Acta Polytechnica CTU Proceedings **12**:94–98, 2017
doi:10.14311/APP.2017.12.0094

available online at http://ojs.cvut.cz/ojs/index.php/app

SIMULATION OF ELECTRIC AND HYBRID VEHICLES IN A VEHICLE SIMULATOR BASED ON A DETAILED PHYSICAL MODEL, FOR THE PURPOSE OF HMI EVALUATION

D. Rozhdestvenskiy[a, b, *], J. Fulem[b]

[a] *Czech Technical University in Prague, Faculty of transportation Science, Horská 3, Prague, 120 00, Czech Republic*

[b] *Their Ricardo Prague s.r.o., Palác Karlín, Thámova 11-13, 186 00, Praha 8, Czech Republic*

[*] corresponding author: rozhddmi@fd.cvut.cz

Abstract. In this article, we propose a software solution to study HMI of electric and hybrid electric vehicles in vehicle simulators. We will start with the description of a development process of a physical model for HEV simulation in IGNITE software and equation-based language Modelica. A short introduction to the language, its possibilities, and explanation, why it is more suitable for the development of such models (vehicle powertrain), are presented in the first part of the article. A fusion mechanism of the physical engine with the model by means of FMU (Functional Mock-up Interface)[1] is also described in this part The second part is dedicated to the description of the model constructed in third party software IGNITE. This model has a detailed calculation of energy consumption and energy flow based on the selected control strategy. The last part of the article describes a possible experiment methodology.

Keywords: Electric vehicle, Hybrid vehicle, Powertrain simulation, HMI simulation.

1. Introduction

Electric and hybrids vehicles are slowly gaining popularity not only because it can help to reduce CO2 emissions and improve the environmental situation in big cities [2], but because of the rapid development in the area of electric motors batteries and Fuel Cells technologies. During the latest Paris Motor Show (2016), both VW and Mercedes Benz claimed to have overcome so far minimal acceptable limit of their pure EV (300 km is a minimal range acceptable by the user when considering to switch from conventional cars to EV or HEV [3]) almost twice. This can be achieved because of the new EV and charging architecture developed by companies[4].

Additionally, the ambitions of the automotive industry to introduce semi-autonomous and, later, fully autonomous vehicles on public roads before 2020, puts an extra urgency into design and evaluation of HMI technics for such vehicles.

User vehicle interaction is shifting to a new, so far, unstudied area with familiar knobs and gauges are being substituted with touchscreens, menus, and voice interactions[5].

The notion of "familiar" car is slowly vanishing, putting a new driver (user) in a position where he has to learn how to drive a vehicle all over again if he switches the brand. This become even more crucial if we take into account that car sharing is slowly substituting car ownership[6].

