Modular Simulation Application for Designing and Analyzing Electric Vehicles

Danquah Benedikt, Koch Alexander, Weiβ Tony, Lienkamp Markus
Technical University of Munich, Chair of Automotive Technology
Boltzmannstraße 15
85748 Garching, Germany
Email: danquah@ftm.mw.tum.de

Pinnel André
TESIS GmbH
Baierbrunnerstr. 23
81379 Munich, Germany
Email: a.pinnel@tesis.de

Abstract—The automotive development process is highly complex and is getting more diverse with increasing possibilities in modeling, computing and analyzing. Especially in the sector of electric vehicles exists a rising amount of new simulation methods dealing with areas like propulsion systems, power supply and thermal management. To combine the knowledge from those areas in one comprehensive model, this paper develops an approach for a modular, library based system simulation. The aim is to create a defined workspace by specifying boundary conditions and introducing components for each part of the simulation. The core of the approach is the definition of the elements each component must contain and which interfaces they must include to facilitate complete system analyses. This paper serves as a guideline for modular entire vehicle simulations especially when used as open source or free software. For the proof of concept an essential part of the paper is an entire vehicle model including a library with exchangeable components, which will be additionally published as open source. For a plausibility check the results are compared to premeasured test data of a prototype electric vehicle.

Keywords—entire vehicle simulation, modular simulation, vehicle component library, open source model

I. INTRODUCTION

The automotive development process is highly complex and is getting more diverse with increasing possibilities in modeling, computing and analyzing real systems [1] [2]. Especially the beginning of the concept phase proposes a lot design freedom for the fulfillment of vehicle requirements. In order to compare the possibilities, it is important to estimate the properties of the final product as early as possible [3]. For this reason, in the area of concept development, system simulation is an essential tool that is of great importance for the quality of the research and development work. Using different software and having many opportunities in modeling, results in a rapid growing number of simulation tools [4]. Especially in the sector of electric vehicles, there is a rising amount of new simulation and system identification methods, which engage in areas like propulsion systems, power supply, electric machine design and thermal management from different angles [5].

Both academia and industry need to reuse and extend already existing models to increase synergy effects. To combine the knowledge from the mentioned areas in one complete model, this paper develops an approach for a modular, library based system simulation. The aim of this paper is to gain modularity by making parts of the simulation exchangeable and reusable. This is achieved by introducing components, workspaces and boundary conditions. Conventional simulation techniques, however, are not sufficient because of their lack of a module specific structure especially when they use external calculations [6, pp. 44-45]. As a solution, conventional techniques and basic ideas of object oriented programming are combined to create a modular entire vehicle simulation. The core of the approach is the definition of elements and interfaces each component must contain for a precise collaboration. This paper serves as a guideline for modular vehicle simulations especially when used as open source or free software. It provides the fundamentals of a research and development process as a community.

For the proof of concept a sample model including a component library is implemented with a real prototype electric vehicle, which will be published as an open source project. Starting from the basic longitudinal simulation model, it is possible to exchange and rearrange components individually, to analyze and identify entire vehicle dependencies. This modular model creates an overall test environment for entire vehicle simulations. Its approach is not restricted to electric vehicles. It can be extended to most power train topologies and used in other fields of engineering.
II. STATE OF THE ART

Before developing a modular simulation model, the boundary conditions must be defined. They specify under which circumstances this approach can be used and for which kind of simulation it is most suitable. For this question to answer simulations are set into an overall context. Figure 1 concludes commonly used simulation types. According to Zimmer [7, p. 35] there are virtual, digital simulations. Langermann [8, pp. 5-6] defines that those simulations can be geometry or functional oriented. This paper focuses on the second ones where the functional correlations of a system are considered. According to Brychta [9, p. 55] functional simulations can be either mathematically, signal flow or symbolic oriented. This approach is applicable to all of them however, this paper will predominantly focus on signal based simulations. Typical software in the field of entire vehicle simulations using functional based approaches are Matlab/Simulink and DYMOLA/Modelica [6, p. 44], [9, pp. 55], [10].

![Simulation Types Diagram](image)

In the field of longitudinal vehicle simulations, there are backward and forward simulations. Guzzella [6, pp. 37-41] describes that in a backward simulation, the required driving force of the vehicle is calculated by its driving resistance. With this force the state of the propulsion system is calculated backwards. In contrast figure 2 shows a forward simulation, which represents the behavior of the vehicle more realistically [6, p. 40].

![Vehicle Simulation Diagram](image)

The forward simulation needs a feedback controller to set the power of the propulsion system. Starting on the left side the state of the electrical system is calculated first. Secondly the mechanical part and the vehicle dynamics are estimated. Combinations of backward and forward simulations are possible.

