Automated Model Network Configuration for the Simulative Evaluation of Electromechanical Drivetrain Concepts

Yasin, Abdullah[1]*, Irnich, Lukas[1]

[1] Institute of Machine Elements and Systems Engineering, RWTH Aachen University

** Korrespondierender Autor:*
Abdullah Yasin
Schinkelstr. 10, 52062 Aachen
✉ *abdullah.yasin@rwth-aachen.de*

Abstract

A variety of possible technical concepts can be considered for an electromechanical drivetrain. To evaluate them, for each a simulation of the physical behavior, lifetime or efficiency must be performed. However, the required modelling can be very time-consuming. This article describes a method to reduce manual modelling efforts, allowing the simulative evaluation of large technical solution spaces. Therefore, a modelling standardization for variant model networks is presented, which can be easily configured to simulate specific concepts and purposes. In addition, an approach for the automated identification, configuration and execution of concept- and purpose-related models is introduced. The method is implemented in MATLAB®, Simulink® and System Composer® and validated using the example of the electromechanical drivetrain of a passenger car.

Keywords

MBSE, Model Reuse, Model Configuration, Electromechanical Drivetrains, System Composer

 | DOI: 10.35199/dfx2024.22

1. Introduction

The electrification of mechanical drivetrains can significantly reduce the carbon emissions of vehicles and mobile machinery. In view of the EU's planned ban on the sale of new petrol and diesel vehicles from 2035 [1], a rapid switch to electromechanical drivetrains in various mobile applications is imminent. During the development process of electromechanical drivetrains, many different concepts can be created and investigated. These can differ in their functional architecture (e.g.: purely electrical energy storage or storage of hydrogen with subsequent conversions into electrical energy) or in the used technical solutions (e.g.: Synchronous Motor or Asynchronous Motor). A solution space with hundreds of possible concepts can be considered. Each of such concepts have own requirement-specific advantages and disadvantages regarding e.g. their physical behavior, lifetime or efficiency [2]. To objectively evaluate those concepts for specific requirements and to determine the most suitable one, the behavior of each individual concept must be modeled and simulated in a virtual environment. Modeling e.g. the physical behavior of just one single electrotechnical drivetrain concept, e.g. in Simulink® [3], already requires an enormous amount of manual modeling effort, time and cost. To model and simulate the physical behavior of hundreds of possible concepts, the modeling effort must be significantly reduced.

Two approaches can be used to reduce manual modelling efforts: First, the reuse of simulation models and second, an efficient integration strategy for simulation models.

By combining reused scope-specific (e.g. Asynchronous Motor, Lithium-Ion-Battery) and purpose-specific (e.g. functional-physical behavior, heat dissipation) simulation models, various drivetrain concepts can be created and simulated for required simulation results (resp. purposes) [4]. For each technical solution a variety of different simulation models exist regarding their purpose and scope as well as their input- and output parameters. Models for physical behavior simulation of technical solutions used in electromechanical drivetrains can be created in the tool Simulink®, e.g. using Simscape® Blocks. According to [5] Simulink® is currently the most suitable tool for the physical behavior simulation of electromechanical drivetrains. Simulink® models can be stored within a customized Block Library, reused and manually integrated into a drivetrain concepts physical behavior simulation model. If all models are available for a specific concept and simulation result, this approach of model reuse and integration can already reduce manual modelling efforts by a lot. For large solution spaces, the manual and still very time-consuming scope- and purpose-specific integration remains.

Exactly for this case the Model-Based Systems Engineering (MBSE) tool System Composer from MathWorks® offers an efficient model integration opportunity using Variant Components [6]. Thus, only one single variable model network containing various Simulink® models for all possible scopes and purposes needs to be created manually. This model network can be configured and run concept- and purpose-specifically by a minimum of effort or theoretically even fully automated.

For the configuration, however, suitable models must first be identified for a specific concept and a specific purpose. The large number of possible model combinations (e.g. for just ten possible concepts, each consisting of five technical solutions, each solution with four reused simulation models, already 10.240 model combinations can be configured) and their compatibility check (Is the required simulation purpose fulfilled? Do the model scopes match the required concept? Do the model interfaces and input and output parameter match?) make this still a complex and time-consuming task. Another challenge is that the implementation of model networks in System Composer currently lacks a standardization for the reuse of models (e.g. regarding model hierarchy, structuring and interfaces) to enable automatic configuration and execution of the identified models. Today, reused simulation models often must be modified manually before the model network can be configured and run within the System Composer.

