Synchronization of system architecture, multi-physics and safety models

Michel Batteux, Jean-Yves Choley, Faida Mhenni, Luca Palladino, Tatiana Prosvirnova, Antoine Rauzy, Maurice Theobald

Synchronization of system architecture, multi-physics and safety models

Michel Batteux[1], Jean-Yves Choley[2], Faïda Mhenni[2], Luca Palladino[5], Tatiana Prosvirnova[3,4], Antoine Rauzy[6], and Maurice Theobald[5]

[1] IRT SystemX, Palaiseau, France
[2] Quartz Laboratoire, Supmeca, Saint-Ouen, France
[3] Laboratoire Genie Industriel, CentraleSupélec, Gif-sur-Yvette, France
[4] ONERA/DTIS, UFTMiP, Toulouse, France
[5] SAFRAN Tech, Châteaufort, France
[6] Norwegian University of Science and Technology, Trondheim, Norway

Abstract. To face the growing complexity of technical systems, engineers have to design models in order to perform simulations. To avoid inconsistencies, the integration of different models coming from various engineering disciplines is one of nowadays main industrial challenges. In this article we present model synchronization, a framework to ensure consistency between models coming from different engineering domains, based on S2ML (System Structure Modeling Language). We show how the introduced framework can be used to handle consistency between system architecture models (using SysML language), safety models (using AltaRica 3.0 language) and multi-physics simulation models (using Modelica language).

Keywords: Heterogeneous models · model synchronization · model structuring · AltaRica · SysML · Modelica

1 Introduction

Technical systems are getting more and more complex. To face the increasing complexity of systems, engineers are designing models. These models have different maturity levels and are designed at different abstraction levels and for different purposes. The integration of models coming from various engineering disciplines, such as system architecture, control, multi-physics simulation, automatic code generation, safety and performances analyses, is one of today's industrial challenges.

Collaborative data bases (PDM/PLM) and tools to set up traceability links between models provide a support to manage models in version and configuration, but not to ensure consistency between them. Model transformation techniques (e.g. [7], [12], [3], [10]) assume a master/slaves organization of models, which is not realistic in practice.

In this article we present model synchronization – a framework to ensure consistency between models coming from different engineering domains. In this

approach each engineering discipline uses its own modeling languages and tools which makes it possible to have flexibility and to be efficient in conducting virtual experiments on the system under study.

The framework is based on the thesis that systems engineering modeling formalisms are made of two parts:

- An underlying mathematical framework, which aims at capturing some aspects of the system behavior, e.g. differential equations for Modelica [5] and Matlab-Simulink, Data-Flow equations for Lustre, Guarded Transition Systems for AltaRica 3.0;
- A structuring paradigm that makes it possible to build and organize models by assembling parts into hierarchical descriptions.

Behavioral descriptions are specific to each engineering domain. On the contrary, the structures of models reflect to some extent the structure of the system under study. Therefore, our framework focuses on structural comparisons and is based on S2ML (System Structure Modeling Language) [1]. Models from different engineering domains cannot be compared directly. First, they are abstracted into a pivot language (S2ML). Second, their abstractions are compared. To support model synchronization we develop the SmartSync platform, which is used to compare S2ML abstractions of heterogeneous models.

To illustrate our proposal we use a case study – an Electro-Mechanical Actuator (EMA) of an aileron for a small aircraft. We show how the introduced framework can be used to handle consistency between system architecture models (designed in SysML [4]), multi-physics models (designed in Modelica [5]) and safety models (designed in AltaRica 3.0 [2]).

This work continues the work on model synchronization presented in [9] and [6]. An interesting study [8] uses model synchronization techniques with hierarchical graphs.

The remainder of this article is organized as follows. Section 2 introduces the case study. Section 3 describes the model synchronization framework. Section 4 presents the results. Finally, section 5 concludes this article and discusses future works.

2 Case study

The considered case-study is an Electro-Mechanical Actuator (EMA) for general aviation small aircraft. The EMA is intended to actuate the aileron, replacing the usual rod, cables and lever mechanisms. The proposed actuator is driven by the aircraft electrical networks, controlled by the on-board FCC (Flight Control Computers) with a set point consistent with the pilot instructions, taking into account the aileron feedback position and the EMA feedback.

