MULTI-DOMAIN MODELING AND DISTRIBUTED REAL-TIME SIMULATION OF AN AUTONOMOUS VEHICLE

Tamas Juhasz[1], Ulrich Schmucker[2]

[1] Institute for Mobile Systems, Otto-von-Guericke University
Universitaetsplatz 2, Magdeburg, 39106, Germany
[2] Virtual Engineering Expert Group, Fraunhofer Institute for Factory Operation and Automation, Sandtorstrasse 22, 39106 Magdeburg, Germany

tamas.juhasz@ovgu.de (Tamas Juhasz)

Abstract

In this paper the modeling and simulation of an autonomous vehicle is presented. The complete reliability of the vehicle's onboard embedded systems is crucially important, especially when their functions are relevant to safety. Increasingly extensive embedded software and even shorter manufacturing times in the automotive industry make it nearly impossible for developers to test real prototypes down to the last detail. For these reasons the functional safety verification of a complete car system should involve virtual engineering techniques. Our goal is to analyze the safety of these embedded systems by means of co-simulation and hardware-in-the-loop tests. This task requires a real-time capable model of the complete mechatronic system. The simulation of the complete car system is distributed among separate simulators that are communicating in real-time with each other: the sensors, the navigation algorithms and the road environment are also co-simulated with our distributed multi-domain model of the driving and steering subsystems and the chassis. The paper reflects also some modeling thoughts that had to be met in order to guarantee a good simulation performance using the Modelica language and Dymola environment.

Keywords: System Modeling, Autonomous Vehicle, Modelica, Dymola.

Presenting Author's biography

Tamas Juhasz received his PhD title in Informatics from the Technical University of Budapest, Hungary in 2009. His thesis has focused on different areas of computer-aided engineering: multi-domain modeling and simulation of physical systems including advanced visualization methods. His recent research addresses the partially automated translation of CAD information to Modelica models, allowing a rapid way of analyzing and optimizing a mechatronic system before it even physically existed.

1 Motivation

1.1 The ViERforES project

The researchers in the project "Virtual and Augmented Reality for Maximum Embedded System Safety, Security and Reliability" are working on concepts intended to help companies optimize the safety, security, reliability and availability of their products and applications. This project is supported by the Federal Ministry of Education and Research, Germany.

In many products, complex software and microprocessors concealed inside so-called embedded systems (ES) assume a key role in the functional interaction of all the components. Therefore, such systems' accuracy and complete reliability is crucially important, especially when their functions are relevant to safety. Increasingly extensive software and even shorter manufacturing times make it nearly impossible for developers to test every system down to the last detail.

In automotive applications there is a large amount of embedded systems being used. For the reasons above, the functional safety verification of a complete car system should involve virtual engineering techniques. Between individual ES components there are so many interactions possible that cannot be verified with conventional methods. A realistic model of the car's hardware is needed in order to run tests in a simulated virtual reality. This model has to communicate also with the real embedded systems later, thus real-time-capable modeling with possibilities for hardware-in-the-loop simulation is required.

1.2 The platform RAVON

RAVON (Robust Autonomous Vehicle for Off-road Navigation) is a robot car being developed at the Robotics Research Lab at the Technical University of Kaiserslautern, Germany [1]:

Fig. 1.: A photo of RAVON

This unmanned vehicle is used as a test-bed to investigate behavior-based strategies on motion adaptation, localization and navigation in rough outdoor terrain. The basic chassis – called RobuCar – was produced by the French firm Robosoft.

In comparison to a modern passenger car, RAVON has a similar amount of embedded systems onboard, and it also uses bus-systems widely adopted in the car industry (e.g.: CAN). These facts allow us to use RAVON as a pilot platform during researching on the subject. Conversely to the passenger car manufacturers' secrecy, we have the advantage that all necessary data of RAVON stays at our disposal.

We found a possibility to let the embedded software of RAVON communicate with our simulated car instead of the real system in a transparent way, thus we can also validate our models by using the same input and measurements of the real system.

2 Background

In the last decades the object-oriented paradigm of the software engineering infiltrated the modeling of physical systems. This concept allows reusing and integration of acquired knowledge of previous models within multiple modeling domains.

