Parametrization of *Modelica* Models on PC and Real time platforms

Matthias Kellner Martin Neumann Alexander Banerjee Pritesh Doshi
ZF Friedrichshafen AG
Graf-von-Soden-Platz 1, D-88046 Friedrichshafen, Germany
matthias.kellner@zf.com martin.neumann@zf.com alexander.banerjee@zf.com

Keywords: model based development, dynamic model parametrization, SW-Tests, ZBF-Parameter, Realtime

1 Introduction

Throughout the development process of control units for new transmission system, computer models are needed to perform different tasks, such as concept evaluation and the design and testing of controllers in MiL, HiL and SiL environments. These models will be used within different CAE-tools and different environments. To avoid redundancies and sources of errors the parametrization of these models using the same set of parameters is preferred, since these will change during the development process. Furthermore, the parameters will be kept within at one location. The only possibility to deal with this problem is to keep models and parameters separated, which means that models have to be parameterized using a set of files. Unfortunately, some environments do not allow file I/O operations. Even though no file I/O operations are available, for example on Real time platforms, there is still a strong need for flexible parametrization. Several approaches have been developed to overcome these challenges, especially for Dymola/Modelica models which will be presented within this paper.

In the following chapter the integrated use of vehicle models within ZF electronics development will be introduced. In chapters 3 and 4 the ZBF-parameter format will be discussed as well as the different parametrization approaches. With the help of an example the use of one of the approaches will be illustrated. Finally the results will be summarized and open questions will be addressed.

2 Integrated use of powertrain models within ZF electronics developments

Within this chapter the use of simulation models within the ZF electronics development will be illustrated, concentrating on passenger car applications. The focus will be an integrated use of these models from specification phase up to series application. The use of Dymola models within Hardware in the Loop (HiL)-simulation requires some adaptations within Dymola for parametrizing models on real time platforms.

Rising demands for better comfort, more power and lower fuel consumption as well as increased integration of different systems lead to a higher weighting on software developement. Although the development period has to decrease, the quality of the software and the level of customer satisfaction have to be continuously improved. For validation and verification purposes almost 40% of the total budget for software development has to be invested [1].

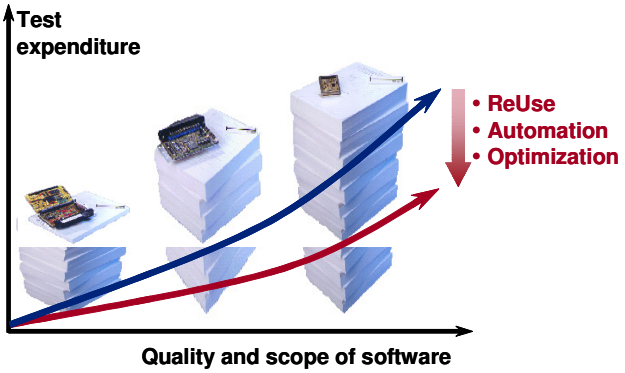

Figure 1: Test Expenditure of Transmission Software

As can be seen in Figure 1 the necessary testing expenditures increase disproportionately due to higher quality requirements. These ever increasing demands, which are closely linked to costs, can only be taken care of by improving efficiency. The application of ReUse, automation and optimization of testing processes can reduce the testing expenditures up to about 30% [2].

In order to reduce the time used for developing a new product the parallelization and decoupling of mechanic, electronic and software development is needed. By reusing models within different testing environments the service expenditures can be reduced. Testing at an early stage in development will also lead to a decrease in development expenditures.