In summary, although the manual modelling effort can already be reduced today, there is still a lack of a method to automatically identify suitable models in a model network for a specific concept and purpose, and a lack of standardization for modelling a network so that it can be automatically configured to run the identified and reused simulation models.

Therefore, the research aim of this paper is to develop a method to automatically identify, configure and execute suitable and reusable simulation models for a required concept and purpose in a model network consisting of reusable simulation models. This will enable the simulative evaluation of large solution spaces in electromechanical drivetrain development.

To achieve this goal, first the current state of research is discussed, and research questions are formulated. The developed method is then presented and validated using electromechanical drivetrain concepts of a passenger car and the physical behavior simulation for different purposes (e.g. just physical behavior, or additional efficiency estimation, or heat dissipation). The implementation is carried out in MATLAB®, Simulink® and the System Composer. A summary and conclusion of the results follows.

2. State of Research

To automatically identify and execute suitable simulation models for a required concept and purpose in a model network consisting of reusable simulation models, following two methods are needed. In this chapter we will discuss the current state of research regarding those two areas of methodology.

1. A standardization method to create a model network with reusable simulation models so that it can be configured automatically to run identified simulation models.

A number of MBSE modelling standardization methods, based on the Systems Modelling Language (SysML) address the development of CPS, e.g. SYSMOD [7], SPES [8], mecPro [9] and FAS4M [10]. All those modelling methods have in common, that electromechanical drivetrain concepts can be implemented and theoretically be linked to physical behavior simulation models. What these methods don´t cover is simulation model network standardization regarding its configuration or model reuse. The motego method [4], another SysML modelling standardization method, is focused on the seamless implementation and evaluation of electromechanical systems in a model-based and functional oriented manner. The approach offers a SysML-based standardization for model network implementation regarding its efficient configuration [11] and model reuse [4]. According to [4] and [11] model networks can be structured following functional architectures, concepts, their technical solutions and all their possible variations. Basic time-independent physical behavior models can be integrated into these solutions and parametrically combined via function-oriented interfaces (e.g. rotational mechanical energy) so that they can be simulated combined for the overall concept. The function-oriented structure allows efficient reuse of the individual simulation models. For each technical solution, all possible simulation models are implemented according to their formalized classification for scope and purpose [4]. In [11], this standardization is used to build a model network with reusable models, which can be configured efficiently, but still manually, to simulate different concepts. This approach is specifically developed for SysML model network standardization. Model network implementation according to the motego method can be carried out in e.g. CATIA Magic [12], a specific system modelling tool. The integration and simulation of time-dependent physical behavior simulation model is possible, but not efficient enough to compared with software tools such as Simulink®. In [13] a mechanism for the variable implementation of models in Simulink is presented, which is used for automatic code generation. In [14] an approach for the

implementation of component variants in Simulink was developed. Both approaches are inadequate for the configuration of model networks.

Current standardization methods for model networks do not yet allow automated configuration of reusable and sufficiently accurate models to simulate physical behavior. This represents the first research gap.

2. A method to automatically identify suitable simulation models in a model network for a specific concept and purpose.

Partially or even fully automated model identification methods are already presented in a few papers. Here the most relevant methods regarding a fully automated identification of suitable simulation models in a model network for a specific concept and purpose are discussed. In [15] a method for the automated selection of physical behavior models for functional architectures is presented for Modelica®. The method is based on the flows between functional blocks (e.g. rotational mechanical energy or electrical energy). The identification criteria are identical functional flows. However, with this method a model identification for a specific concept and simulation purpose is not supported. Furthermore, in comparison to Simulink®, only standardized bidirectional interfaces are used in Modelica®. This makes a compatibility check of two connected models obsolete. [16] developed a methodology to efficiently explore and evaluate possible concepts for hybrid electromechanical drivetrains based on the VEHLIB library [17] in Simulink®. However, for the simulation and evaluation of a specific concept and simulation purpose, suitable models must be selected and implemented manually. As part of the Montego method in [4] an approach for the usage of model signatures [18] is presented. A model signature contains the model input parameters, output parameters and internal parameters. Together with the model classification, model signatures can be used to analyze its suitability for a specific scope and purpose as well as it´s compatibility regarding available parameters. In [19] an approach for automated simulation model identification in a model network regarding their compatibility is presented. The method is based on the formalized model classification and signatures in SysML. However, the method does not consider any concept or purpose-specific identification. For possible variations, individual model networks must be implemented manually.