There are different relevant kinematic architectures such as a 4-bars with a crank and rod mechanism, a 3-bars with an electric cylinder or a direct drive with a motor and a gearbox mounted on the axis of the revolute joint between the wing and the aileron. In this work, we will focus on the 3-bars architecture.

This architecture is illustrated in Fig.1. Linked to the wing and the aileron with two spherical joints, the EMA is made up of a housing that encapsulates all the components, a DC motor controlled by a Micro Controller Unit (MCU) (not represented), a gearbox and a screw and nut assembly to transform the gearbox output rotation into a translation of a rod that will in turn push or pull the aileron.

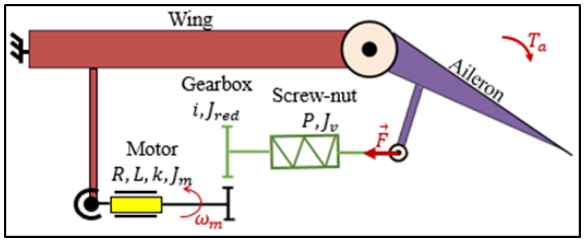

Fig. 1. EMA 3-bars architecture.

3 Model synchronization

3.1 Principle

Integration of engineering models can be achieved by model synchronization process, i.e. the process by which one can ensure that two possibly heterogeneous models are "speaking" about the same system. Two models, written into two different languages, can generally not be directly compared. The idea is thus to abstract them into a pivot language and to compare their abstractions (see Fig. 2). The synchronization of models goes in two steps. The first step of model synchronization process is the abstraction, which consists in extracting the common part that can be compared from the models. The second step is to compare the abstractions. The third step, the so-called concretization, consists in eventually adjusting initial models if inconsistencies have been detected.

3.2 S2ML as a pivot language

S2ML (System Structure Modeling Language) [1] aims at providing a structuring paradigm of systems engineering modeling languages. It unifies concepts coming from object-orientedand prototype-orientedprogramming languages. As heterogeneous models can be essentially compared by their structure, S2ML is a perfect candidate as a pivot language for the abstraction.

S2ML is made of the four basic elements: ports, connections, blocks and attributes. Ports are basic objects of models (e.g. variables, events, parameters). Connections are used to describe relations between ports (e.g. equations, transitions, assertions). Blocks are containers composed of ports, connections and

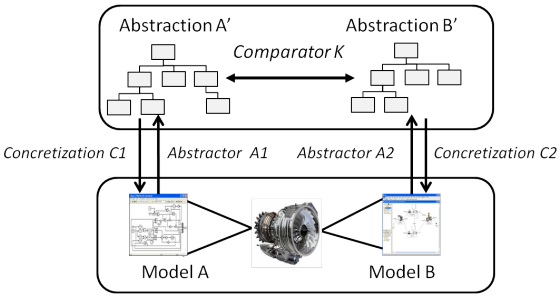

Fig. 2. Model synchronization: principle.

other blocks. Attributes are couples (name = value) used to associate information to ports, connections and blocks.

Example: Consider a non repairable component (*NRComponent*) in AltaRica 3.0 having a Boolean state variable *vsWorking* and a failure event *evFailure*. Its S2ML abstraction would be as illustrated in Fig. 3.

```
class NRComponent
  port vsWorking(kind="variable", type="Boolean", init="true");
  port pLambda(kind="parameter", type="Real", value="1.0e-5");
  port evFailure(kind="event", delay="exponential(pLambda)");
  connection [ evFailure, vsWorking ](type="transition", guard="vsWorking",
        action="vsWorking := false");
end
```

Fig. 3. S2ML code for the block *NRComponent*.