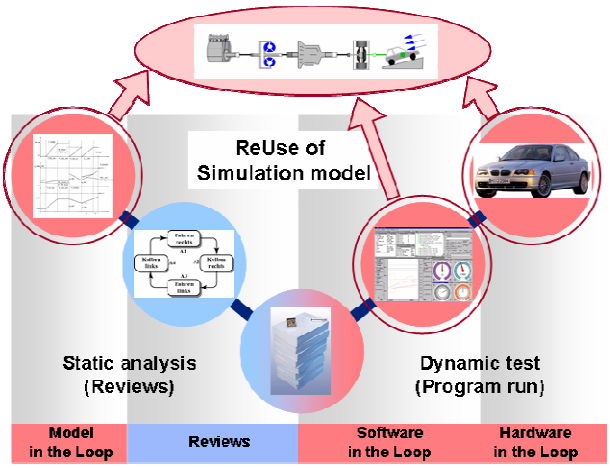

Figure 2: Testing as part of the software development process

To ensure a development of software in parallel and independently, appropriate environments for development and testing are needed. These should also include simulation capabilities. In order to service the software development process, different simulation and development tools are in use, which allow the simulation of complex full vehicle models as well as for testing control units on HiL test rigs. All tools have to be suitable for an integrated model based development process orientated along the V-model approach and also must provide thorough analysis possibilities which can facilitate the finding of errors at an early stage. For the modelling of powertrains the software tool Dymola is thought to be the standard.

The electronic development simulation models are primarily in use for testing the functionality and suitability for the series application of control unit software [4]. These tests are an essential part of the software development process.

Dynamic models are needed to test software independently without involving the mechanic and electronic hardware. With the help of different testing environments the software can be tested at each level of maturity.

Figure 2 shows the development phases and the adjoined testing environment and methods. The testing environments Model-in-the-Loop (MIL), Software-in-the-Loop (SIL) and HIL cover the whole process of software development. Hence models are not only needed in the conceptual phase, the left part of the V-model, but rather in the testing phase, presented on the right side of the V-model. In order to minimize service expenditures a unitary use of simulation models within all testing environments must be stipulated. The consequent application of the ReUse-concept does not only reduces the service expenditures for about 50%, but also reduces expenditures for integrating these models in the development process while heavily simplifying the version management.

Since in reality the integration of ZF products strongly varies among the different customer applications, a vast amount of different models are needed. One major objective is then to generate a universal model that can be configured for the appropriate customer application by solely changing model parameters.

One major disadvantage of Modelica is the inability to easily parametrize models for different development platforms. Therefore, an approach has been developed in ZF that has the ability to separate models from the parameters. The model parameters are stored within standardized ZF-ASCII-files, which will then be loaded at initialization [3]. In order to ensure a parametrization of models with ZBF-data on platforms without file I/O, e.g. dSPACE, modifications and adaptations within Dymola have been done. These procedures will be described in the following chapters.

3 ZBF-Format and Parametrization of models on platforms With File I/O

The ZBF-Format will be discussed in chapter 3.1, with an emphasis on its advantages in comparison to other parameter formats. In the following paragraphs an approach will be discussed which enables a parametrization of models by using ZBF-data within environments with File I/O.

3.1 ZBF-Parameter

As stated in the previous chapter, there is a strong need to separate models from parameters. A detailed description of the ZBF-format can be found in [3]. For the sake of completeness an example of the ZBF-format is included in Figure 3.

```
J1    [kgm^2]   0.1
; scalar parameter
InU   [-]       0    1    2
OutY  [-]       0    1    2
; two vectorial parameters
Test_Table2D[
[-]   U1 [-]   0    1    2
2     Y  [-]  -2   -1    0
1     Y  [-]  -1    0    1
0     Y  [-]   0    1    2
Test_Table2D]
; Two-Dimensional-Table
```

Figure 3: Parameterization with File I/O

ZBF originated from the strong need for exchange of formatted data between different Excel-programs. Thereinafter a broad use has been promoted for C und C++ calculation programs. It has been finally declared as a standard within ZF.

A big advantage of the ZBF-format is that it allows the provision of parameters which do not comply with SI-units, something usual for transmission design (e.g. [rev/min] instead of [1/s]) New approaches such as XML are not in use, since a large amount of programs are already able to read ZBF-data files. Furthermore, it is quite simple to transfer ZBF-data-files to Excel and edit these data files by using simple test editors.

