

## **AUTOMATED EXECUTION OF SIMULATION STUDIES DEMONSTRATED VIA A SIMULATION OF A CAR**

Sven Dominka

Dept. of Mechanical Engineering  
The University of Melbourne  
Melbourne, AUSTRALIA

Eduard Broecker  
Frank Schiller

Institute of Information Technology in Mechanical Engineering  
Technical University Munich  
85748 Garching, GERMANY

### **ABSTRACT**

In the automotive area, a huge number of different technologies were developed in the last couples of years. The car manufacturers have to meet the challenge of selecting the most suitable components for their purpose. One way of doing so is to evaluate these components by the help of a car simulation model. Because of the huge number of different technologies, also a lot of simulation models are needed to be able to compare all of those innovative technologies. In this paper, a system is described, with which it is possible to automatically generate simulation models. This system is used to generate varied simulation models which represent different car configurations. After the generation of a model for each car configuration, a parameter optimization is automatically executed. The result is a chosen car simulation model with its set of parameters, which presents an optimal solution for the regarded aim, e.g. efficiency.

### **1 INTRODUCTION**

With the globalization, the competition between different automotive manufacturers increases permanently. While the costs of cars mustn't increase, they have to become better and build more cutting-edge cars than their competitors. The customer demands in different aspects as comfort, safety, performance and environmental compatibility increase. For fulfilling those customer demands, a huge number of different technologies were developed in the last couples of years and are developed in time. These technologies span over a variety of power train components, like continuously variable transmission, Direct-Shift Gearbox, Hybrid vehicle, and safety features, like pre-crash functionality (intelligent belt pretension, intelligent breaks), lane keeping assist and emergency steering assist. The availability of innovative components is immense. In just one magazine, four articles were about new power train technologies (Birch 2007, Carney 2007, Brooke 2007, Birch 2007b). And the list of publications for new car innovations could be continued in any order. The actual

benefit of such components integrated into the complete car is often validated by the help of modeling and simulation. Furthermore, "vehicle system modeling is an important part of optimizing overall vehicle performance" (Tiller 2003). Especially for novel technologies like Hybrid Electrical Vehicles, simulation is very important for its design and evaluation (Hellgren 2002). For that purpose, simulation models of a car has to build up. Such simulation models, which are, or at least should be, built up hierarchically and modular, consists of all the major components of a car, e.g. engine, transmission, wheels, climate control unit, which are needed for the particular simulation purpose, for instance reduction of fuel consumption or the improvement of road performance. Those simulation models contain also the innovative components that have to be analyzed.

To be able to pick up the best of those new, innovative components, the car developers have to compare them against each other. Unfortunately, such comparing can be very time consuming and work intensive, as the components may be influenced by other, neighboring components. This statement is illustrated by means of an example: Assuming that a car configuration with a fuel consumption as small as possible should be ascertained. Therefore a variety of different automatic transmission types are available. The choice of the transmission is depending on the adopted engine. For low fuel consumption, the thermal power loss from the engine as well as the transmission has to be kept low. The power loss in turn is depending on the engine speed.

For fulfilling the requirements of maximum acceleration and low fuel consumption at the same time, the gear transmission ratio for high power engines should be smaller than for less power engines. That means that the transmission to be chosen depends on the engine. A similar dependence also exists for electrical motors and the connected gearboxes (Ross, Johansson and Wikander 2006).

But again, to actually compare different sets of car components with each other, the optimal parameter set for each car configuration has to be ascertained before. That means for example, that before different transmissions (in

combination with one particular engine) could be assessed, the switch curve for each transmission has to be adjusted because of the interplay with other neighboring components.

Summing up, to choose the best car configuration, that is the best set of innovative car components, it is necessary to compare all the possible alternatives against each other. Before the real comparison, for each variant, the optimal parameter set have to be identified.

This demand seems to be too high for practical purposes mainly because of the high efforts that are linked to such a claim. Still, in this paper, an approach is presented, with which such a comparison of varied, simulated car configurations is possible.

This approach comprehends the automated generation of simulation models. These generated models represent the possible alternatives of car configuration that means each model with its combination of car components is unique. After the model generation, an automated parameterization process is started to find the best suitable parameters for each variant. In the end, the simulation results are evaluated and the best car configuration with its set of parameters selected.

At first, the concept of automatically generating simulation models is described. The tool which generates simulation models is termed 'model generator' below. Afterwards, the parameterization process is explicated. In the end, the implementation of these concepts and its results are shown.