The class *NRComponent* contains three ports *vsWorking*, *evFailure* and *pLambda* having different attributes, and a connection, which represents the transition labeled by the event *evFailure*. In S2ML, ports, connections and blocks are interpreted by themselves. But a particular modeling language, implementing S2ML as its structuring paradigm, can give a concrete interpretation to ports, connections and blocks. For example, in AltaRica 3.0 variables, parameters, events and observers are interpreted by ports; transitions and assertions are interpreted by connections, blocks are interpreted by blocks.

S2ML provides three relations: composition, inheritance and aggregation.

Composition is the simplest structural relation: a system composes a component means that the component "is part of" the system. In S2ML, the composition is represented by adding different components within the code of the system as shown in the example below.

Example: In the example given in Fig. 4, the block *EMASystem_1* contains blocks *ElectricPower*, *MCU* and *Motor* and also other blocks, ports and connections not represented here.

Inheritance makes it possible to an element (block or class) to acquire all the properties of another element without explicitly duplicating them. Inheri-

```
class Motor
  extends NRComponent;
  port vfFromMCU (type ="Boolean", reset = "false");
  port vfToGearbox (type="Boolean", reset = "false");
  connection assertion [vfToGearbox, vsWorking, vfFromMCU];
end
block EMASystem_1
  // ports
  block ElectricPower
    extends NRComponent;
    port vfToMCU ( type = "Boolean", reset = "false");
    connection assertion[ vfToMCU, vsWorking];
  end
  block MCU
    extends NRComponent;
    // the remainder of the block MCU
  end
  Motor Motor;
  // the remainder of the block EMASystem_1
end
```

Fig. 4. S2ML abstraction of the AltaRica 3.0 model of the EMA system.

tance implements the "is a" relation between modeling elements. In S2ML, the inheritance is represented by the keyword "extends".

Example: In the AltaRica 3.0 model of the EMA system, all the components extend the class *NRComponent* (see Fig. 3) as they may fail in operation. In the example given in Fig. 4, the block *ElectricPower* extends the class *NRComponent* defined previously. It contains all the ports and connections of the class *NRComponent* and adds a port *vfToMCU* and a connection *assertion*.

Aggregation is a "uses" relation between modeling components. It makes it possible to represent components which are not a part of the subsystem and may be shared by several subsystems. The clause "embeds" in S2ML refers to an aggregation.

S2ML also proposed two different ways to reuse modeling elements: Prototype/Cloning and Class/Instance mechanisms.

The first way comes from prototype-oriented programming languages. A block is a container for ports, connections and other blocks. Each block is a prototype, i.e. it has a unique occurrence in the model. A system may contain similar components or subsystems. To avoid duplicating the description of a block, it is possible to clone an already existing one. In S2ML, the cloning of a block is obtained by the keyword "clones".

The second way to avoid duplicating the description of a block originates from object-oriented programming languages and consists in declaring a model of the duplicated block in a separate modeling entity, the so-called class, and then in instantiating this class wherever we need to use it again. Obviously, the class is referred to by the keyword "class" in S2ML. Each instance of the class is obtained by writing the name of the class followed by names of the created instances.

Example: In the example given in Fig. 4 a class *Motor* is defined. It is instantiated inside the block *EMASystem_1*.

Unfolded model Any hierarchical model is semantically equivalent to an unfolded (also called instantiated) one. An unfolded S2ML model is a model made of a hierarchy of nested or aggregated blocks, connections and ports. This model is obtained by applying recursively rewriting rules, the so-called unfolding rules. These rules resolve inheritance, classes instantiation, blocks cloning and paths of aggregated elements. An unfolded (or instantiated) model is used in the comparison step of the model synchronization.