For model networks, there are not yet sufficient methods of automated model identification for specific concepts and purposes. This represents the second research gap.

3. Research Gaps and Questions

To enable concept evaluation based on physical behavior simulation for large solution spaces in electromechanical drivetrain development, manual modelling efforts must be reduced. Therefore, software tools such as the MATLAB® System Composer can potentially be used to create just one single variant network of reusable models that can be configured for concept and purpose-specific simulation and evaluation. However, suitable models currently still must be selected manually. This requires not only the identification of concept and purpose specific models, but also checking their compatibility. Due to the large number of possible model combinations resulting from the combinatorics of the variant network, this can be a very time-consuming task. According to the current state of the art, there is no sufficient method that can enable automated model identification and configuration for such model networks. For an automated configuration of the model network for the previously identified models, the current state of the art as well lacks a suitable standardization methodology of simulation models and their interfaces. This can lead to serious comparability problems, especially in the context of long-term model reusability. This research gap for automated model identification and configuration and its potential for MBSE was also identified and described in

[20]. The aim of this paper is therefore to develop a method to automatically identify, configure and run suitable and reusable simulation models for a specific concept and purpose in a model network. To achieve this goal, the following research questions are proposed:

1. How must the model network be standardized so that it can be configured automatically to run identified simulation models?
2. How can suitable simulation models be identified automatically in a model network for a specific required concept and simulation purpose?

The method is presented in Chapter 4 and validated in Chapter 5 using the exemplary concept development of the electromechanical drivetrain of a passenger car. MATLAB®, System Composer and Simulink® are used for the implementation of the methodology. A conclusion follows in Chapter 6.

4. Methodology

The method presented is divided into two subchapters. The first subchapter (Chapter 4.1) addresses the standardization of variable model networks to enable their automated configuration for individual simulation model combinations covering multiple concepts and simulation purposes. The second subchapter (Chapter 4.2) describes an approach for the automated identification and configuration of suitable simulation models for an individual concept and simulation purpose.

4.1. Model Network Standardization

Automated model network configuration to run identified models requires standardization of simulation model variations, ports, parameter naming, computer-readable model signatures, and model integration into the network. The standardization method is illustrated in Figure 1.