Zimmer [7, pp. 58] separates the simulation process in the three parts preprocessing, mainprocessing and postprocessing which are illustrated in figure 3. The user input takes place during the preprocessing. In this phase, for example, parameter inputs are made or the driving cycle is selected. It also includes the initialization routine where the simulation setup is initialized by selecting the solver and the sample times. Missing parameters are calculated which can be performed with calculation tools. Calculation tools are automated system identification methods such as parameter estimations for electric motors, batteries or other components. The mainprocessing contains the simulation routine. The postprocessing includes the result evaluation and the visualization.

![Preprocessing, Mainprocessing, Postprocessing Diagram](image)

Several system simulations exist to support the vehicle development process. Modularity is important to calculate properties of different vehicles. Tschochner [11] and Nehmet [12] developed longitudinal entire vehicle simulations. For the simulation to be modular both separate the simulation in several components which are described in detail from the mathematical point of view. One challenge in a simulation is the parameterization of the model. To estimate missing parameters from basic inputs calculation tools are developed. For example Horlbeck [13] developed an analytic efficiency determination of electric motors and Tschochner [11] designed an interpolation method for efficiency maps.

There is an increasing amount of simulations, component models and calculation tools which are available for the community. It is important to make them reusable and to combine all of these. The mentioned approaches however, primarily consider the physical and mathematical correctness of the model. The connectivity and the reusability in a community is not focused. There is no
standard when using a model in the community. This paper makes a statement how the structure of a simulation and its components should be assembled when used in a large area or published as open source. The paper provides the fundamentals for efficient collaboration in a wide range that will support the research and development process in a community. It shows how to conclude several components and tools in one framework.

III. MODULARIZATION APPROACH

This paper presents the developed modularization approach for simulation models which should serve as a guideline for open source models. The approach provides the fundamentals for efficient collaboration in a large range. The aim is to gain modularity by making parts of the simulation exchangeable and reusable by introducing separable submodels for each component. One component does not only consist of its simulation model but also of its parameters, calculation tools and initialization commands. Commonly used initialization commands are variable definitions. As the state of the art shows, these component models and tools get more sophisticated and gain more importance in the research and development process of electric vehicles.

To understand the challenges of modular simulations figure 4 exemplary shows two motors which are used for longitudinal forward simulations. They have different parameter sets, initialization routines and estimation tools. Both focus on an efficiency simulation. Motor one is commonly used and needs the parameter set 1 and an initialization routine. Since the single efficiency is given by a parameter no specific calculation tool is needed. Horlbeck [13] in contrast is using a motor specialized on a comprehensive efficiency calculation. It needs the more detailed parameter set 2 and is using the motor geometry, power data and materials to calculate its efficiency map. This example shows that initialization commands, calculation tools and parameters are different depending on which component the model uses. The routines are mostly calculated externally in variable locations. When a component is exchanged (eg. component one with component two) in the simulation model, also the parameters initialization commands and calculations tools have to be merged. This leads to low flexibility and has a high error potential because the changes can lead to inconsistencies.

As a solution the parameters and component specific preprocessing routines must be directly attached to the component. So the routines will automatically change with the component and be updated for each simulation. To merge the model with its parameters, initialization commands and calculation tools, both conventional simulation and object oriented programming techniques will be combined. The basics of object oriented programming establish the fundamental requirements of this approach. The components of the simulation must own the following properties:

- Exchangeability,
- Reusability,
- Encapsulation.

Encapsulation means that the information and variables calculated in one component are hidden. The components communicate exclusively with predefined interfaces. This guarantees that components do not interfere with each other, which means that they don’t overwrite variables or confuse same named functions. A defined workspace must be created by specifying boundary conditions for each component. To do so this paper shows the structure of a component to be reusable, exchangeable and encapsulated. Further more a possible implementation will be described exemplary by creating a component library.

IV. COMPONENT STRUCTURE

To achieve the needed requirements of a modular simulation, the concept is to separate the simulation model in parts by introducing submodels. Those parts are called components. This section explains the developed structure of a component. As figure 5 shows, one component consist of its interfaces its system boundary and its component workspace. These parts are now explained in detail:

System Boundary: The system boundary separates the component from its environment and guarantees its encapsulation. Consequently the variables, parameters and calculation tools have their own space. The system boundary has to ensure that the component can exclusively exchange information via its predefined interfaces. The challenge of a modularization is where to draw
the line of the system boundary. A decision has to be made which calculation operations to conclude in one component. To find a suit system boundary a tradeoff has do be found. For the highest modularity a component should represent the smallest increment of the simulation possible. For less implementation effort generalized components are preferred. Because the component will be an atomic instance of the simulation, a suitable compromise regarding its size must be chosen.