3.3 SmartSync platform

The proposed platform for model synchronization SmartSync is illustrated in Fig. 5. The first step of the model synchronization is the abstraction, which

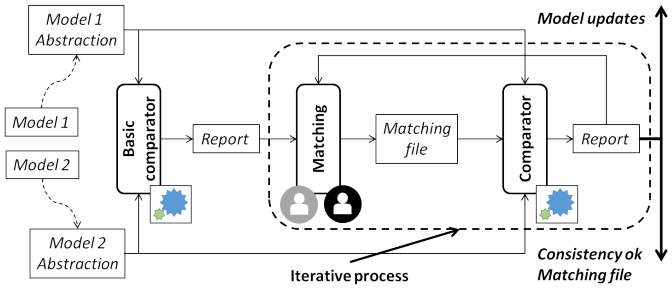

Fig. 5. Models synchronization process.

consists in translating models into S2ML. This step is still done manually for the moment but it can be automated. In the next step, the abstractions of the different models are compared two by two and a report of the comparison is generated. This report is then analyzed by the members of the different teams that built the initial models. Together, they produce a matching file that matches the same elements in the two models. The next step of the comparison process consists in comparing the initial models using the matching file. Another report is then generated that contains a list of inconsistencies. This report is analyzed again by the members of both teams. The matching file is updated with new corresponding elements. The updated matching file is used again in the comparison of the model abstractions and so on. The process iterates until all the inconsistencies have been resolved. At each iteration, if an inconsistency is detected, one or both models should be updated.

The outcome of the model synchronization is twofold. First, it allows to detect model inconsistencies in which case models need to be updated. Second, it allows to validate the model consistency. Models can then be used to produce performance indicators and so on.

Different types of comparators (see [11] for an interesting survey on model comparison techniques) for S2ML models can be defined, for instance:

– Dictionary, which consists in matching the names of different elements (ports, nested/aggregated blocks and connections);
– Structural, which consists in matching the names of different elements and the structure of the model;
– Topological, which consists in matching the names of different elements, the structure of the model and the connections between ports.

Note that the choice of abstractors and comparators depends on the system under development and the level of maturity of the project.

4 EMA case study: model synchronization

We present a collaborative design of the EMA system introduced in Section 2. The collaboration is between three teams: system architecture, multi-physics simulation and safety. Each team performs different activities. The first activity is modeling, i.e. the creation of models, which is performed independently by members of each team using different modeling languages and tools. The second activity is model synchronization, i.e. the verification of consistency between models that ensures that models are describing the same system. This activity is performed by the members of both teams (for example, system architecture and safety teams) and involves the SmartSync platform. The results of this activity can be twofold: models can be validated or inconsistencies can be detected.

4.1 Modeling

The EMA system is modelled from three different points of view: system architecture, multi-physics simulation and safety.

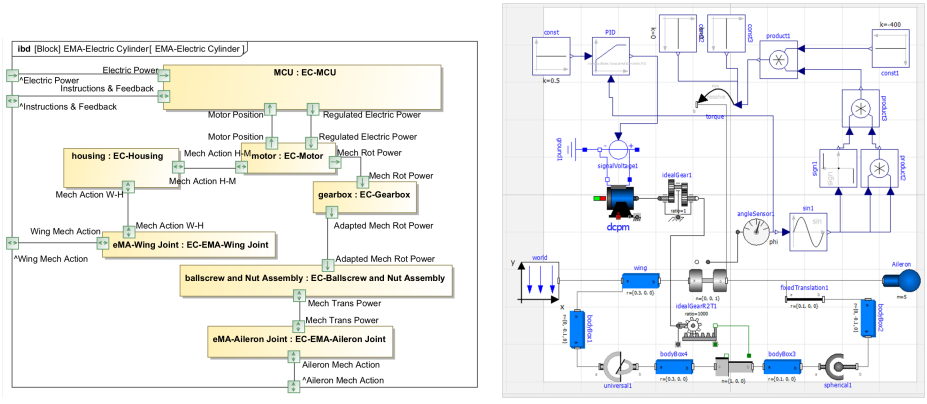

Fig. 6. SysML and Modelica models of the EMA system.

System architecture model is created using SysML [4] with a particular focus on system physical architecture. The internal block diagram representing the

EMA physical architecture is given Fig. 6 on the left. It has been done using SysML plugin of MagicDraw modeling tool. This model has no redundancies and does not represent the system environment. The incidence sensor is supposed to be a part of the block *Motor*; it is represented by a port *Motor Position* of the block *Motor* connected to the port *Motor Position* of the block *MCU*.