With the help of an easy example, the differences of these different formats can be illustrated. A scalar parameter in ASCII such as the moment of inertia of the engine can be given as:

ZBF:
```
JMot  [kgm/s^2]  1.5
```

XML:
```
<Identifizierer>
<name>JMot</name>
<einheit>kgm/s^2</einheit>
<wert>1.5</wert>
</Identifizierer>
```

NetCDF:
```
netcdf      motor{
dimensions:
One = 1;
variables:
float JMot(One);
JMot:long_name = "Motor-
trägheitsmoment";
JMot:units = "kgm/s^2";
data:
JMot = 1.5;
}
```

3.2 Parametrization of models on platforms with File I/O

For the development of control function, models are used by CAE-tools running on PC-platforms with a file I/O operating system. A description of the approach on how to parameterize models on platforms with file I/O has been given in [3]. A short summary of the approaches in the forthcoming chapter, together with the necessary terms, will be presented next.

The parameters which are usually scalars, vectors and matrices are stored separately form the model at a central location (Figure 4). In order to read these data files, an appropriate parser will be linked to the model at the time of compilation. With the help of the parser, the model then reads all the necessary data at initialization and all parameters will be stored within a special data structure.

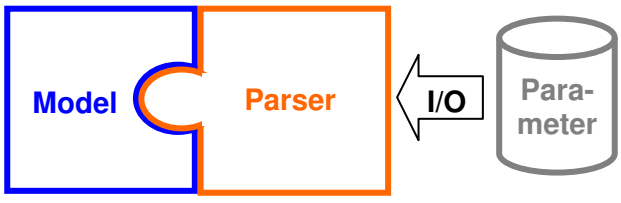

Figure 4: Parameterization with File I/O

The use of self developed C-Functions which will be linked to the model during compiling help to find the parameter and assign it to the component. Hence the model can be implemented within different CAE-applications, which run on an operating system with file I/O. The big advantage is that there is no need to modify the parametrization process based on what is required for the present application.

4 Parameterization of models on platforms Without File I/O

For controller testing, Dymola models have to be implemented in environments, such as SiL and HiL, with programs that do not allow for file I/O operations, such as dSPACE. Therefore another method has to be used. Moreover, the strong need for rapid prototyping calls for flexible parameterization within these environments. In the following chapter two realizations will be discussed. The first one will be referred to as "static parameterization" and the second "dynamic parameterization".

4.1 Static Parameterization

Within environments that do not allow file I/O operations one straight forward approach is to attach the parameter files to the existing Code. This will be done by converting the ZBF-files into C-code files and storing the parameters in a single character-string. Afterwards, these files will be linked to the model including the Parser throughout compilation (Figure 5).

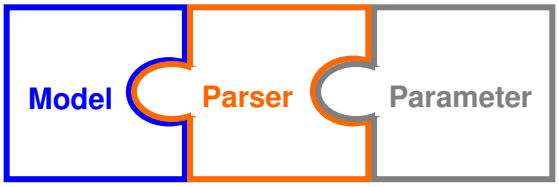

Figure 5: Static Parameterization

A slight extension of the existing Parser algorithm allows for proper parsing of the string and hence parameterization of the model at initialization. This approach is very useful in situations where parameters do not change very often or the model has to be exported as a single binary source.

4.2 Dynamic Parameterization

In order to test the robustness of a controller for various model settings, the static approach can be extended. This is done by changing the parameters directly within the code. Therefore a method is used which has been applied in dSPACE for easy re-parameterization of models on their hardware.

For this purposes the Dymola model will be imported into Matlab/Simulink as an S-Function. An extra parameter will be added to the S-Function during its generation by modifying the SimStruct to Dymola interface file *ss2dym.c*. Using this additional parameter the new set of ZBF parameter files can be passed on to the model in the form of an array of double values. An array is generated by Matlab from the default set of parameter in order to locate the parameter memory which will be needed later for re-parameterization (Figure 6).