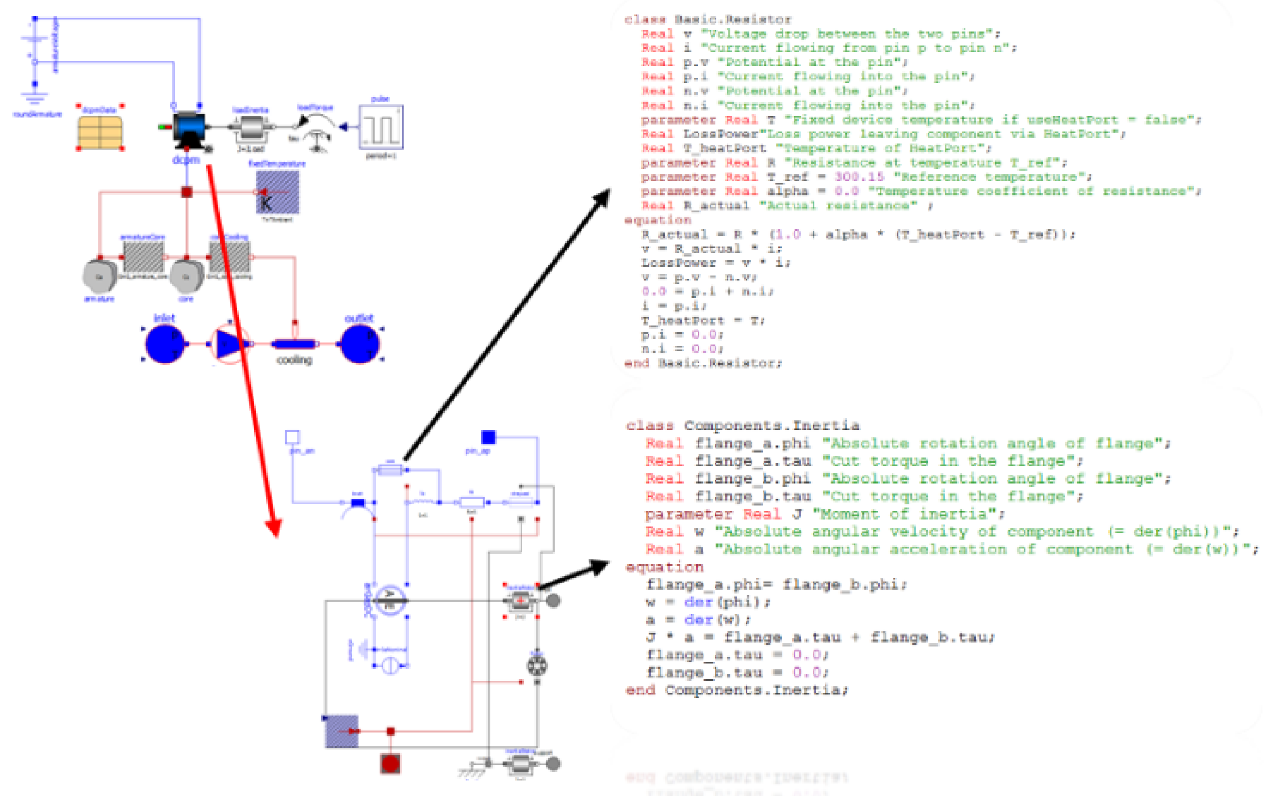
The processes of development, design, and evaluation of HMI is to adapt to this rapidly changing environment. An efficient way of assessment of vehicle interfaces at different stages of their development can be achieved by testing it in a simulated environment. A model of an interactive EV and HEV driving simulator with a possibility of rapid adaptation to vehicle type or architecture, with an accurate simulation of the powertrain components developed in Modelica language, is hereby suggested.

2. Modelica programming language

Modelica is an open source object-oriented language for modeling of massive, complex systems. It is suited for multi-domain modeling (including electrical, physical, thermal, mechanical and other domains), additionally, it can be used for hardware-in-the-loop simulations and for embedded control systems [7] Models in Modelica are mathematically described by differential, algebraic and discrete equations. From a user perspective, models are described by schematic object diagrams). The internal part of a component is defined by another schematic or on a lower level, by an equation-based description of the model in Modelica syntax. An example of such a model is shown in Figure 1.

Components can be combined in a complex model by drawing connection lines between "connectors". Connectors describe the interaction possibilities (electrical pin, mechanical flange, or an input signal.

Regarding development time for complex models, Modelica language is on average, twice as fast as any other technics in proprietary code or in any other



```
class Basic.Resistor
  Real v "Voltage drop between the two pins";
  Real i "Current flowing from pin p to pin n";
  Real p.v "Potential at the pin";
  Real p.i "Current flowing into the pin";
  Real n.v "Potential at the pin";
  Real n.i "Current flowing into the pin";
  parameter Real T "Fixed device temperature if useHeatPort = false";
  Real LossPower"Loss power leaving component via HeatPort";
  Real T_heatPort "Temperature of HeatPort";
  parameter Real R "Resistance at temperature T_ref";
  parameter Real T_ref = 300.15 "Reference temperature";
  parameter Real alpha = 0.0 "Temperature coefficient of resistance";
  Real R_actual "Actual resistance" ;
equation
  R_actual = R * (1.0 + alpha * (T_heatPort - T_ref));
  v = R_actual * i;
  LossPower = v * i;
  v = p.v - n.v;
  0.0 = p.i + n.i;
  i = p.i;
  T_heatPort = T;
  p.i = 0.0;
  n.i = 0.0;
end Basic.Resistor;
```

```
class Components.Inertia
  Real flange_a.phi "Absolute rotation angle of flange";
  Real flange_a.tau "Cut torque in the flange";
  Real flange_b.phi "Absolute rotation angle of flange";
  Real flange_b.tau "Cut torque in the flange";
  parameter Real J "Moment of inertia";
  Real w "Absolute angular velocity of component (= der(phi))";
  Real a "Absolute angular acceleration of component (= der(w))";
equation
  flange_a.phi= flange_b.phi;
  w = der(phi);
  a = der(w);
  J * a = flange_a.tau + flange_b.tau;
  flange_a.tau = 0.0;
  flange_b.tau = 0.0;
end Components.Inertia;
```

Figure 1. Modelica model architecture (example of schematic diagram and corresponding programming code).