The multi-physics model of the EMA is designed with Modelica [5] modeling language and the OMEdit tool [7]. Its graphical representation is given Fig. 6 on the right. It represents not only the EMA system itself but also its environment (e.g. the wing, the aileron, the pilot commands, etc.) in order to be simulated.

The safety model is created using AltaRica 3.0 modeling language [2] and the OpenAltaRica platform [8]. Fig. 7 shows the graphical representation of the AltaRica 3.0 model of the EMA system. This model is an extended reliability block diagram, where blocks represent system components and their failures, connections between blocks – the propagation of failures. The block *Observer* models the failure condition – loss of the aileron incidence control.

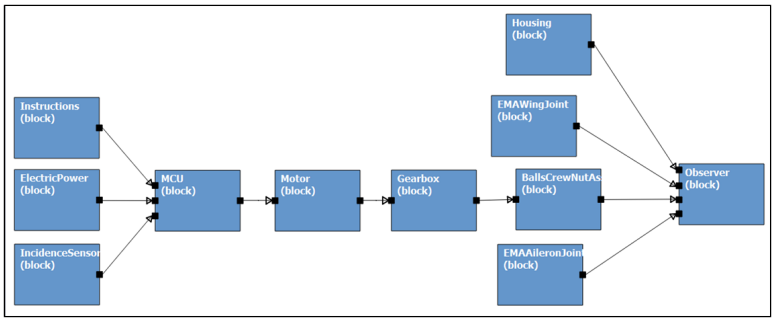

Fig. 7. Graphical representation of the AltaRica 3.0 model of the EMA system.

4.2 Synchronization of system architecture and multi-physics models

First, both models are abstracted, i.e. transformed into S2ML. For SysML internal block diagrams the transformation is quite simple: blocks/parts are transformed into S2ML blocks, ports into S2ML ports and connections between ports are transformed into S2ML connections between the corresponding S2ML ports.

For Modelica the choice has been done to completely abstract the internal behavior of each Modelica class, only variables involved in the "connect" clause are considered. Thus, Modelica classes are transformed into S2ML classes, instances of classes are transformed into instances of the corresponding S2ML classes, variables involved in the "connect" clause are transformed into S2ML ports,

[7] https://openmodelica.org/
[8] https://www.openaltarica.fr/

"connect" clauses between variables are transformed into connections between the corresponding ports in S2ML. All the other Modelica elements (variables, parameters, equations, etc.) are not considered in the S2ML abstraction.

In the next step, the abstractions are compared and a report is generated. This report is analyzed by members of both teams. The following differences are detected:

- Different names of blocks (e.g. the block *Motor* in the SysML model corresponds to the block *dcpm* in the Modelica model);
- SysML block corresponding to several Modelica blocks (e.g. the block *BallScrewAndNutAssembly* in the SysML model corresponds to the blocks *idealGearR2T1*, *prismatic1*, *bodyBox3* and *bodyBox4* in the Modelica model);
- Elements of the multi-physics simulation model not represented in the system architecture model (e.g. blocks *world*, *Aileron*, *wing* have no equivalent in the system architecture model as they are part of the system environment).

All the differences are listed in the matching file, which makes it possible to establish the correspondence between the two models. The following table shows an extract of a matching file.

Type	**Model1 (SysML)**	**Model2 (Modelica)**
block	main.EMASystem_1	main.EMA_3bars_BF_sin2PID
block	BallScrewAndNutAssembly	idealGearR2T1, prismatic1, bodyBox3, bodyBox4
block	EMAAileronJoint	spherical1
block	EMAWingJoint	universal1
block	Gearbox	idealGear1
block	MCU	PID, signalVoltage1
block	Motor	dcpm
block	forget	wing
block	forget	Aileron
block	forget	world
...	...	...

The first column is the element type (port, block, aggregated block or connection). The second column is the name of the element of the first model, the third column is the name of the corresponding element in the second model. When there is no correspondence, the keyword *forget* is used. It is possible to add a fourth column with comments to justify matching decisions.