## 2 MODEL GENERATOR

The idea of automated generation of simulation models can be found several times in literature (Plönnings Neugebauer and Kabitzsch 2004, Balci 1990, Lee and Zobel 1996, Lorenz and Schulze 1995, Wong and Hwang 1992, Gonzalez et al. 2001). This approach is quite similar to the automated process of developing models (Taylor et al. 2001, Abourizk Shi Hccabe and Hajjar 1995) and to the automated process of simulation (Kang Szygenda 1994). Especially the automated model generation for electronic systems has been investigated in recent years (Roychowdhury 2003, Kang 1997, Monti 2001, Feng et al. 2007).

The approach of automatically generating simulation models described in this paper is based on the idea of connecting pre-modeled components, which are stored in a model library. The model generator first reads out an appropriate model library. The model library contains a collection of different car components, e.g. a range of different engines, transmissions, wheels and electrical components, like electrical generators and batteries. In addition to the real physical description, each model contains information about the model class, the possible neighbors and the provided interface of the model. By the help of this information, the model generator is able to systematically connect the components and build up (meaningful) models.

### 2.1 Importing model library

The model generator was not only designed for one special simulation application, but for a general purpose. Therefore the model generator does not know anything about the models to be generated at first. By reading in a model library, the model generator is able to get all the information about the subcomponents, out of which the models are put together. This information is split into two parts. The first part is the physical description of that particular subcomponent. In the case of an internal combustion engine, this description could e.g. include the transformation of energy from the fuel to the mechanical and to the thermal energy (see Figure 1). The second part of the provided information is solely for the model generator. It includes:

- Model type: this information is necessary for the classification of subcomponents. It denotes the general functionality of the particular subcomponent.
- Interface: the interface shows the neighboring subcomponents. Two subcomponents could only be connected to each other, when their interface type is identical.
- Parameter: The parameter data specifies which parameters of that particular subcomponent should be varied and which values should be used.

The result of the reading-in process is a table with the imported subcomponents. Each column contains the different subcomponents of one model type, e.g. different kind of engines. The number of columns is thus the number of model types, which are read in. An example of such a table is shown in Table 1.

Table 1: Exemplary table with subcomponents

FUEL TANK	ENGINE	TRANSMISSION	BRAKING SYSTEM	WHEELS	...
Diesel_tank	Gasoline-engine	Automatic-6Gear	Conventional	All-season	...
Unleaded-95	Diesel-Hybrid-engine	Twin-clutch	Regenerative	High-Performance	
Unleaded-98	Gasoline-Hybrid-engine	Automatic-CVT	...	...	
...	...	...			

For each of these table elements, the following data are stored:

- Name
- Physical description
- Model type
- Interfaces
- Parameters

After the successful importing process, the generating of simulation models can be started.

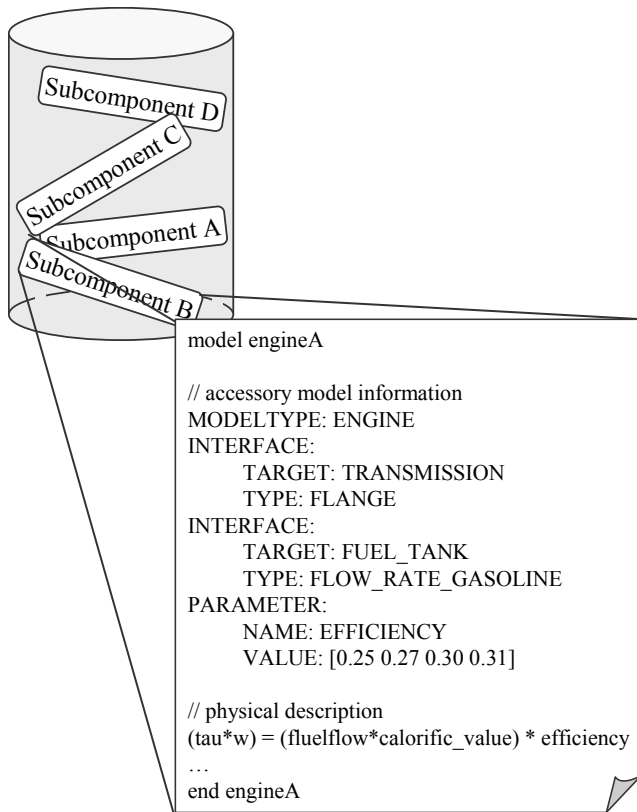


Figure 1: Subcomponent in a model library

## 2.2 Generating simulation models

The generation of simulation models works as follows: As the generated models are simulated automatically (e.g. over night) and the simulation could be run in a simulation cluster, the amount of time spent for simulation is not too critical. Hence, to achieve exhaustive results, a full factorial design was chosen for the comparing of structural model variations which means that "each factor [is] tested in combination with every level of every other factor" (Barton 2004). Therefore, the model generator generates all possible combinations of simulation models. The possible combinations of simulation models can be determined as follows:

Assuming that a (meaningful) simulation model contains exactly one instance of each model type, then the maximum number of possible combinations is:

$Y$  = Maximum number of possible combinations

$N$  = Number of model types

$a_i$  = Number of variants for the  $i$ -th model type

$$Y = \prod_{i=1}^N a_i$$

This number must be reduced for neighboring components which do not fit together because of unequal interface types. This statement is illustrated with the example of Figure 2. As a diesel engine could only be operated/ be powered with diesel fuel and a gasoline engine only with gasoline, the number of possible combinations has to be reduced at this point. The number of possible combinations and hence the number of generated simulation models in this case is:

$$(1 \text{ FUEL TANK} \times 1 \text{ ENGINE} + 2 \text{ FUEL TANK} \times 2 \text{ ENGINE}) \times 3 \text{ TRANSMISSION} \\ \times 2 \text{ BRAKING SYSTEM} \\ \times 2 \text{ WHEELS} \\ = 60 \text{ SIMULATION MODELS}$$

The actual procedure of assembling subcomponents to a simulation model depends on the simulation used. In Section 4, the generation of simulation models for the simulation tool Dymola (© Dynasim AB) is described.

## 3 PARAMETERIZATION AND SIMULATION

After the model generation, these models have to be simulated to identify the best solution. Each model is simulated with a set of different parameters that have been declared in the accessory model information block (see Figure 1). After simulating the generated models, the results have to be evaluated. Thus, for each simulation model, the following three steps have to pass through:

- Determination of parameter combinations
- Execution of simulation with each parameter set
- Evaluation of simulation results

Subcomponents may have different parameters, even when they have the same model type. A regenerative braking system, for instance, is aiming to regain as much energy as possible by the help of an electrical generator. In contrast, a conventional braking system just wants to decelerate the vehicle in the right way. Whereas the regenerative braking system may have a parameter like the numbers of windings, the conventional braking system may only have a parameter for the amplifying of brake power.

For each subcomponent, the parameters to be varied are given in the accessory model information block (see Figure 1). That means that each simulation model has its own set of parameters (which is likely to be different to the other generated simulation models). Each parameter has a certain number of values. In the event of a full factorial design, the number of simulation runs for each simulation model is:

$Z$  = Number of simulation runs for the simulation model

$M$  = Number of parameters (to be changed) in the simulation model

$\beta_j$  = Number of values for the  $j$ -th parameter

$$Z = \prod_{j=1}^M \beta_j$$

For the execution of a simulation, a connection to the simulation tool, in which the simulation should be executed, is needed. The generated simulation models together with the respective parameter set must be handed over in any way. The simulation tool has also to be provided with the simulation settings, like start time, end time, solver and step size. At last, the simulation results must be returned from the simulation tool for evaluating these results. There are some possibilities for coupling tools, like either file-based, via DLLs or communication channels through COM or CORBA (Kossel et al. 2006). As simulation tools differ strongly, it is not possible at this point to specify generally, how a coupling to a simulation tool could be achieved. However, in the next section, a coupling between the simulation tool Dymola (© Dynasim AB) and the tool MATLAB (©Mathworks), in which the model generator was implemented, is shown.

#### 4 IMPLEMENTATION

In this section, an example for the automated execution of simulation studies with a simulation automation system is shown. For that realization, the tools Dymola (© Dynasim AB) and MATLAB (©Mathworks) are used. The reason for this tool selection is as follows: Dymola provides a very good possibility to model large, complex and multidisciplinary systems quite easy. Even a lot of carmakers use Dymola (Tiller Bowles and Dempsey 2003, Knobel Janin and Woodruff 2006, Kossel 2006) and "the use of Modelica to model automotive systems is increasing" (Tiller Toller and Kuang 2002).

As a lot simulation tools do not provide an open architecture with a standard interface, it is not easy to establish a connection to those tools. Some of them however offer an interface to the tool MATLAB. Other reasons for the use of MATLAB are the already existent optimization algorithms and its large instruction set.

#### 4.1 Example of use - Dymola Powertrain Library

The example of use for the automated execution of simulation studies is based on a commercial Modelica library, called 'Powertrain' (Otter Dempsey and Schlegel 2000, Schweiger Dempsey and Otter 2005). The Powertrain library provides a wide range of 1D rotational mechanical components and other elements for the modeling of vehicle power trains. This library includes, amongst others, some vehicle engines, transmissions, axles, wheels, different drivers and car resistances (see Figure 2). The use case at this point is to find the power train configuration with the least fuel consumption.

#### 4.1 Modifying the Powertrain Library:

As mentioned in Section 2.1, for importing a model library correctly, accessory model information is needed. This information has to be added to the Powertrain library or more precisely to the subcomponents in that library, which are used for building-up power train simulation models. Figure 3 shows an example of how the subcomponents in the Powertrain library were extended with the accessory model information.