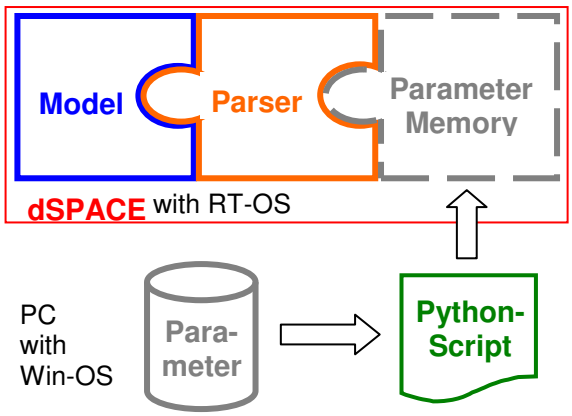

Figure 6: Dynamic Parameterization

The model with the additional S-function parameter (Figure 7) is exported into the dSPACE Real-Time platform using the Matlab RTI workshop. While exporting the model into dSPACE, the double array is converted into C-Code and subsequently linked to the model. This guarantees that the appropriate memory space can be accessed for dynamic re-parameterization. An SDF-file for dSPACE simulator is generated as well.

Finally the parameters can be transferred from the PC to the computer with the RT-OS with the help of Control-Desk. Fortunately, the re-parameterization can be done outside of Matlab. With the help of a python script the model parameter files will be re-parsed on the PC-platform with a regular file I/O, where the parameters will be converted into a double array of the same structure as the one for exporting purpose.

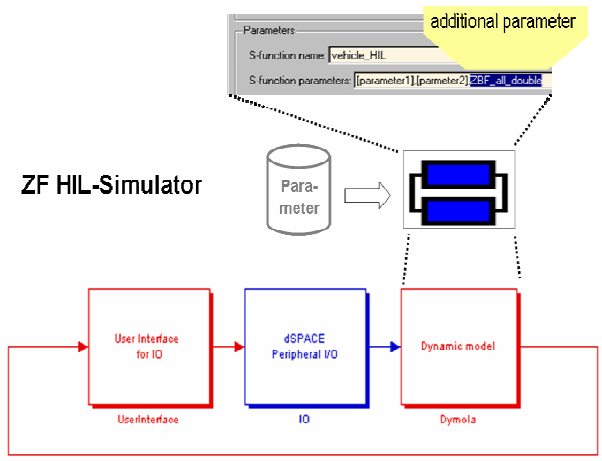

Figure 7: Additional S-Function Parameter

In order to "reload" the new parameters, the default parameters within the Real-Time model are accessed by another python script using the read/write routines from *rtplib* and ControlDesk. The length of the array is matched with that of the default array and the default parameter in the model is overwritten with new one. Finally, the initialization flag is activated and the model is re-initialized.

5 Dynamic ZBF-Parametrization of a passenger car model on dSPACE-HIL-Simulator

For Dymola vehicle models at ZF, all relevant mechanical, electrical and hydraulic modules needed for software development and HiL testing have been modelled.

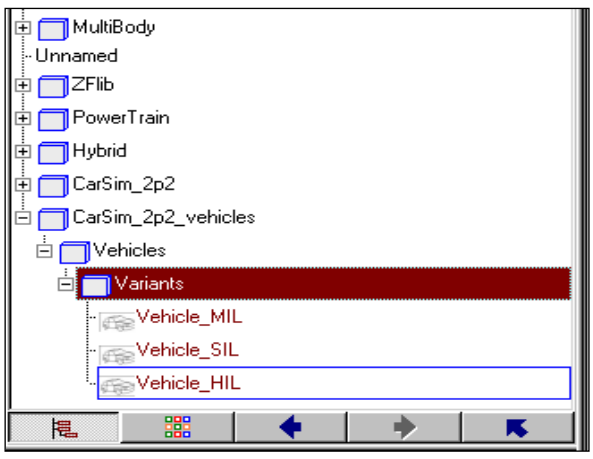

Figure 8: Modelica Libraries for HIL Tests

With the help of commercialized model packages, for example PowerTrain and MultiBody, as well as ZF-specific packages, as ZFLib, Hybrid and CarSim, powertrain models can be developed for different testing purposes (vgl.Figure 8).