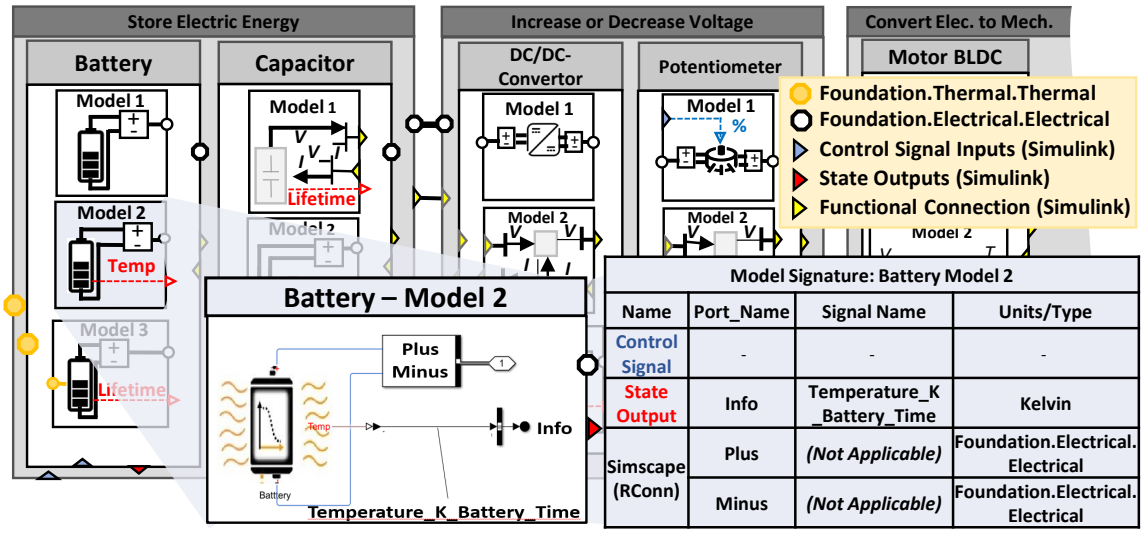

Figure 1: Model Network Standardization

We need to create model variations in the model network to enable its configuration. Therefore, a standardization for all variation opportunities (e.g. to configure the network to simulate different concepts behavior, or to run it regarding different simulation purposes) is needed. Variant simulation models can be implemented and nested in System Composer using Variant Components. We consider model variations in multiple hierarchical layers in the

network to enable the best possible reusability and configurability. Therefore, a first Variant Component in the highest layer – the function – contains variant technical solutions. For the function Energy Storage this can be a Battery or a Capacitator (see Figure 1). Within these technical solutions a second Variant Component is used to nest all its Simulink® simulation models. These can differ in terms of purpose or input-output-combinations (e.g. physical behavior or with additional lifetime estimation or heat dissipation) (see Figure 1).

Figure 1 shows the simulation model 2 of the battery. This model is implemented as a custom Simscape Block with a temperature estimation based on the heat generation from the battery efficiency and the specific heat capacity of its cells. The model uses a DC electrical Simscape connection and a Simulink output for the temperature. Other models are also implemented as custom Simscape Blocks, such as Battery Model 3, Capacitor Model 2 or Capacitor Model 3. For models that only have Simulink connections, the model is implemented as a simple calculation with Simulink Blocks. For example, model 2 of the DCDC converter contains simple gain blocks to decrease the voltage value and increase the current value.

A standardization of the simulation model interfaces and ports is needed. All simulation models nested in a Variant Component need identical interfaces and ports. Regarding the hierarchical structure the functional layer specifies this standardization for all lower-level simulation models. In addition, all functions and models must use the same interfaces and ports so that they can be connected. These functional interfaces must be created for possible Simulink® (bidirectional) as well as for Simscape® models. In CPS each function, technical solution and model can interact with the system and the surrounding environment in terms of energy flows, material flows, and control signal flows. For each function, possible interactions are limited. For example, a battery supplies DC electrical energy and may interact thermally with the environment or with energy management system. It is typically the case that a battery will not engage in any material interactions of functional significance. Consequently, the battery is equipped with ports for both DC electrical and thermal energy flows (see Figure 1). Since material and energy flows are conserved, Simscape® connections can also be used by models in addition to Simulink® directional ports. Figure 1 shows a functional block of Electrical Energy Storage function with a Battery and Capacitor as alternate effects. Each functional block is given a Control Signal Simulink® input port because some models may have state dependent or control signal dependent behavior. The input port for control signals has always a directional Simulink® type because control signals do not have conservative and bidirectional nature as energy or material. Lastly, a performance output port is included for state signals which are used for the feedback loop with the control system. CPS such as hybrid drivetrains included feedback-based control software for functional and safety performance of the system. In addition to the feedback loop, the elements of this ports also provide purpose-specific information of the model which would later be used to identify suitable models. Purpose-specific signal outputs for requirement validation is a common practice according to [21].

Table 1: Example of naming structure of inputs and outputs of the simulation models

Component	Quantity Name	SI Dimension	Scope	Dependencies
Battery	Temperature	Q	Battery	Time
	Temperature	Q	Anode	Time_xyz
	Voltage	I_1L2MT_3	Battery	Temperature_Time
DC Converter	Heat	L2MT_2	DCDC Converter	Time
Motor BLDC	Torque	L2M2T_2	Motor1	Time
…	…	…	…	…

For the suitable connection of model parameter inputs and outputs, their naming must be the same. Therefore, a standardized parameter naming is necessary. Within this approach the

following standard naming structure is used: *QuantityName_SIDimension_Scope_Dependancies*. The quantity name serves to identify the state variable or the performance metric. This enables the requirement-specific model selection. Next part of the naming convention is the SI dimension and scope. The last part are dependencies which give information about the fidelity of the output such as time dependency, or a spatial resolution (indicated by “xyz”) is present. The naming structure of elements also serves the automated identification, later. Some examples of the naming are listed in Table 1.

