS.input:=true;
    GB.input:=DCM.output;
    V.fromGearbox:= GB.output;
    V.fromLimpHome:= LHS.output or MS;
    S.input:= GB.output or LHS.output or MS.output;

observer Boolean topEvent=V.OutFlow==false;
observer Boolean noFeedbackCalculator=S.output== false;
end
```

(c) Altarica model (Updated).

Fig. 8: Concretization phase.

V. Discussion

The methodology proposed provides a first step towards the implementation of an effective consistency management framework.

In this paper, we focused as a first step only on structural aspects of mechatronic systems. The behavioral part of models is considered to be purpose-dependent. Therefore, we are convinced that the main way to compare models is to compare their structures as the structure of models reflects the structure of the system in a limited extend. Moreover, we have suggested the methods for ensuring consistency between architectural models, but these methods can also be used in further works to manage consistency between behavioral models.

In the following, we generalize the various categories of differences considered in this paper. Firstly, we enumerate the different types of differences captured from the example application.

- **Unconventional naming** Models use different terms to refer to the same concept in the distinct domains, since each designer uses very specific vocabulary to represent his model. The fact that designers use distinct vocabulary can generate differences between models that have minor impacts on the system design. To bridge the gap between different domain models, a common terminology is required and thus allows automatic definition of interconnections between the modelled information.
- **Level of abstraction** Designers can model their views at different level of abstractions. As a consequence, the design information can be decomposed in different ways among the different views. For instance, the valve is decomposed in three sub-components in SysML model, but in the Modelica and Altarica models is considered as an atomic component. This difference is considered to have a minor impact in the design process as no contradictions are generated between models.
- **Specific design concerns** The multi-disciplinary modeling allows designers to concentrate on specific design concerns, and the various modeling languages used provides different packages that can be used to define specific concerns. For instance, the designer added some components to evaluate the non-linearity of the system in Modelica model.

Then, the various types of inconsistencies identified in our work are shown bellow.

- **Conflicting information** This type of inconsistencies can have severe impacts in the overall system design since it provides contradictory information between the different viewpoints. As an instance, the main spring and the limp home spring are linked to different components among the different models in our case study. This error is due to the separation of concerns and a misunderstanding and miscommunication between designers.
- **Level of abstraction** This type of differences can also be considered as inconsistencies if it produce severe impact in the systems development process. For example, in Modelica model the DC motor is decomposed into two parts that contradict the SysML and Altarica model where the DC motor is considered as an atomic component.

TABLE III: Results of the comparison between SysML and Modelica models (Comp1)

Comparison 1			
SysML	**Modelica**	**Differences**	**Inconsistencies**
electric power	Time table	Labeling	-
Main Spring	LH spring1	Labeling	Connection
Limp Home Spring	LH spring2	Labeling	Connection
Gearbox	Gear	Labeling	-
DC motor	DC motor	-	Decomposition
-	Jm	-	
Valve	valve	Labeling/Decomposition	-
-	fixed	Language specification	-
-	fixed2	Language specification	-
Position sensor	-	-	Concern
Instructions & Feedback	-	-	Concern
fluid input	-	Concern	-
fluid output	-	Concern	-
-	Stop spring	Concern	-
-	aerodynamic torque	Concern	-
-	friction	Concern	-

TABLE IV: Results of the comparison between SysML and Altarica models (Comp2)

Comparison 2			
SysML	**Altarica**	**Differences**	**Inconsistencies**
electric power	electric energy	Labeling	-
Position sensor	sensor	Labeling	-
Instructions & Feedback	calculator	Labeling	-
Main Spring	Limp Home Assembly	Labeling/Decomposition	Connection
Limp Home Spring	Limp Home Assembly	Labeling/Decomposition	Connection
Valve	valve	Labeling/Decomposition	-
fluid input	-	Concern	-
fluid output	-	Concern	-
-	engine	Concern	-
-	potential energy	Concern	-

- **The omission of essential elements in the system design** In this case, the inconsistency may lead from human errors and a miscommunication between designers. For example, the simulation designer did not present the sensor in his model, however the study conducted consists of controlling the air entered into the combustion engine so the sensor is considered as an essential element to predict the performance of our system.