In Figure 9 a vehicle model is shown as it is used for HIL-Simulation at ZF. It consists of the following sub-modules: engine, alternator, torque converter with torque converter clutch, ZF automatic transmission, rear axis, simple vehicle model with brakes, Control units (electrical/hydraulical), signal bus and I/O-interfaces.

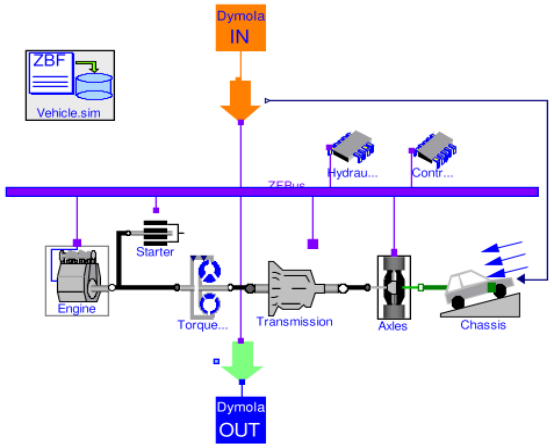

Figure 9: Dymola-Vehicle used in MIL, SIL and HIL environments

The degree of detail in the powertrain modules has to be adapted according to the field of application. Models which are used for testing purposes require a thorough consideration of internal interactions. For example the interaction of a set of clutches while doing a change in ratio requires a detailed application of system hydraulics modelling. Whereas the vehicle module has been simplified to a minimum and the engine has been represented by look-up tables in order to guarantee real time capability.

Due to the fact that ZF products will be implemented in different vehicle settings there will be a vast variety of models. The only possibility for dealing with this situation is following a modular approach, storing modules in libraries and separating models from data. The models can be parametrized by using data sets which relate to a specific version of vehicle set up. The basis for the parameter format is the ZBF-format, which has been described in chapter 3.1. The parametrization process of a vehicle model which is used for HiL-testing of a control unit on a dSPACE is explained in the following section.

The parameters have to be stored in ZBF-data files as well as an appropriate allocation within the model has to be done. Afterwards the model will be in-

cluded in a Simulink block. For the first time generating the model S-function the parameters will be read from file and stored in a double array within Matlab. At the same time the S-function will be supplemented by an additional parameter, which most often is referred to as the third parameter. This parameter is the essential link to the double array. Whenever the model is transferred to dSPACE by applying the RTI-Workshop the double array will be converted into a C-File and afterwards linked to the model. All described steps will be done automatically.

With the help of the described dynamic parametrization approach (see chapter 4.2) model parameters can be changed on the dSPACE-simulator. This is done by applying appropriate Python-scripts which allow for an easy change of parameters without starting the implementation process once again. This approach is quite essential, since not all dSPACE-HiL-simulators at ZF provide a Matlab development environment.