Because the compatibility and the model configuration depends on the specific parameter inputs and output combination, model signatures are generated. Model signatures are formal description of parameter inputs and outputs of the model. Thus, model signatures can be useful in evaluating the compatibility of models. A script is developed which generates tabular model signature in terms of port names, designated positions (input, output, Simscape® ports), special structured name, and unit. This information is stored in tables which are utilized later in the filtering and compatibility of model identification. An exemplary, automatically generated model signature is shown in Figure 1 for Battery model 2.

Multiple Functions can now be referenced and connected in a functional architecture of an electromechanical drivetrain within the System Composer. This functional architecture represents our model network. With the described standardization for model networks, the overall concept can be configured by manually selecting a technical concept for each function. This can be achieved by making a particular technical solution active. Afterwards, the simulation models for each active technical solution can also be configured by making it active. The model network configured in this way for a specific concept and a specific purpose can be parameterized, executed and simulated immediately. The configuration can be done manually or automated within a MATLAB® script.

4.2. Automated Model Identification and Configuration

In this subchapter an approach to enable the automated identification and configuration of suitable models in a model network for a specific concept and with a specific purpose is presented. The model identification approach consists of three steps, the *User Input* for the required concept and purpose, the *Model Filter*, to filter out all not suitable models regarding their scope and purpose and the *Compatibility Check*, where remaining and functionally connected models are checked for their parameter input output compatibility. In the additional forth step *Configure and Run Simulation* the model network is configurated and run automatically according to the identified models. In Figure 2 an overview of the process is given. The presented method is implemented in MATLAB®.

For the configuration of a specific concept and purpose, a user input is needed. The user input is based on a MATLAB® list, containing all required technical solutions of the concept (resp. model scopes) together with the required simulation purpose of each solutions model. Figure 3 illustrates two exemplary user input sets. In one case, a Brushed DC Motor, in the second case, a Brushless DC Motor, should be used in the concept. Additionally, to the physical behavior in the first set the battery temperature should be simulated, in the second case the lifetime. The user inputs are implemented with a similar naming structure as described for the naming convention of signals. Requirements can be implemented in the MATLAB® Requirements Manager and used as input also. Natural language requirement can even be formalized for such a naming standardization automatically according to [22].

In the second step, the architecture of the model network is read out. This results in a MATLAB® table containing all functions of the functional architecture, each technical solution and all possible simulation models. The formalized signatures are called based on the simulation model names. The model scope and model purpose are then compared with those of the user input. The usuitable models, that do not fulfill the required model scopes and purposes are deleted from the previously created table and relevant model is identified (see

Figure 2) e.g From user input, battery scope input eliminates capacitor and temperature purpose input filters model 1 in battery as relevant model. The temperature purpose input is compared with all model signatures of battery technical concept till relevant model is identified. This model filter can significantly reduce the amount of possible model combinations and therefore reduce the computational time of the implemented algorithm.