2.1 The Modelica language

The Modelica standard is developed continuously since 1997 by an international non-profit association [2]. This modern, object-oriented modeling language allows describing a complex mechatronic system over multiple engineering domains. It is based on bi-directional mathematic equations, whereby the causal relations are only determined by the direction of information flow. Besides process and control system-oriented components it supports modeling of mechanical-, electronic-, hydraulic, pneumatic and thermal subsystems of a complex mechatronic system [3].

2.2 The Dymola simulator

Dymola is our chosen environment for simulating Modelica models [4]. It can manipulate the equation system of a complex model automatically, so the number of equations and the rank of the system will be significantly reduced. Using a non-linear fixed-step solver we can simulate our model in real-time.

Moreover there is a possibility for embedding a complete Modelica model into a Simulink block, integrating the advantages of the Matlab simulation environment as well.

3 Modeling of RAVON in Modelica

Our task in the ViERforES project is to create a virtual car model in Modelica that is capable to mimic the dynamics of the RAVON chassis (including electric drives, power steering, battery, etc.). Considering the existing embedded software for higher-level behavioral tasks (such as navigation, image processing, etc.), our model must accept the same reference signals that would be used to control the real car and it has to react similarly to them, just like the real RAVON would do. This allows us to carry out

safety tests transparently without risking any personal injuries or hardware losses.

In the extended RAVON system the individual virtual and real components can communicate with each other in real-time over a network using a mirrored shared memory concept. During our tests the aforementioned individual behavioral modules (navigation, obstacle avoidance, etc.) are assumed to communicate with our virtual chassis model in that manner. The simulation of the virtual chassis itself is distributed among multiple computers because of computational requirements.

The modeling task of RAVON was carried out in multiple domains that are described in the following sections individually.

Using abstract model interfaces we can exploit the object-oriented model encapsulation feature of Modelica, so the replacement / refinement of any modules can be done transparently, as long as they implement the same communication interface.

3.1 Automated translation of the mechanical structure

The RRLab – our project partner in TU Kaiserslautern – put the necessary CAD data of RAVON in Pro/Engineer format at our disposal:

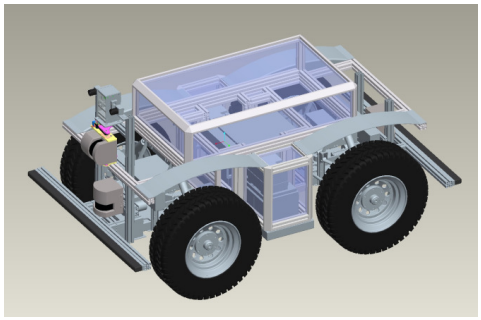

Fig. 2.: Pro/Engineer model of RAVON

We have developed an exporter add-in module for Pro/E that is seamlessly integrated to the user interface. The output of this module contains XML information about the physical and kinematical properties (masses, moments of inertia, joint constraints, etc.) of the mechanical model and also includes triangulated VRML geometry directly exported from the CAD system. In the recent years we also developed a translator [5] that could easily generate the parameterized Modelica model of RAVON's mechanical structure directly of the aforementioned XML / VRML data.

This auto-generated chassis model served only as a good basis for further modeling tasks in the mechanical domain. The tire model and the gears in each wheel had to be manually added in order to finalize the mechanical model.

3.2 The tire model and the road interface

A tire model has the task to compute 3D tire deformations and resulting longitudinal and lateral forces and torques based on known contact parameters.

We implemented a dynamic tire model by utilizing Taylor-expansion of the steady-state forces and torques described in [6]. However this approach is rather simple, it has an advantage of being effective in matching measurement data, thus it is very suitable for real-time vehicle dynamic simulations. The tire model requires only a few geometrical and steady state properties (e.g.: diameter, wheel dimension, stiffness / damping factors) that can be easily determined experimentally.

In order to simplify the tire-road contact parameter processing, we decided to use a strict horizontal plane for the road geometry. The desired 3D inclination values of the real road are input variables of our road interface.