The set of inconsistencies and differences types is very preliminary and restricted, since a small system was used as a case study. To expand the list of differences detected between various models, a larger case study needs to be employed that incorporates additional views and models common to designing mechatonic systems, allowing for further types of inconsistencies and differences to be identified.

Our methodology provides a first step towards implementing an efficient framework for managing consistency. Using the simple case study of the ETB designed through three models, we were able to demonstrate that models can be represented by graphs to allow the comparison of models at the same level of abstraction and then differences and inconsistencies are identified using graph comparison algorithms.

However, there are a number of challenges still to be addressed in the future works, when creating a general framework for consistency management in a multi-disciplinary design process . First, we should extend our methodology techniques to evaluate the consistency between behavioral models. Also, we intend to build a shared repository for different designers integrated in the design process to provide a common vocabulary used by the various disciplines to reduce the iterative cycles of the comparison process. Finally, we will automate the process of inconsistencies resolutions to reduce the dependence of this activity to the "interface" expert actions.

VI. Conclusion

Traditionally, system engineering, dynamic simulation and safety engineering aspects are described in different modeling languages (e.g. SysML, Modelica and Altarica). As a result, the system engineering model is decoupled from dynamic simulation and safety engineering models therefore the risk of inconsistencies is high. A complex system can fail due to miscommunication among engineers and resulting in wrong decisions taken during design process.

In this paper, we propose a methodology to evaluate consistency in multi-view modeling approach for complex

systems. We show how the proposed approach covers all phases of an early detection of inconsistency problems in mechatronic systems design between different views such as system engineering view, dynamic simulation view and safety engineering view for safety assessment.

The first phase transforms the different views of a system in a common representation based on directed graphs. The second phase allows to define a mapping of elements between models by comparing the system structures of these models and evaluate differences and inconsistencies detected between abstracted models. The final phase consists of managing differences and inconsistencies in order to provide consistent information between designers of the different viewpoints.

Using the methodology described in this paper, engineers will be capable of managing system consistency through defining a mapping between structuring constructs of modeling languages at a higher level of abstraction while still maintaining the benefits of preserving the sources models of any undesirable information from other domains and accept the specification of each language used in the design process. Moreover, the proposed approach combines the benefits of synchronizing modeling languages and the ease of communication between experts during the design life cycle.

In order to consolidate our work, a case study on the Electronic Throttle Body (ETB) was given.

Future works should focus on two aspects: 1) proving the technical viability and practicality, and measuring the effectiveness of the methodology by implementing a set of supporting tools and 2) investigate the possibility of applying our conceptual approach to evaluate the behaviour consistency of a complex system.

References

[1] P. Mauborgne, S. Deniaud, E. Levrat, E. Bonjour, J.-P. MicaÃńlli, and D. Loise, "Preliminary Hazard Analysis Generation Integrated with Operational Architecture - Application to Automobile," in *Complex Systems Design & Management* (F. Boulanger, D. Krob, G. Morel, and J.-C. Roussel, eds.), pp. 297–309, Cham: Springer International Publishing, 2015.

[2] P. Mauborgne, S. Deniaud, É. Levrat, É. Bonjour, J.-P. Micaëlli, and D. Loise, "The determination of functional safety concept coupled with the definition of logical architecture: a framework of analysis from the automotive industry," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 7278–7283, 2017.

[3] S. Kleiner and C. Kramer, "Model based design with systems engineering based on rflp using v6," in *Smart Product Engineering*, pp. 93–102, Springer, 2013.

[4] C. J. Paredis, Y. Bernard, R. M. Burkhart, H.-P. de Koning, S. Friedenthal, P. Fritzson, N. F. Rouquette, and W. Schamai, "An Overview of the SysML-Modelica Transformation Specification," *INCOSE International Symposium*, vol. 20, pp. 709–722, July 2010.

[5] W. Schamai, P. Fritzson, C. Paredis, and A. Pop, "Towards Unified System Modeling and Simulation with ModelicaML: Modeling of Executable Behavior Using Graphical Notations," pp. 612–621, Oct. 2009.