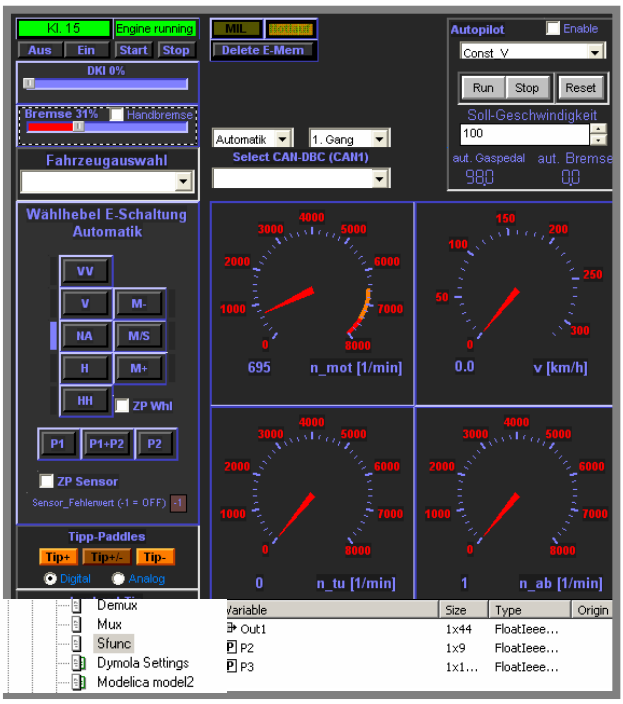

Bild 10: ControlDesk Interface

Appyling ControlDesk a graphical I/O user interface is set up, which interactes with the model on the dSPACE board. All necessary inputs and outputs of the testing environment can easily be monitored or changed. Typically the hardware configuration, e.g. the control unit variant and CAN-Bus-system can be selected. All control inputs of the Dymola-model such as ignition, selection of gear ratio, throttle and brake pedal position can be changed manually or automatically.

Finally by applying the mentioned Python-scripts within the windows-OS the appropriate set of ZBF-parameters of a vehicle variant will be read from file. By activating the third parameter within the model-SDF-file, parameters will be mapped into the allocated memory space of the model which is implemented on the dSPACE platform. Hence parameters can easily be changed while the model is operating at running time.

6 Summary and Outlook

In order to use models within different tools and environments models and parameters have to be kept separate from each other. Within ZF these parameter files are set up according to a standardized description referred to as ZBF. Approaches have been developed which enable a uniform and unanimous use of these files on all simulation platforms independent of whether they provide file I/O routines or not. For environments with file I/O, typically PC platforms, an approach based on linking a Parser algorithm to the model has been outlined in [3]. For environments without file I/O two realizations referred to as static and dynamic parameterization have been developed, where the latter allows for flexible parameterization. The static method generates a single source from model, parser and parameters. The dynamic method utilizes the method used by dSPACE. With the help of a Python script, the parameters will be read from ZBF files and directly mapped into the parameter memory of the model, hence facilitating the modification of parameters on Real-Time platforms.

Future work includes the issue of overruns occurring at initialization, presenting opportunities for improvements, especially for some specific environments which do not allow for overruns even at initialization. The optimization of code can also be possibility by applying the parameter evaluation feature within Dymola, which changes parameters into numbers and hence simplifies the code. Presently, his is not possible whenever parameter-files are in use. An extension which allows for optimization even when parameter files are in use can be very helpful if a more efficient Dymola code is to be developed.

7 References

[1] G. Bauer, M. Gromus, M. Neumann and C. Tapia. Model-based software development in production applications with a closed-loop controlled lockup clutch in a ZF 6-speed transmission, Fisita 2004

[2] H. Deiss, B. Aumann, T. Schober Time to Market in der Softwareentwicklung - Reuse und Standardisierung bei Getriebesteuerungen - Elektronik im Kraftfahrzeug, Baden-Baden 2000, Germany

[3] J. Köhler and A. Banerjee Usage of *Modelica* for transmission simulation in ZF, pp. 587-592, Gerhard Schmitz, Editor, Proceedings of the 4th International Modelica Conference, Hamburg March 7-8, 2005, Germany

[4] R. Gonzelez-Ramos, M. Neumann, A. Banerjee and J. Köhler Standard drive train models for increased Testing Efficiency, pp. 243, Proceedings of the 4th IAV Symposium, Berlin Juli 9-10, 2005, Germany