Each line of the added information begins with the character string `"//?#@#/"`. The double slash at the beginning indicates a comment in Dymola.

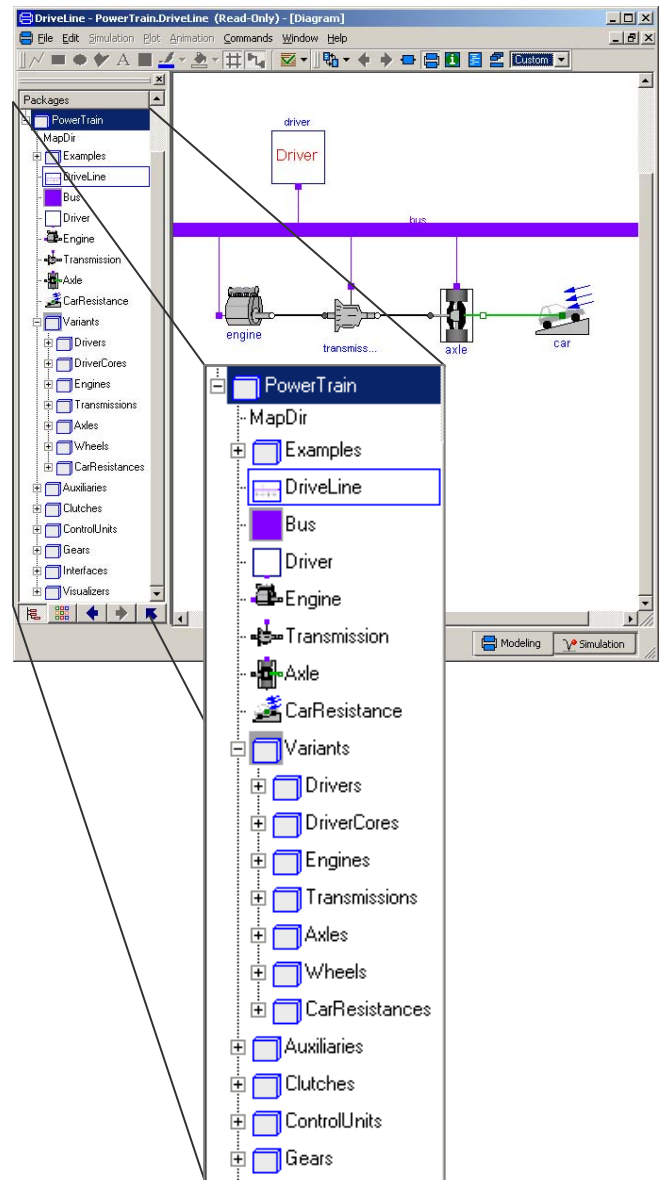
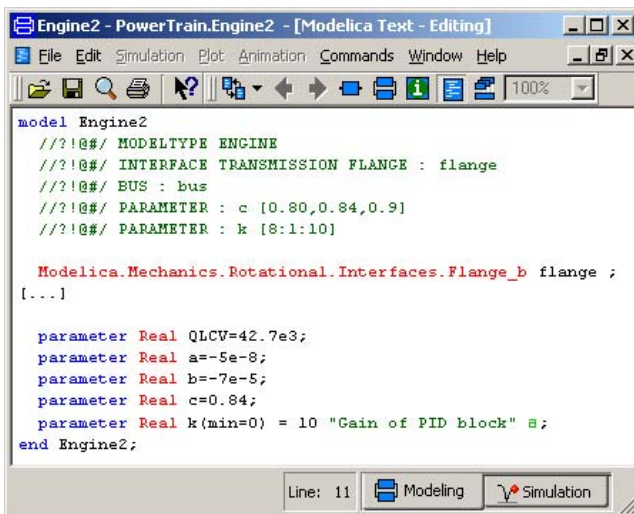


Figure 2: Dymola with Powertrain library

The rest of that string is for indicating that accessory model information is provided in this line. The first word in each line denotes the type of provided information, which is either the model type, an interface, a parameter or a bus connection:

- **MODELTYPE**: the only adjunct is the actual term of the model type. It is important that exactly the same term is used for all subcomponents of the same type.
- **INTERFACE**: there are three adjuncts. The first one is the neighboring component that means the model type of the subcomponent to which a connection can be established. The second adjunct is the interface type. Two components can only be connected to each other, if they provide the identical interface type. The third adjunct is specific to the model generation for Dymola. In this case, the name of the variable for that particular interface inside of the physical description block must be given.



```

model Engine2
  //?!@#/ MODELTYPE ENGINE
  //?!@#/ INTERFACE TRANSMISSION FLANGE : flange
  //?!@#/ BUS : bus
  //?!@#/ PARAMETER : c [0.80,0.84,0.9]
  //?!@#/ PARAMETER : k [8:1:10]

  Mdelica.Mechanics.Rotational.Interfaces.Flange_b flange ;
  [...]

  parameter Real QLCV=42.7e3;
  parameter Real a=-5e-8;
  parameter Real b=-7e-5;
  parameter Real c=0.84;
  parameter Real k(min=0) = 10 "Gain of PID block" #;
end Engine2;