As we can see, for example the block *wing* of the Modelica model has no correspondence in the system architecture model because it belongs to the system environment but it is needed to be able to perform multi-physics simulations. The block *MCU* in the SysML model corresponds to two blocks in the Modelica model: *PID* and *signalVoltage1*.

The produced matching file is used to compare again the abstractions of the system architecture and multi-physics simulation models. In the next step of the comparison, new differences are detected. They are analyzed again by the members of both teams and several inconsistencies are detected. Some of them are summarized in the following table.

Type	**Model1 (SysML)**	**Model2 (Modelica)**	Comments
block	EMASystem_1	EMA_3bars_BF_sin2PID	
block	EMAWingJoint	universal1	
port	MechanicalActionHW		Error in SysML
port	WingMechanicalAction	frame_a	
port		frame_b	Error in SysML
block	Gearbox	idealGear1	
port	AdaptedMechanicalRotPower	flange_a	
port	MechanicalRotPower	flange_b	
block	Motor	dcpm	
port	MechanicalAction		Not implemented in Modelica
port	MechanicalRotPower	flange	
port	RegulatedElectricalPower	pin_ap	
port	forget	pin_an	
...	...	...	

There is an error in SysML model: the connection *MechanicalActionHW* between *Housing* and *EMAWingJoint* has to be replaced by a connection between *BallScrewAndNutAssembly* and *EMAWingJoint*.

As we can see, one of the possible outcomes of the model synchronization is the detection of inconsistencies. In this case initial models should be adjusted.

There are other approaches to create links between SysML and Modelica models, for instance ModelicaML [10].

4.3 Synchronization of system architecture and safety models

As previously, first, both models are abstracted. For AltaRica 3.0 the transformation is straightforward, as the language uses S2ML as its structural paradigm. State and flow variables, events and parameters are abstracted to S2ML ports; transitions and assertions are transformed into connections; different structural constructs like inheritance, cloning, instantiation, etc. are transformed into their equivalents in S2ML.

Then the abstractions are compared and a report is generated. This report is analyzed by members of both teams. The following differences are detected:

- Different names of blocks (e.g. the block *BallscrewAndNutAssembly* in the SysML model corresponds to the block *BallsCrewNutAssembly* in the AltaRica 3.0 model);
- Different names of ports (e.g. the port *Motor.RegulatedElectricPower* in the SysML model corresponds to the port *Motor.vfFromMCU* in the AltaRica 3.0 model);
- Elements of the system architecture model not represented in the safety model (e.g. *Motor.MechanicalActionHM* has no correspondence in the safety model);
- Elements of the safety model not represented in the system architecture model (e.g. state variables, failure events, parameters, etc. have no equivalent in the system architecture model).

The following table shows an extract of a matching file.

Type	**Model1 (SysML)**	**Model2 (AltaRica 3.0)**
block	EMASystem_1	EMASystem_1
port	ElectricalPower	ElectricPower.vfToMCU
port	InstructionAndFeedback	Instructions.vfToMCU
block	forget	Observer
block	BallScrewAndNutAssembly	BallScrewNutAssembly
block	EMAAileronJoint	EMAAileronJoint
port	AileronMechanicalAction	vfOut
port	MechanicalTransmissionPower	forget
port	forget	evFailure
port	forget	pLambda
port	forget	vsWorking
...	...	...

As we can see, the block *Observer* of the safety model has no correspondence in the system architecture model because it represents safety related information (i.e. the failure condition to study). The port *ElectricalPower* in the SysML model corresponds to the port *ElectricPower.vfToMCU* in the AltaRica 3.0. It is important to note that the block *ElectricPower* of the safety model has no equivalent in the architecture model. In the system architecture model this block is not represented as it belongs to the system environment, whilst the safety analyst decided to represent it in his model because the failure of the electric power causes the occurrence of the failure condition.

The produced matching file is used to compare again the abstractions of architecture and safety models. In the next step of the comparison, new differences are detected. They are analyzed again and the matching file is populated with new matching information. Models are compared again. Finally, no more differences are detected. The consistency between system architecture and safety models is verified. The matching file establishes the correspondence between the two models.