[6] F. Mhenni, N. Nguyen, and J.-Y. Choley, "SafeSysE: A Safety Analysis Integration in Systems Engineering Approach," *IEEE Systems Journal*, vol. 12, pp. 161–172, Mar. 2018.

[7] C. Adourian and H. Vangheluwe, "Consistency between geometric and dynamic views of a mechanical system," in *Proceedings of the 2007 summer computer simulation conference*, p. 31, Society for Computer Simulation International, 2007.

[8] C. Guychard, S. Guerin, A. Koudri, A. Beugnard, and F. Dagnat, "Conceptual interoperability through models federation," in *Semantic Information Federation Community Workshop*, p. 23, 2013.

[9] K. Thramboulidis, "The 3+1 SysML View-Model in Model Integrated Mechatronics," *Journal of Software Engineering and Applications*, vol. 03, no. 02, pp. 109–118, 2010.

[10] J. Gausemeier, W. Schäfer, J. Greenyer, S. Kahl, S. Pook, and J. Rieke, "Management of cross-domain model consistency during the development of advanced mechatronic systems," in *DS 58-6: Proceedings of ICED 09, the 17th International Conference on Engineering Design, Vol. 6, Design Methods and Tools (pt. 2), Palo Alto, CA, USA, 24.-27.08. 2009*, 2009.

[11] K. Ruohonen, "Graph theory. tampereen teknillinen yliopisto. originally titled graafiteoria, lecture notes translated by tamminen, j., lee, k," *C. and Piché, R*, 2013.

[12] T. Mens and P. Van Gorp, "A Taxonomy of Model Transformation," *Electronic Notes in Theoretical Computer Science*, vol. 152, pp. 125–142, Mar. 2006.

[13] R. Somkunwar and D. V. M. Vaze, "Challenges and Issues of Graph and Subgraph Isomorphism," *International Journal of Computer Technology Applications*, vol. 8, p. 5, 2017.

[14] J. R. Ullmann, "An Algorithm for Subgraph Isomorphism," *Journal of the ACM*, vol. 23, pp. 31–42, Jan. 1976.

[15] B. T. Messmer and H. Bunke, *Subgraph isomorphism in polynomial time.* Universität Bern. Institut für Informatik und Angewandte Mathematik, 1995.

[16] L. Cordella, P. Foggia, C. Sansone, and M. Vento, "Performance evaluation of the VF graph matching algorithm," in *Proceedings 10th International Conference on Image Analysis and Processing*, (Venice, Italy), pp. 1172–1177, IEEE Comput. Soc, 1999.

[17] M. Asiler and A. Yazıcı, "BB-Graph: A new subgraph isomorphism algorithm for efficiently querying big graph databases," *arXiv preprint arXiv:1706.06654*, 2017.

[18] A. Hagberg, D. Schult, and P. Swart, "Networkx reference, release 2.1," 2018.

[19] M. Mcharek, M. Hammadi, J.-Y. Choley, T. Azib, and C. Larouci, "Modeling and multi-objective optimization of an electronic throttle in open-loop," in *2016 11th France-Japan & 9th Europe-Asia Congress on Mechatronics (MECATRONICS)/17th International Conference on Research and Education in Mechatronics (REM)*, pp. 331–335, IEEE, 2016.

[20] OMG, "OMG Systems Modeling Language (OMG SysML)," 2012.

[21] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language.* Waltham, MA: Morgan Kaufmann, 2nd ed ed., 2012. OCLC: ocn754518532.

[22] M. Association, "Modelica - A Unified Object-Oriented Language for Systems Modeling Language Specification Version 3.4," 2017.

[23] A. Association, "AltaRica 3.0 Language Specification Version 1.1," 2017.

[24] A. B. Rauzy, "Guarded transition systems: A new states/events formalism for reliability studies," *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, vol. 222, pp. 495–505, Dec. 2008.

[25] M. Batteux, T. Prosvirnova, and A. Rauzy, "System Structure Modeling Language (S2ML) Language Specification- Version 1.0," p. 53, 2015.