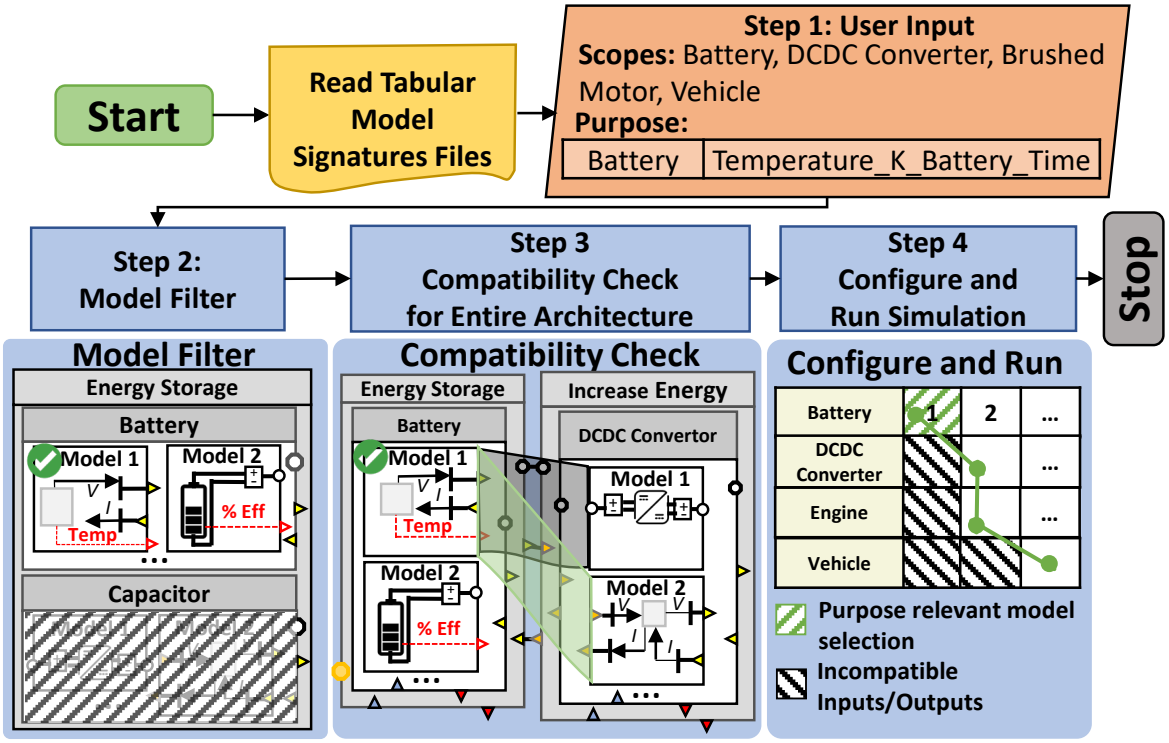

Figure 2: Steps in Identification of purpose-specific and compatible model identification

In the third step, the input-output-parameter compatibility is checked. Compatibility is necessary to ensure the executability of the identified models. For example, the simulation model of the battery may have voltage as a Simulink® output and current as input, but the connected converter model has a bidirectional electric Simscape® connection. The Compatibility Check is shown as step 3 in Figure 2. In step 2, models are already identified in one (Figure 2 Battery Model 1) or more technical concepts (see Figure 3 Output Set 2 where battery and motor models are identified in step 2). From identified models (Energy Etorage in Figure 2), the models of adjacent functional blocks (Energy Level Change in Figure 2) are identified. By comparing the model ports and formalized signal name in the model signatures, compatibility can be checked. For example, input/output ports and formalized signal names of all model signatures of DCDC Converter technical solution are compared with those of battery model 1. Once model signatures match, the corresponding model is identified e.g Model 2 in DCDC Converter. The process ends when a compatible model has been identified for each technical solution in entire architecture. The result is the identification of models that are suitable for the simulation of the required concept and the required simulation purpose.

Once the simulation models have been identified, the model network is configured automatically in step four by making these models active in the model network using software code. Lastly, the simulation is executed.

This approach and based on subchapter 4.1 suitable models can be automatically identified, configurated and run in a model network for a specific concept and with a specific purpose.

5. Validation

For the validation of the methodology two sets of user inputs were used (see Figure 3). In set one, e.g. a brushed motor and a battery are selected, both for the simulation of their physical behavior. In addition, the batteries lifetime must be calculated. In the second set a brushless DC motor is selected together with the battery and DCDC Converter. In this case the batteries temperature must be simulated additionally. In both cases, suitable models of the model network shown in Figure 3 can be automatically identified, configured and executed. The simulation results generated in this way are also shown in Figure 3. For the validation, a model network consisting of 28 simulation models were created.