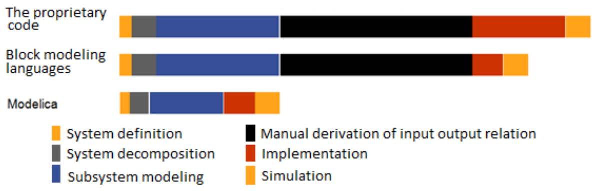


Figure 2. Development time chart.

block modeling languages. This is explained by several factors:

- A developer does not have to manually derive relationships between inputs and outputs between subsystems models in Modelica are mathematically described by differential, algebraic and discrete equations. No particular variable needs to be solved for manually
- Modelica code has been developed since 1996 so a lot of subsystem components have been already developed and available for public
- Implementation time and complexity is the same as for any Block modeling language

All these points are summarized in Figure 2.

As mentioned above this puts Modelica language ahead of others, especially when dealing with simulation of such a complex and multidisciplinary system as hybrid and electric vehicle powertrain. Integration of the model in the virtual world creates a new engineering task which can be solved with an application of the FMI (functional mock-up interface standard).

3. Function mockup interface

Functional Mock-up Interface (FMI) is a standard to support both model exchange and co-simulation of dynamic models developed in different independent tools, its primary goal is to support the exchange of simulation models between suppliers and OEMs, providing a possibility to protect product knowledge, which could be recovered from these models.

This standard is going to be used to perform a simulation of the EV or HEV powertrain inside a physical world simulated in ODE or Bullet physical engine.

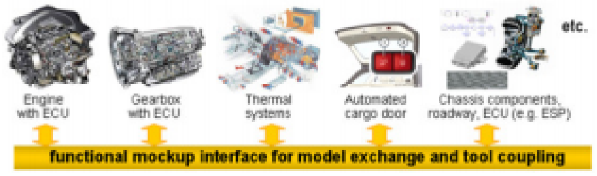

Engine with ECU
Gearbox with ECU
Thermal systems
Automated cargo door
Chassis components, roadway, ECU (e.g. ESP)
etc.
functional mockup interface for model exchange and tool coupling

Figure 3. FMI integration concept[1].

4. Model creation and validation

A lot of commercial software available on the market are using Modelica as a simulation language, and either already have, or are planning to implement FMU import/export routine. One such software is IGNITE, developed by RICARDO Software. IGNITE is a physics-based system simulation package focused at complete vehicle system modeling and simulation. It consists of set of libraries containing vehicle and thermal system components such as engine, transmission, controllers, motors and e.t.c.

These components allow faster and easier modeling and parameterization of a powertrain system for any type of vehicle, and a simulation of temperature dependent systems, which will allow further possibilities to simulate HVAC and battery management of HEV and EV.

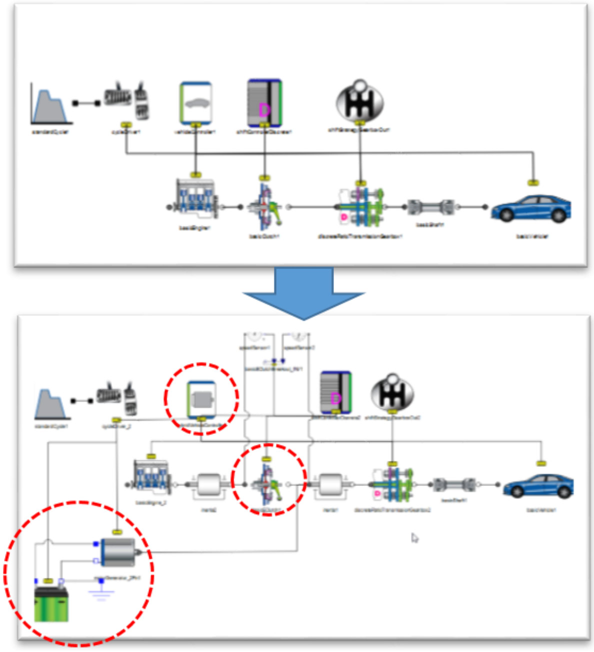


Figure 4. Model hybridization.

Creation of a model of HEV in IGNITE is a straightforward process that consists of several steps [8]:

(1.) Collect all necessary vehicle information (tires, aerodynamics, mass etc.). Vehicle manufacturer usually provides this data in technical documentation.