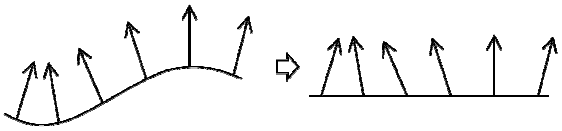

Fig. 3.: Inclination vectors of the road interface

Our virtual vehicle is always riding on this flat road, while there is a block in the model that computes the external force that arises continuously due to the varying inclination of the real road. The sum of the vertical gravity force and this external force **G'** yields the "actual" gravity force **G** that would act on the inclined chassis (see Fig. 4).

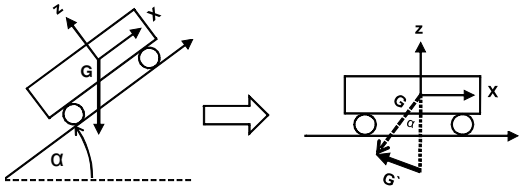

Fig. 4.: Computing the additional external force G' due to inclined road

Let the magnitude of the car's weight be $G=m \cdot g$, and let the road inclination be given with a 3-component normal vector **N** in the coordinate system of the chassis. The additional **G'** force can be computed then as follows:

$$\mathbf{G'} = G \begin{pmatrix} N_x \\ N_y \\ 1 - N_z \end{pmatrix} \qquad (1)$$

In our model a time-varying μ friction coefficient can also be specified for each wheel individually. In case we want to model some wheels losing ground contact, we can specify a small $\mu=10^{-4}$ factor for those tires.

3.3 Steering subsystem

RAVON has Ackermann-type independent steering at both axles. Each steering actuator has a central controlled angle *d*, the value of which must be transformed to the individual left and right wheel's yaw angle *d1* and *d2*, respectively:

$$d_{1,2} = d\frac{4b^2\left(1+\tan^2 d\right)}{4b\left(b \pm tw\tan d + b\tan^2 d\right)+tw^2\tan^2 d} \qquad (2)$$

The *b* and *tw* parameters in equation (2) stand for the axle-base distance (1206 mm) and the track-width (1156 mm) values of RAVON, respectively.

3.4 Drive subsystem

RAVON – our test platform – has individual 4-wheel drive with one Huebner HAC 71.11.6 permanent magnet synchronous motor and a Cyclo Drive 6000 1:59 gear per each wheel.

For the effective modeling of the drive subsystem we have decoupled the pure mechanical parts (*Chassis*) from the further detailed electromagnetic domain (*Drive*, see Fig. 5). We use 1D rotational flange connectors to drive the active revolute joints in the mechanical *Chassis* model and the bus connector concept of Modelica to forward other important signals between these main blocks:

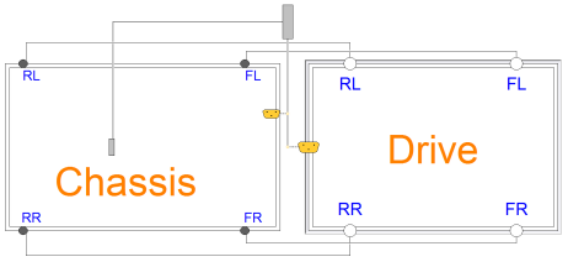

Fig. 5.: The topmost Modelica blocks of RAVON

The main *Drive* block at the right hand side consists of four electric drive blocks and a central battery block (Fig. 6). For the sake of a good signal routing we use the virtual bus concept here, too:

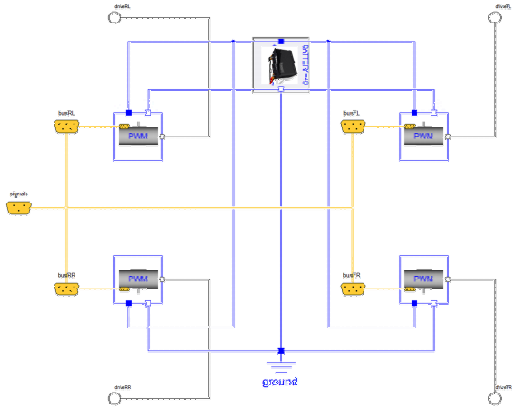

Fig. 6.: The main Drive block's schematics

In our actual case the real-time capability of the complete virtual RAVON model has the highest priority. Therefore we decided to substitute the modeling of the synchronous motor first with a velocity and current controlled DC motor model. This simplification is allowed under our circumstances because the controlled system's dynamic behavior is very similar in both cases [7]. The following figure shows the contents of a drive block of a single wheel:

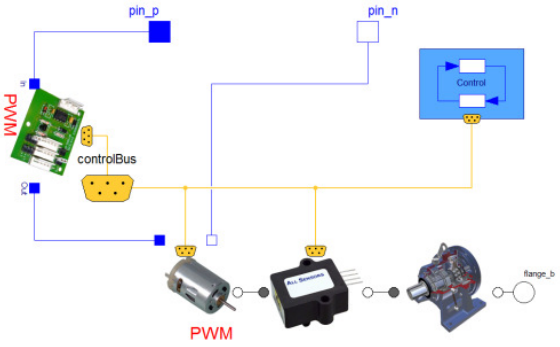

Fig. 7.: Modelica model of a single wheel's drive

Inside a drive block we use the aforementioned common virtual bus that connects the PWM module with the motor and hooks up the sensor model with the controller block, as well. Note that the generalized bus concept of Modelica also simplifies the analysis after the simulation, because such grouped signals can be found in one location and extracted from a bus at once.

3.5 The cascade controller model

Fig. 8 shows the cascade structure of our PI velocity and current controller. The main input regime for our controller block is the reference motor velocity, just like for its real counterpart. The output of the last block (*current_Controller*) is the *uRef* reference voltage level for the power electronics of the attached motor. Each signal in our models has standard SI units: the inputs from higher (behavioral) levels must also comply with this rule.

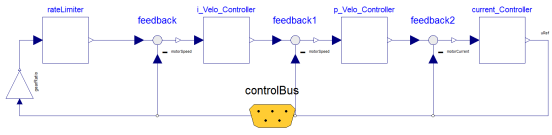

Fig. 8.: The cascade controller model

The parameterization of the controller uses the known physical parameters of the Huebner HAC 71.11.6 motor (such as power class, maximal and nominal torque, current and voltage values, etc.) and was fine-tuned by taking a wheel's inertia as nominal load and real measurements into account.

3.6 Central energy source

RAVON has a centralized 48V lead-acid battery pack as onboard energy source. The simulation of electrochemical processes within batteries poses

usually a problem: it needs rather big computational power. We chose a dynamic battery model [8] that uses an equivalent electrical circuit and a few parameters to approximate a general battery behavior in charge and discharge modes. It is based on a controlled voltage source and needs the charging / discharging currents as continuous variables:

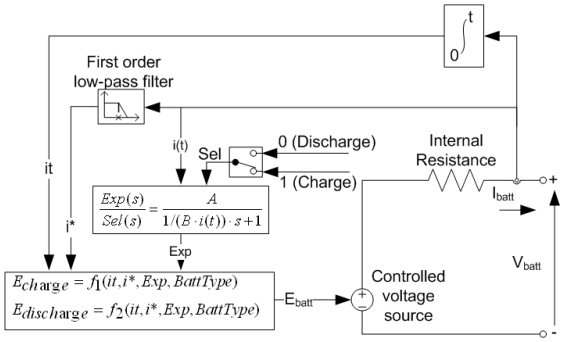

Fig. 9.: The equivalent circuit of the battery model

In the SimPowerSystems Toolbox of MathWorks [9] we found a real-time capable implementation of a general battery model. It provides a set of predetermined charge behavior of a parameterized Lead-Acid, Lithium-Ion, Nickel-Cadmium or Nickel-Metal-Hydride battery.

As it was mentioned before, we use a mirrored shared memory concept to let multiple simulators communicate over a network of computers in real time. We created a few new Modelica blocks to create an interface to the shared memory, which allows exchanging I/O signals from within Dymola with other simulators (e.g.: with Matlab / Simulink for the battery case). From Modelica we have to output a filtered current signal measured between the poles of our battery interface, and input the actual battery voltage signal that feeds our ideal voltage source there.

3.7 Motor model

In our Modelica model we apply four individual PWM-regulator blocks that adjust the battery voltage level for each motor according to their respective controller's *uRef* output.

As we mentioned before, in our first implementation we use the DC permanent magnet motor model from the standard *Modelica.Electrical* library. This includes the magnetic domain's electric equivalent equations for torque produced in the air-gap between the rotor and the stator.