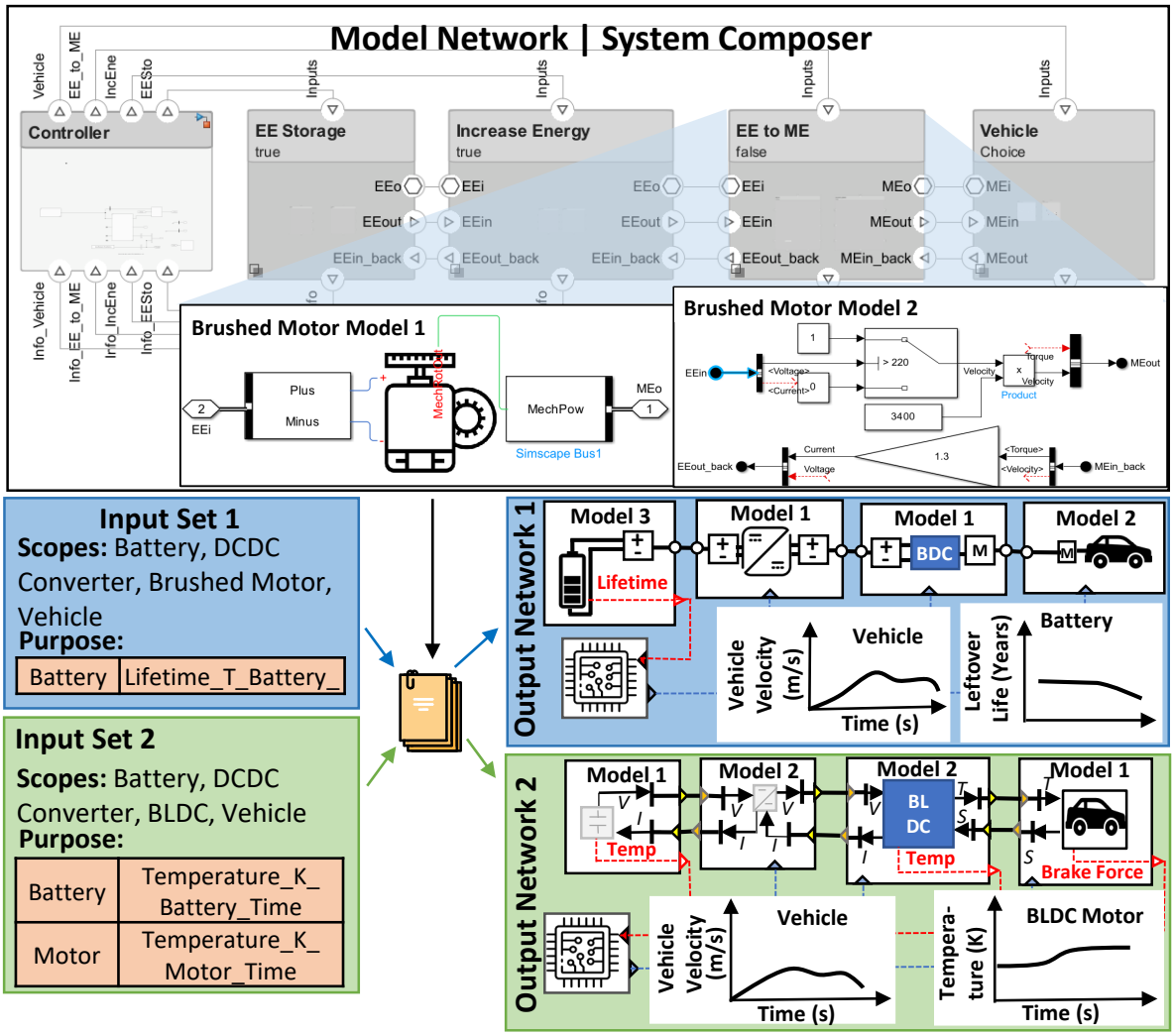

Figure 3: Validation of the Method

6. Conclusion

Electromechanical drivetrains are complex CPS with the potential to reduce vehicles CO_2 emissions. Different concepts and requirements lead to different performances. Thus, alternative concepts must be simulated and evaluated to identify the best suitable concept. Therefore, manual modelling efforts must be reduced significantly. In this paper a method was presented to automatically identify, configure and run suitable simulation models for a specific concept and purpose in a model network. The methods consist of a standardization for configurable model networks and an approach to identify, configure and run suitable simulation automatically. The methodology was validated, by implementing a set of 28 simulation models for multiple technical sub-solutions of electromechanical drivetrains and purposes related to physical behavior in a single configurable model network. Using the implemented

methodologies and two separate sets of user inputs for required concepts and purposes for a specific electromechanical drivetrain an automated model identification, configuration and simulation was performed.

Applied in practice, the method can be used to implement a reusable model library for the simulation of physical behavior for electromechanical drivetrains. The library can be used to generate a model network that can be automatically configured for individual concepts and purposes. The method thus enables simple and fast exploration of technical solution spaces with objective evaluation based on simulation results.