```

Figure 3: Simplified illustration of an extended engine model

- **BUS**: the BUS statement is specific to the model generation for Dymola. If that particular subcomponent uses a bus communication, the variable name for that bus connection has to be specified.
- **PARAMETER**: The first of the two adjuncts is the name of a parameter which must already exist inside of the physical description block. The parameter name is followed by its values to be examined.

## 4.2 Simulation automation system

In this section, a developed tool is described, which actually does the automated execution of simulation studies.

The tool functionality is divided into:

- Importing model libraries
- Generation of simulation models
- Parameter variation and simulation

As this tool facilitates the automated execution of simulation studies including the generation of a variety of simulation models based on a set of subcomponents, it is termed 'simulation automation system'. In Figure 4, a screenshot of the simulation automation system is shown.

### 4.2.1 Importing model libraries

The importing of model libraries in case of Dymola works as follows: The selected Dymola model library is parsed for the key words 'model [subcomponent]' and 'end [subcomponent]'.

The text between these two key words is stored as the model description for this particular subcomponent. The result of this parsing process may be a long list of varies sub models. But actually, not each of these sub models is suitable as a subcomponent for the generation of simulation models. Only those models with accessory model information are taken for the model generation process. Therefore, all of the imported models are searched for the added information 'model type', 'interfaces', 'inserted bus' and 'parameters'. After that searching process, the simulation automation system shows the result in the form of an overview of the adequate subcomponents for the model generation process. In case of the example with the Dymola Powertrain library, the following model types with its number of subcomponents per class are available:

- DRIVER: 2
- ENGINE: 1
- TRANSMISSION: 3
- AXLE: 2
- CARRESISTANCE: 2

### 4.2.2 Model generator

For being able to automatically generate simulation models, it is necessary to know the exact syntax and semantics of the simulation language which should be used. In case of Dymola, a simulation language, called Modelica, is used. The main parts of the Modelica language specification could be gleaned in (Modelica Association 2005). Dymola is an object-oriented simulation tool which is based on hierarchical modeling. The models on the lower level generally map the behavior of a (sub-) system with physical formulas and equations. The upper level models uses such lower level models and just put them together. For the generation of simulation models, upper level mod-

els are used that means already existing subcomponents are put together in different ways. A simplified Dymola model layout for an upper level model may look like as follows (see Figure 5):

```

model model_name
  Subcomponent_A instanceA;
  Subcomponent_B instanceB;
  Subcomponent_C instanceC;
  ...
equation
connect(instanceA.interface1,instanceB.interface1);
connect(instanceB.interface2,instanceC.interface1);
  ...
end model_name;
    
```

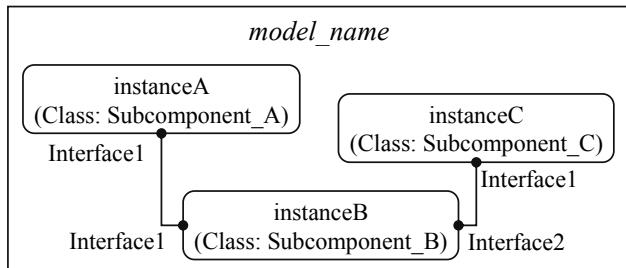


Figure 5: Graphical interpretation of an upper level model layout

In case of the example with the Dymola Powertrain library, the accessory model information contains that:

- the DRIVER subcomponents offer a BUS interface
- the ENGINE subcomponents offer a BUS interface as well as a interface to the model type TRANSMISSION
- the TRANSMISSION subcomponents offer a BUS interface as well as a interface to the model types ENGINE and AXLE
- the AXLE subcomponents offer a BUS interface as well as a interface to the model types TRANSMISSION and CARRESISTANCE
- the CARRESISTANCE subcomponents offer a interface to the model type AXLE

As in this example every subcomponent has the same interfaces than the other subcomponents with the same model type, a general structure for each generated simulation model can be shown in Figure 6.

The numbers of generated simulation models are numbers of subcomponents of each model type multiplied with each other. Therefore, the numbers of generated simulation models is 24 (see Figure 7).