The encapsulated workspace contains all primary parameters defined by the user and calculates secondary parameters. The initialization can include sophisticated tools. For example it is used to calculate efficiency maps for motors. The simulation routine is part of the main processing and is called, once the simulation starts. It contains the simulation model and has access to all variables defined by the initialization routine. A simulation model for example is a signal based simulink model of an electric motor. The encapsulated workspace contains all primary parameters, secondary parameters and simulation signals. Since there is a separate encapsulated workspace, components can work independently and will not interfere with the rest of the model because each model can have its own variable definitions. A double assignment or the overwriting of variables with a multiple use of the same component is thus excluded. If one component is changed in the model also the initialization routines are changed accordingly because they are attached. As a result, the component can be used in a wide variety of applications. Due to its defined interfaces, it can be applied independently of the overall model.

Interfaces: Because an entirely closed system is not suitable, each component has two types of predefined interfaces to communicate with its environment. One interface is used for the exchange of parameters during the preprocessing and one for the simulation to exchange signals. There is one input interface for primary parameters. They are defined externally, loaded into the component and are used by the initialization routine. The output interface for secondary parameters contains the processed input parameters, the initialization results and can be utilized by other components. Parameters should be used for time independent variables. The simulation routine uses the in- and output interfaces of the component to exchange signals during the simulation. Those interfaces are used in common simulations and are intended for time dependent variables.

Including a defined system boundary, initialization and simulation routines and this set of interfaces, the components are exchangeable and encapsulated. The simulation process starts with the preprocessing which involves the initialization of each component. Since all components have access to each others parameters a flexible initialization can be executed. For example a suit gear can be calculated using the maximum motor torque as input parameter. This leads to a modular simulation and is a powerful and flexible setup.

V. IMPLEMENTATION OF A COMPONENT LIBRARY

After the modularization approach is explained an example library is created. The aim is to show the functionality and how to apply the approach in practice. For that, an example set of components of an electric vehicle is created in Matlab/Simulink. To maximize the reusability, those components are summarized in a library. It is including necessary parts of the power train as well as an acceleration calculation and components like driving strategies or control units. Those components can be assembled to an entire vehicle simulation model. The implementation shows how to achieve the requirements of reusability, exchangeability and encapsulation.

A. Component Design

System Boundary: The system boundary of a component is created by establishing a masked Simulink submodel shown in figure 6. The result is an encapsulated component workspace, which includes the Simulink...
model and stores all necessary variables, parameters and signals. The subsystem contains the Simulink simulation model which can be developed at this instance. The initialization commands and tools are attached and executed in the component workspace. The tools are implemented by calling them in the initialization callback of the component, which is located in the mask settings. So all common Matlab functions can be executed during the initialization routine for calculating all secondary parameters. The functions needed for the initialization are stored in an external data folder which is attached to the component.

Fig. 6. Component Implementation

**Interfaces:** An interface for signals is required to ensure the data exchange between the components during the simulation. They are implemented with a solution, Simulink already provides. With the `in` and `out` blocks signals can be exchanged between components. The interface is mainly intended for time-varying states of the component during the simulation. In contrast to the signals, parameters do not change with time. It must be determined in advance which properties of the component are specified as a signal and which as parameters. To establish the interface for the exchange of parameters, the masking method of Simulink is used. In the mask the input parameters are defined. There the primary input parameters can be inserted which is executed manually or programmatically. This way the parameters are assigned to the component directly. Once they are inserted they will be available in the component workspace. They can be used for the calculation of further parameters as well as for the simulation and are encapsulated from other components and are independent from the Matlab/Simulink base workspace.

The interface for secondary parameters is created by giving reading access to the entire workspace of the component. This is realized by a Matlab function which can either load the whole workspace or selected variables. This way, all components are able to import data from other components and use them for their own calculations. Thus a high flexibility and an automated preprocessing of components are achieved. By accessing other components, additional networking is possible, which could be necessary for the calculation of an driving strategy.

**B. Concluding to a Library**

For a better usability and easier administration the set of component models is concluded in one library. The advantage is that all components can be centrally managed. To increase the reusability, the components are sorted in a uniform and clear structure. For a better comprehensibility of the library and to avoid misunderstanding, each component has its own documentation. To make the documentation more effective it is attached to the simulation model as near as possible. For this purpose the documentation is attached to the mask. Further more, the components have additional information about their metadata:

- Author, date, predecessor model,
- Description of the model, Simulink Version,
- Time resolution,
- Scope of validity of inputs and outputs,
- Name and type of in- and outputs,
- parameter list
- application area
- Valid parameterization range,
- Sources used for development,

This library includes all parts needed for a longitudinal simulation and can be extended by adding components.