(2.) Collect powertrain information (gear efficiencies, shifting maps, engine and electric machine maps and battery characteristics). This data can either be obtained from the automaker directly on request or derived from an experimental data mathematically.

(3.) Build a conventional vehicle model using IGNITE libraries:

- Parametrize the model with collected data.
- Run test cases to validate the model.
- Turn the conventional model into a hybrid – add hybrid components (Electric motor, battery, controller).

Validation of the model cacan be performed using standard driver cycles, for instance, NDC or FTP72, results obtained from the simulation on the drive cycle should be compared with an experimental data or analyzed analytically. Results of the simulation contain comprehensive data for every model component and provide the possibility to perform this analysis. An example of the validation for a P2 hybrid with 1.4L TSI motor and 25 kW electrical motor is shown in Figure 5. The simulation showed an average fuel consumption of 4.3 L/100km, which is equivalent to 100 g/km CO2 emissions on NEDC drive cycle for this model. These results closely corelate with the experimental date for similar vehicles. This validation was performed during the development stage of the hybridmodelt at Ricardo P.L.C.and described in details in [8].

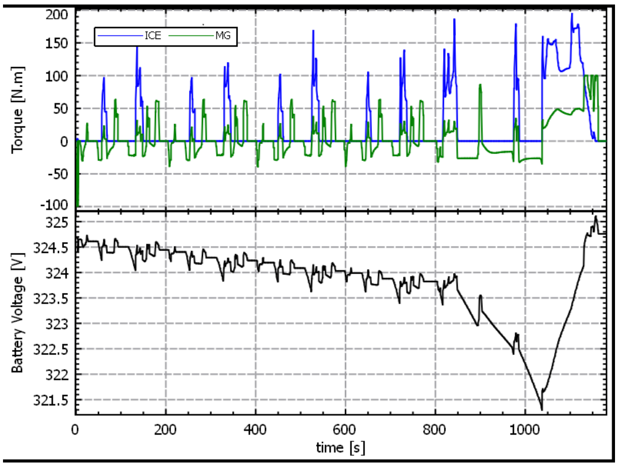


Figure 5. Engine Map, Torque and SoC results.

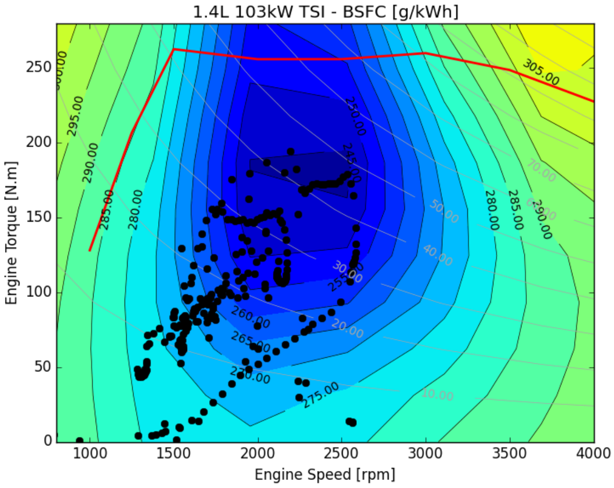


Figure 6. Engine Map results.

5. Integration of FMI into the simulator

Once the model was created and validated, it has to be modified, to perform a simulation with a physical engine. As we are interested in the interactive simulation for the purpose of HMI evaluation, we need to provide the means of control for a driver to interact with the model. At least 3 inputs have to be introduced to provide a longitudinal control of the simulated vehicle to the user. These inputs are:

- Acceleration demand (Acceleration pedal position <0,1>).
- Break demand (Brake Pedal Position <0,1>).
- Current Vehicle Velocity.

Velocity input provides a feedback of an actual speed of the vehicle simulated in the ODE (Open Dynamic Engine) simulation environment, which is calculated there with respect to elevation grade, tire/road contact friction, collisions, bumps etc.

As a direct output, an applied wheel torque should be obtained from the model, this torque is applied to the physical model of the vehicle in the ODE or Bullet physical engine which is simultaneously handling latitudinal behavior. All other losses (Aerodynamic, acceleration, internal and rolling losses) are calculated in IGNITE model directly.

For the purpose of HMI simulation, we have to obtain additional outputs from the model to simulate gauges and control element of HEV. These inputs are:

- State of Battery charge SoC.
- Engine speed.
- Electric machine engine speed.
- Battery temperature.
- ICE state.