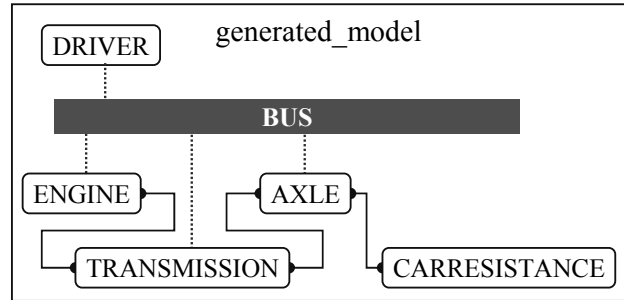


Figure 6: General structure for generated simulation models

generated_model1.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model2.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model3.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model4.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model5.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model6.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model7.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model8.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model9.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model10.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model11.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model12.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model13.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model14.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model15.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model16.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model17.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model18.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model19.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model20.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model21.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model22.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model23.mo	1 KB	MO-Datei	01.02.2008 13:07
generated_model24.mo	1 KB	MO-Datei	01.02.2008 13:07
Interfaces.mo	29 KB	MO-Datei	12.07.2006 11:17
package.mo	513 KB	MO-Datei	01.02.2008 10:56

Figure 7: Generated simulation models in Windows Explorer

### 4.2.3 Parameter variation and simulation

The process of parameter variation and simulation is quite dependent on the coupling between the simulation tool Dymola and the simulation automation system, which is implemented in MATLAB. Actually there are three possibilities of coupling. The first possibility is a script- and file-based coupling. For running a simulation, a Dymola script-file is generated. This file contains a script with all the necessary simulation settings and commands. After running Dymola with that script file, Dymola writes the simulation results in a result file which could be read in via the simulation automation system.

The other opportunities for coupling Dymola with MATLAB are based on either a DDE communication or a proprietary Dymola-Simulink interface (Modelica Association 2005).