**VI. IMPLEMENTATION OF A SIMULATION MODEL**

For the proof of concept an essential part of the paper is an entire vehicle model including a library with exchangeable components. This section shows that the structure of the components and the library can be practicably used for modular simulations. Based on the library an entire vehicle model for longitudinal forward simulations is created in Matlab/Simulink. It uses a serial bus according to Rohr [14, pp. 37-42] to achieve a controlled simulation signal flow. Since the bus is built up successively, this structure offers the advantage that it is documented when each signal was last calculated. Figure 7 shows the basic setup of the implemented model.

The longitudinal dynamics model is divided into several bus blocks. The figure shows the most important
main blocks and the Timer and Bus Reset block. These blocks use the library components. In the Driving Cycle block, the target velocity data are loaded. The block called Driver and Environment contains the PI controller and the environmental states. The third block, Control Unit, converts the throttle and brake pedal positions of the controller into nominal torques. In Drive Train the target torques are converted into an actual drive and braking force. They are calculated by simulating the motor, the transmission, the brakes and the tires. Further more the consumption is calculated. The resistance forces are determined in the Dynamics bus block. The drive and braking forces of the power train result in an acceleration, speed and position of the vehicle.

For a plausibility check the model is parametrized by a prototype electric vehicle based on a Smart Fortwo which is described in [15], [16]. For the parametrization individual measurements were executed beforehand [17] [18]. To run the simulation, during the preprocessing the parameters have to be added to the components and the test setup has to be chosen (eg. driving cycle or environment). Then the components are initialized and the simulation can start. After the simulation postprocessing is done.

VII. TEST AND RESULTS

The simulation results are compared to premeasured and validated data of Wacker [17]. They contain the energy consumption and vehicle data of the adapted driving cycles WLTC, NEDC, and Artemis. During the experiment the original cycles could not be executed because the power of the vehicle was to low. To show that the simulation model gives reasonable results, the energy consumption of the test and the simulation is compared. For an accurate analysis the target velocities of the test data are inserted in the simulations. This reconstructs the experiment. Each simulation guarantees that the root mean square error (RMSE) between the target and the actual velocity is lower than 0.2 m/s. The simulation results must be accurate to be conclusive and usable. It is required that the deviation of the consumption between simulation and measurement is lower than 5% which is a commonly used value.

As shown in Figure 8 the vehicle follows the target velocity of the WLTC Class 2 cycle. The RMSE of this experiment is 0.055 m/s. The relative deviation of the overall consumption between the measurement and the simulation is 1.04%. Additionally the simulations were executed for other cycles. The deviations of the consumption between the measurement and the simulation are listed in Table I.

<table>
<thead>
<tr>
<th>TABLE I. CONSUMPTION WLTC, NEDC AND ARTEMIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit</td>
</tr>
<tr>
<td>Measurement kWh</td>
</tr>
<tr>
<td>Simulation kWh</td>
</tr>
<tr>
<td>Deviation kWh</td>
</tr>
<tr>
<td>Relative Deviation</td>
</tr>
</tbody>
</table>

This table shows, that all deviations of the consumption are below 5%. Consequently the simulation can be considered as accurate.

VIII. SUMMARY AND CONCLUSION

An approach for a modular system simulation is generated in this paper. The aim is to gain modularity
by making parts of the simulation exchangeable and reusable by introducing independent components for each part of the simulation. The main idea is to combine preprocessing routines like parameter calculation tools and the simulation model in one component. Thanks to defined boundary conditions, encapsulation and uniform interfaces a concept is developed with which different vehicle models and topologies can be created. The interfaces allow components to exchange parameters as well as signals. For the proof of concept a library is created containing all relevant components for longitudinal entire vehicle simulations. The library is used to create a sample modular simulation model. The model is parameterized with a parameter set of a real prototype vehicle simulated with three different driving cycles. The comparison of the results show that the deviation of the consumption is lower than 5% in each case.

The approach allows to analyze and enhance electric vehicles on a global level but is not restricted to them. It enables flexibility in creating suitable simulations depending on individual needs. This simulation setup can serve as a guideline for modular vehicle simulations particularly when they are used as open source or free software. The paper provides the fundamentals for efficient collaboration in a large range that will support the research and development process as a community. The library and the simulation model is published additionally as open source [19].

Contribution

As first author and initiator B. D. developed the general concept and wrote a large part of the paper. B.D and A.K created the structure of the components, and the component library. B. D., A.K. and A. P. implemented the component library and the developed simulation structure. B.D and T. W. created the simulation model and made the comparison the real data. M.L. made an essential contribution to the conception of the research project and revised the paper critically for important intellectual content. M.L. gave final approval of the version to be published and agrees to all aspects of the work. As a guarantor, M.L. accepts responsibility for the overall integrity of the paper.

Acknowledgment

This work was supported by the Bavarian Research Foundation VDI VDE with the research project sim2gether. Many thanks to the designers of the thenoun-project who supported with icons for the used figures.

References