5 Conclusion and perspectives

In this article, we presented experiments on the synchronization of system architecture, multi-physics simulation and safety models of an electro-mechanical actuator for an aileron of a small aircraft. We showed that model synchronization can be used to ensure the consistency of heterogeneous models, designed within different formalisms and different modeling environments.

To support model synchronization, we developed the SmartSync platform, which relies on S2ML as a pivot language. With SmartSync, we studied the EMA system. We checked consistency between system architecture and safety models and detected inconsistencies between system architecture and multi-physics simulation models. The process of making models consistent is iterative and involves representatives of the engineering disciplines at stake. The SmartSync platform

helps not only to check the consistency between models, but also to detect inconsistencies within models and to support the dialog between stakeholders.

As future works, we plan to improve the SmartSync platform, notably by developing new comparison algorithms and abstraction methods.

References

1. Batteux, M., Prosvirnova, T., A.Rauzy: From models of structures to structures of models. In: 4th IEEE International Symposium on Systems Engineering, ISSE 2018. Rome, Italy (October 2018)
2. Batteux, M., Prosvirnova, T., A.Rauzy: Altarica 3.0 in 10 modeling patterns. International Journal of Critical Computer-Based Systems (IJCCBS) **9**, 133 (2019). https://doi.org/10.1504/IJCCBS.2019.10020023
3. David, P., Idasiak, V., Kratz, F.: Reliability study of complex physical systems using sysml. Reliability Engineering & System Safety **95**(4), 431–450 (2010)
4. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language. Morgan Kaufmann. The MK/OMG Press, San Francisco, CA 94104, USA (2011)
5. Fritzson, P.: Principles of ObjectOriented Modeling and Simulation with Modelica 3.3: A CyberPhysical Approach. Wiley-IEEE Press, Hoboken, NJ 07030-5774, USA (2015)
6. Legendre, A., Lanusse, A., Rauzy, A.: Toward Model Synchronization Between Safety Analysis and System Architecture Design in Industrial Contexts. In: Marco Bozzano, Y.P. (ed.) Model-Based Safety and Assessment. vol. 10437, pp. 35–49. Springer (2017). https://doi.org/10.1007/978-3-319-64119-5_3, proceedings of the 5th International Symposium, IMBSA 2017, Trento, Italy, September 11–13, 2017
7. Mauborgne, P., Deniaud, S., Levrat, E., Bonjour, E., Micaëlli, J.P., Loise, D.: Operational and system hazard analysis in a safe systems requirement engineering process application to automotive industry. Safety Science **87**, 256–268 (August 2016)
8. Missaoui, S., Mhenni, F., Choley, J., Nguyen, N.: Verification and validation of the consistency between multi-domain system models. In: 2018 Annual IEEE International Systems Conference (SysCon). pp. 1–7 (April 2018). https://doi.org/10.1109/SYSCON.2018.8369561
9. Prosvirnova, T., Saez, E., Seguin, C., Virelizier, P.: Handling consistency between safety and system models. In: IMBSA 2017 (International Symposium on Model-Based and Assessment). pp. 19–34. Trento, Italy (2017). https://doi.org/10.1007/978-3-319-64119-5_2
10. Schamai, W., Fritzson, P., Paredis, C., Pop, A.: Towards unified system modeling and simulation with modelicaml: Modeling of executable behavior using graphical notations. In: Proceedings of the 7th International Modelica Conference (2009). https://doi.org/10.3384/ecp09430081
11. Stephan, M., Cordy, J.R.: A survey of model comparison approaches and applications. In: MODELSWARD 2013 - Proceedings of the 1st International Conference on Model-Driven Engineering and Software Development, Barcelona, Spain, 19 - 21 February, 2013. pp. 265–277 (2013). https://doi.org/10.5220/0004311102650277
12. Yakymets, N., Julho, Y.M., Lanusse, A.: Sophia framework for model-based safety analysis. In: Actes du congrès Lambda-Mu 19 (actes électroniques). Institut pour la Maîtrise des Risques, Dijon, France (October 2014)