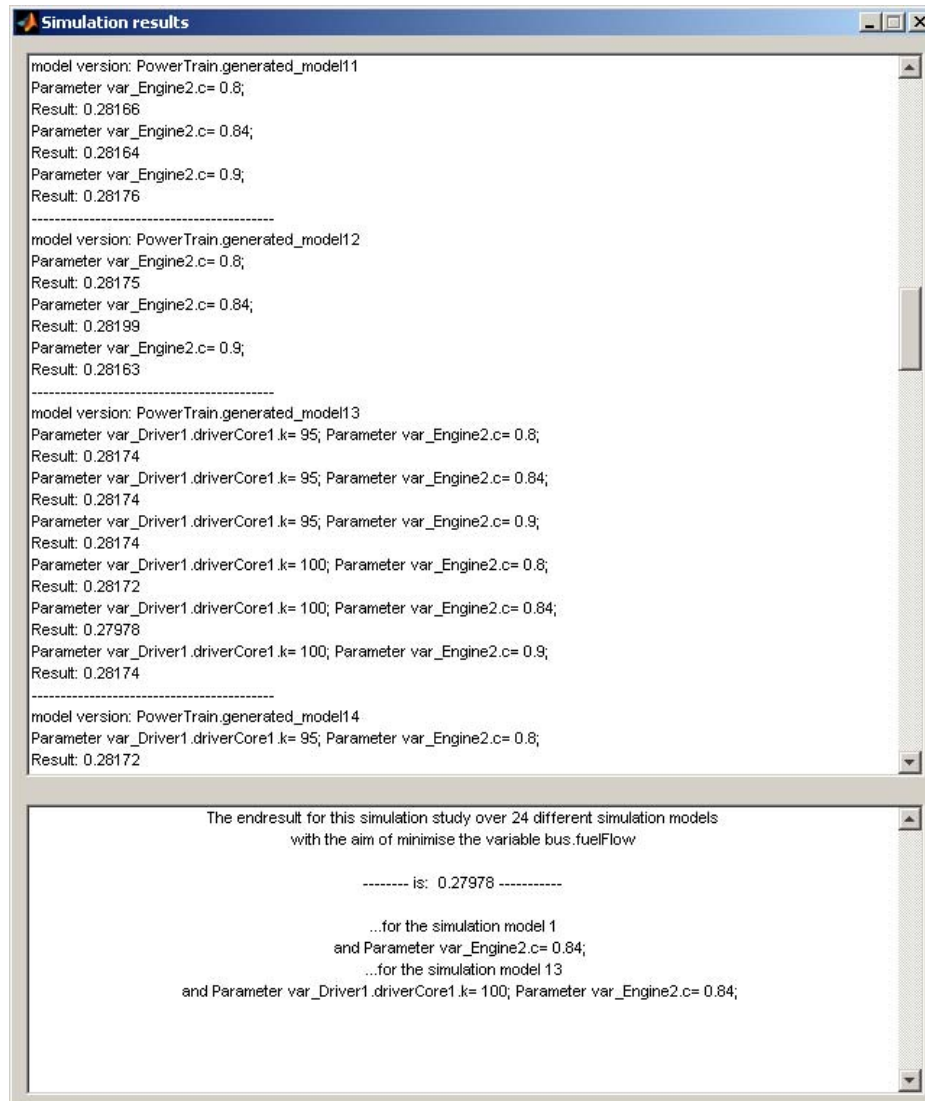


Figure 9: simulation results

In case of the simulation automation system, the script- and file-based coupling is used. The simulation automation system generates a Dymola script file for each simulation model (see Figure 8). These scripts contain commands for opening Dymola with the particular simulation model, the translation of the model and its simulation with varied parameter settings. After writing the results to a specific result file, Dymola is closed again. The result file is then analyzed and a new simulation model with a new Dymola script is executed.

```

clear
cd [...] /PowerTrain/
openModel (" [...] /PowerTrain/generated_model124.mo");
translateModel ("PowerTrain.generated_model124");

lnum = 1;
var_Driver1_driverCore1_k = [95;100];
var_Engine2_c = [0.8;0.84;0.9];
for k1 in 1:size(var_Driver1_driverCore1_k,1) loop
  for k2 in 1:size(var_Engine2_c,1) loop
    var_Driver1_driverCore1_k=var_Driver1_driverCore1_k[k1,1];
    var_Engine2_c=var_Engine2_c[k2,1];
    ResultName="Result"+integerString(lnum,1,1);
    simulateModel ("PowerTrain.generated_model124", [...]);
    lnum=lnum+1;
  end for;
end for;
exit

```

Figure 8: Generated Dymola script (reduced)

## 5 RESULTS

Figure 9 exemplifies the result of an automated execution of simulation studies. It shows the GUI of the developed simulation automation system, which is able to automatically run simulation studies including the generation of simulation models. In this example, 24 different simulation models with diverse subcomponents are generated. After running a simulation study for each model, the results are compared to each other regarding the minimal fuel consumption. Actually, two simulation models with their parameter sets show the best result. This result was achieved by a start time=0s and a stop time = 1s for each simulation model. The run of the simulation study, that is a parameter variation for 24 simulation models took therefore 7.7 minutes on a Pentium 4, 2x3.2Ghz, 2GB RAM. The same simulation study with a stop time = 120s achieved a different result and took 16.7 minutes.

Although this example is only a small one with only few different subcomponents, it illustrates the potential of an automated execution of simulation studies. The actual result of such a simulation study is however dependent on the model library which is used for the automated generation of simulation models. The introduced system combines existing sub models (representing car subcomponents) in a certain way and activates a parameter variation for each simulation model. That means that for achieving good results, the model library has to provide a collection of simulated car components with a certain grade of quality.

## 6 CONCLUSION

In this paper the approach for an automated execution of simulation studies was introduced. With a tool, called simulation automation system, it is possible to compare a huge number of different innovative car components with each other and to find the best combination out of such a collection of components. Therefore, the simulation automation system generates simulation models automatically, which differ from each other in the inserted subcomponents and the structure. After generating simulation models, for each model a parameter variation is carried out. The result of such an automated simulation study is a car configuration (with a certain parameter setting) which fulfills the regarded aim, e.g. least fuel consumption, best. The main benefit of such a simulation automation system is that simulation studies can be executed automatically and hence labor time can be saved. It enables faster progress as simulation studies can be run outside of (human) working hours. The constraints of the current system is the flexibility of model generation. Simulation models can only be generated when a suitable model library is provided.

The further steps in this research will be:

- Updating the present simple parameter variation with the possibility of optimizing each simulation model with suitable optimization algorithms (according to April et al. (2004)).
- Using the simulation automation system with a comprehensive model library, containing models for car components, like hybrid-electric power trains and different engine types.
- Involving multi criteria optimization, e.g. least fuel consumption with most comfort behavior.

## ACKNOWLEDGEMENTS

This work was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD). Further thanks go to the Research Centre for Advanced By-Wire Technology (RABiT), Department of Mechanical Engineering, The University of Melbourne.