These outputs can then be further processed to be displayed on a virtual gauge or in any other graphical form. The architecture of this simulation environment is shown in Figure 7.

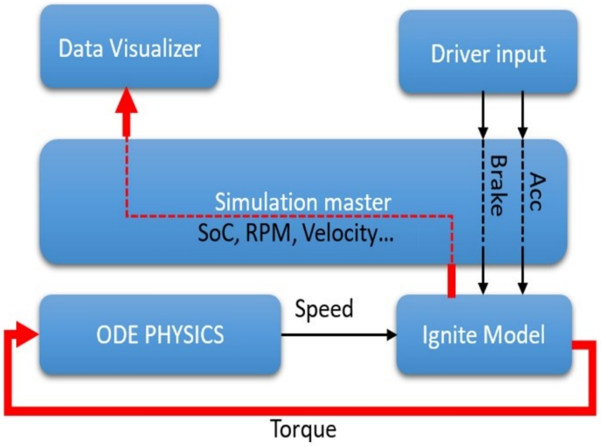


Figure 7. Simulation architecture.

6. Possible experiments with detailed consumption model

After the vehicle model and simulation environment has been created, an experiment itself should be designed. We can look at four possible types of experiments with EVs and HEVs:

- Experiments aimed at determination of general user acceptance of HMI and its efficiency.
- Experiments aimed at study of acceptance and effect of particular HMI element.
- Experiments aimed at comparing different HMI.
- Experiments aimed to determine preferred system types of notifications per different kinds of events.

Even though any of these types of experiments should be adapted to a specific requirement, one may distinguish ten high-level steps essential for any type of experiment. (Janssen at al. 2004):

(1.) To determine the goals of evaluation in terms of usability, safety, workload, acceptability, time frame.

(2.) To describe a system (systems) which will be tested during the experiment.

(3.) To define a high-level scenario (scenarios) in terms of road type, and conditions, visibility, traffic type and actors, tasks, and goals.

(4.) Define the subject testing cohort in terms of amount, age and experience group.

(5.) Define subjective and objective evaluation parameters and instruments for data acquisition.

(6.) Develop experimental instruction for both participants and experimenters.

(7.) Perform a pilot experiment test and data evaluation.

(8.) Carry out the experiment.

(9.) Analyze data.

(10.) Produce summary report and draw a conclusion.

7. Conclusion

In this article, we presented a methodology for development and integration of high fidelity powertrain model of EV or HEV into three-dimensional virtual simulation environment. This model provides a comprehensive and precise data results to simulate next generation HMI for EV or HEV including gauges and functions (such as Power-in-Out gauges, predictive Navigation and others). These models are written in Modelica Programming language and coupled with ODE or Bullet physical world with the help of FMI. This approach allows a rapid development and validation of models, and a quick way of switching between them.

References

[1] T. B. at all. *The Functional Mockup Interface for Tool independent Exchange of Simulation Models.* In Proceedings of the 8th International Modelica Conference.

[2] D. Rozhdestvenskiy, P. Bouchner, A. Mashko, et al. *Dynamic Human-Machine Interface for Electrical Vehicle design guidelines.* SMART CITIES SYMPOSIUM PRAGUE 2015, ISBN 978-1-4673-6727-1.

[3] T. Smith, H. Vardhan, B. Malet, D. Mingay. *Are we there yet. Thoughts on in-car HMI.* 2014.

[4] G. E. Outlook2016. *Beyond one million electric cars.* International Energy agency, Electric Vehicle Initiative.

[5] S. Heath, P. Sant, B. Allen. *Do you feel lucky? Why current range estimation methods are holding back.* EV adoption Hybrid and Electric Vehicles Conference 2013 (HEVC 2013), IET Year: 2013.

[6] P. B. et al. *Professional vehicle drivers training and performance measurements with use of advanced driving simulator laboratory.* 1.ed., Prague: CTU in Prague, Faculty of Transportation Sciences, Department of Vehicles, 2013, 1st, electronic edition. ISBN 978-80-01-05421-5.

[7] J. Tobolar, M. Otter, T. Bunte. *Modelling of Vehicle Powertrains with the Modelica PowerTrain Library.* In: Dynamisches Gesamtsystemverhalten von Fahrzeugantrieben.

[8] J. Fulem. *IGNITE & WAVE-RT powertrain model connected to IPG virtual test driving platform.* In: Ricardo Software UserCon Europe.