List of References

[1] European Parliament: EU ban on the sale of new petrol and diesel cars from 2035 explained. 03.11.2022

[2] Walters, Jim; Husted, Harry; Rajashekara, Kaushik: Comparative Study of Hybrid Powertrain Strategies. In: SAE Transactions 110 (2001) doi: 10.4271/2001-01-2501

[3] Simulink® : MathWorks®. URL https://de.mathworks.com/products/simulink.html

[4] Jacobs, Georg et al.: Design Methodology for Future Products: Data Driven, Agile and Flexible: Springer, 2022 doi: 10.1007/978-3-030-78368-6

[5] Tammi, Kari; Minav, Tatiana; Koratalein, Juha: Thirty Years of Electro-Hybrid Powertrain Simulation. In: IEEE access 6 (2018) doi: 10.1109/ACCESS.2018.2850916

[6] System Composer® : MathWorks®. URL https://de.mathworks.com/products/system-composer.html

[7] Weilkiens, Tim: SYSMOD - the systems modeling toolbox : Pragmatic MBSE with SysML. 3rd edition, version 4.2. Fredesdorf: MBSE4U, 2020 (MBSE4U booklet series)

[8] Pohl, Klaus et al. Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology: Springer Berlin Heidelberg, 2012 doi: 10.1007/978-3-642-34614-9

[9] Eigner, Martin et al.: mecPro2-Entwurf einer Beschreibungssystematik zur Entwicklung cybertronischer Systeme mit SysML. In: Tag des Systems Engineering. Hanser, München, S (2015), S. 163–172

[10] Moeser, Georg et al.: Modellbasierter mechanischer Konzeptentwurf: Ergebnisse des FAS4M-Projektes. In: Tag des Systems Engineering (2016), S. 419–428

[11] Irnich, Lukas et al.: Combining and evaluating function-oriented solutions in model-based systems engineering. In: Forschung im Ingenieurwesen 87 (2023), Nr. 1, S. 375–386 doi: 10.1007/s10010-023-00619-0

[12] Dassault Systèmes: CATIA Magic: CATIA. URL https://www.3ds.com/de/produkte-und-services/catia/produkte/catia-magic/

[13] Leitner, Andrea; Ebner, Wolfgang; Kreiner, Christian: Mechanisms to Handle Structural Variability in MATLAB/Simulink Models. In: Favaro, John; Morisio, Maurizio (Hrsg.): Safe and Secure Software Reuse. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, S. 17–31 doi: 10.1007/978-3-642-38977-1_2

[14] Haber, Arne et al.: First-class variability modeling in matlab/simulink. In: Proceedings of the 7th international workshop on variability modelling of software-intensive systems, S. 1–8 doi: 10.48550/arXiv.1408.5751

[15] Canedo, Arquimedes; Schwarzenbach, Eric; Abdullah Al Faruque, Mohammad: Context-sensitive synthesis of executable functional models of cyber-physical systems (2013) doi: 10.1145/2502524.2502539

[16] Kabalan, Bilal et al.: Systematic Methodology for Architecture Generation and Design Optimization of Hybrid Powertrains. In: IEEE Transactions on Vehicular Technology 69 (2020), Nr. 12, S. 14846–14857 doi: 10.1109/TVT.2020.3041501

[17] Vinot, Emmanuel et al.: Model simulation, validation and case study of the 2004 THS of Toyota Prius. In: International Journal of Vehicle Systems Modelling and Testing 3 (2008), Nr. 3, S. 139–167 doi: 10.1504/IJVSMT.2008.023835

[18] Zerwas, Thilo et al.: Model Signatures for the Integration of Simulation Models into System Models. In: Systems 10 (2022), Nr. 6, S. 199 doi: 10.3390/systems10060199

[19] Berges, Julius Moritz et al.: Automated Identification of Valid Model Networks Using Model-Based Systems Engineering. In: Systems 10 (2022), Nr. 6, S. 250 doi: 10.3390/systems10060250

[20] Wilkings, Fabian et al.: Utilization of system models in model-based systems engineering: definition, classes and research directions based on a systematic literature review. In: Design Science 10 (2024), Nr. 6 doi: doi.org/10.1017/dsj.2024.3

[21] Salado Diez, Alejandro; Wach, Paul: Constructing True Model-Based Requirements in SysML (2019) doi: 10.3390/systems7020019

[22] Salado, Alejandro: Handbook of Model-Based Systems Engineering: Springer, 2023 doi: 10.1007/978-3-030-93582-5