## REFERENCES

- Abourizk, S.M., Shi, J., Hccabe B. and Hajjar, D. 1995. Automating the process of building simulation models, *Simulation Conference Proceedings, 1995. Winter*, pp. 1032-1038.
- April, J. et al. 2004. New advances and applications for marrying simulation and optimization, *Simulation Conference, 2004. Proceedings*, vol. 1, 5-8 Dec. 2004, pp. -86.
- Balci, O. et al. 1990. Model generation issues in a simulation support environment, *Simulation Conference, 1990. Proceedings., Winter*, 9-12 Dec 1990, pp. 257-263.
- Barton R.R. 2004. Designing simulation experiments, *Simulation Conference, 2004. Proceedings of the 2004 Winter*, vol. 1, 5-8 Dec. 2004, pp. -79.
- Birch S. 2007. DiesOtto: a diesel-gasoline technology blend, *automotive engineering*, vol. 115, no. 10, pp. 17-18.
- Birch S. 2007b. Green powertrain technology developments, *automotive engineering*, vol. 115, no. 10, pp. 22-23.
- Brooke L. 2007. GM's surprising new V8 diesel has no manifolds, *automotive engineering*, vol. 115, no. 10, pp. 20-21.
- Carney D. 2007. BMW shows X6 and ActiveHybrid concept, *automotive engineering*, vol. 115, no. 10, pp. 16.
- Feng T.H. et al. 2007. Automatic Model Generation for Black Box Real-Time Systems, *Design, Automation and Test in Europe Conference und Exhibition, 2007. DATE '07*, 16-20 April 2007, pp. 1-6.
- Gonzalez A.J. et al. 1994. Validation of an automated system model generator, *Knowledge and Data Engineering, IEEE Transactions on*, vol. 6, no. 4, pp. 643-648.



- Hellgren, J. 2002. Modelling of Hybrid Electric Vehicles in Modelica for Virtual Prototyping, *The 2nd International Modelica Conference*, pp. 247-256.
- Kang, S. 1997. Knowledge based automatic simulation model generation system, *Circuits, Devices and Systems, IEE Proceedings -*, vol. 144, no. 2, Apr 1997, pp. 88-96.
- Kang, S. and Szygenda, S.A. 1994. The simulation automation system (SAS); concepts, implementation, and results, *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 2, no. 1, pp. 89-99.
- Knobel, C., Janin, G. and Woodruff, A. 2006. *Development and Verification of a Series Car Modelica/ Dymola Multi-Body Model to Investigate Vehicle Dynamics Systems*, Vienna, Austria.
- Kossel, R. et al. 2006. Simulation of complex systems using Modelica and tool coupling, *The 5th International Modelica Conference*, pp. 485-490.
- Lee, C.H. and Zobel, R.N. 1996. Representation of simulation model components for model generation and a model library, *Simulation Symposium, 1996. Proceedings of the 29th Annual*, 8-11 Apr 1996, pp. 193-201.
- Lorenz, P. and Schulze, T. 1995. Layout based model generation, *Simulation Conference Proceedings, 1995. Winter*, 3-6 Dec 1995, pp. 728-735.
- Modelica Association. 2005. Modelica – A Unified Object-Oriented Language for Physical Systems Modeling: Language Specification, Version 2.2, Linköping, Sweden.
- Monti, A. et al. 2001. Symbolic analysis for automatic model generation, *Power Engineering Society Summer Meeting, 2001. IEEE*, vol. 3, pp. 1445-1450 vol.3.
- Otter, M., Dempsey, M. and Schlegel, C. 2000. Package PowerTrain. A Modelica library for modeling and simulation of vehicle power trains, *Modelica Workshop 2000 Proceedings*, pp. 23-32.
- Plönnigs, J., Neugebauer, M. and Kabitzsch, K. 2004. Automated Model Generation from Design Databases at the Example of Building Automation Networks, *Proceedings of the 1st International Symposium on Leveraging Applications of Formal Methods*, pp. 320-327.
- Roos, F., Johansson, H. and Wikander, J. 2006. Optimal selection of motor and gearhead in mechatronic applications, *Mechatronics*, vol. 16, no. 1, pp. 63 - 72.
- Roychowdhury, J. 2003. Automated macromodel generation for electronic systems, *Behavioral Modeling and Simulation, 2003. BMAS 2003. Proceedings of the 2003 International Workshop on*, pp. 11-16.
- Schweiger, C., Dempsey, M. and Otter, M. 2005. The PowerTrain Library: New Concepts and New Fields of Application, *Proceedings of the 4th International Modelica Conference*, pp. 457-466.
- Taylor, V. et al. 2001. Prophecy: automating the modeling process, *Active Middleware Services, 2001. Third Annual International Workshop on*, 6 Aug. 2001, pp. 3-11.
- Tiller, M., Tobler, W.E. and Kuang, M. 2002. Evaluating Engine Contributions to HEV Driveline Vibrations, *The 2nd International Modelica Conference, 2002*, pp. 19-24.
- Tiller, M., Bowles, P. and Dempsey, M. 2003. *Development of a Vehicle Modeling Architecture in Modelica*, Linköping, Sweden.
- Wong, Y. and Hwang, S. 1992. Knowledge-based distributed simulation generator, *Simulation Symposium, 1992. Proceedings. 25th Annual*, 6-9 Apr 1992, pp. 156-161.