3.8 Validating the drive parameters with measurement data

Our next task was to validate the complete virtual drive subsystem's behavior using real experiments with RAVON. The car was lifted up first so the wheels could turn freely. The reference speed of a single motor was set to 1%, 5%, 10%, 50% and 100% of the maximal rotor speed (2000 rev/min).

We could retrieve the effective current and actual speed levels from the real motor's *Bamobil-D3.2* digital controller from *UniTek GmbH* and save these signals using their *NDrive* communication software [10]. The stored signals stayed therefore at our disposal for later comparison with the simulated ones.

In our mechanical gear model there is a parametric block for friction losses. Using this block and the measured current signals at the aforementioned reference speeds we could setup the internal friction parameters. The error between the simulated and measured currents became less than 1% for the steady-state cases. The differences in the transient states are due to the different (partially unknown) control logic of the real *Bamobil* controller of RAVON. Minimizing this difference is one of our future tasks.

4 Results

The following figure shows the distributed simulation structure of the complete RAVON system:

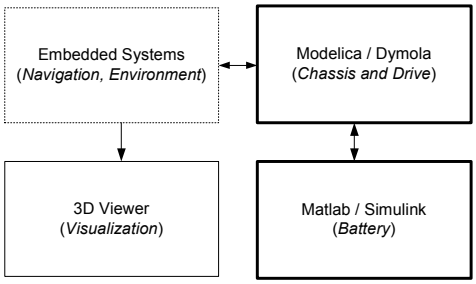

Fig. 10.: Our distributed simulation structure

Each main block of Fig. 10 can refer to one or more physical processes running on individual computers. The blocks on the left hand side are not discussed in details here.

After the model optimizing step of Dymola, the complete Modelica model of RAVON has 2930 time-varying variables and it can be simulated with a 4th order fixed-step Runge-Kutta ODE solver (*Rkfix4*) 25% faster than real time with a fairly small step length of 333 µseconds on a 3.3 GHz Core i7 CPU. Note that Dymola runs under Windows environment, thus it cannot be guaranteed to be hard real-time.

Our Modelica model of RAVON cannot be simulated alone without externally setting the respective input signals for the road inclination, friction values, reference wheel speeds and steering angles in the shared memory first. We are using a network-based distributed shared-memory interface to communicate with the high-level behavioral controllers (which include the embedded software) of RAVON that can be either simulated or real.

The embedded systems' behavioral tasks (navigation on unknown terrain, decision making based on image

processing, etc.) are therefore not part of our Dymola simulation. We are using Dymola only to model the dynamics of the chassis and the drive subsystems, thus using the co-simulation or a hardware-in-the-loop simulation it is possible to make an assessment about the embedded systems' functional safety.

In order to verify our Dymola models' capabilities, we implemented simple test cases first, using parameterized signal sources for steering angles, wheel speeds, etc. For testing the shared-memory communication, we also implemented a simple Matlab / Simulink environmental simulator using the VR toolbox. The user can control the virtual RAVON using a joystick and thereby set various variables interactively. The simulated RAVON chassis can be visualized in our own 3D Viewer, or we can use the Matlab VR toolbox (though the latter case has very poor performance if we don't simplify the geometry enough).

The most important dynamic signals of the drive subsystem can also be read from the shared-memory, thus we can analyze motor currents from Matlab, as well. Furthermore, if we want to add new details to our Modelica model, we can move the models of the control system or the drives partially into a separate Matlab environment, so the whole system simulation will remain real-time capable.

5 Outlook

Using modern processors there is still a reserve computing capacity for further detailed modeling. As the Rkfix4 solver of Dymola is inherently a single threaded process, increasing CPU clock frequencies can scale linearly with the simulation power using a Modelica model. The model's state variables are still well conditioned, so the solver's step length can eventually be increased up to 0.5 milliseconds, without losing too much numerical precision but winning simulation time.

5.1 Future tasks

As it was mentioned before, the detailed modeling of the permanent magnet synchronous motors including their controllers is one of our next tasks in order to analyze transient currents in the drives. This modeling task can also be done independently from the chassis model, so we can use other modeling environments than Dymola, as well.

We will have to include a thermal battery model, as the actual Matlab implementation simply neglects this domain. Because of charging and discharging currents and increasing ambient temperature of nearby components the real batteries get usually real warm during operation. For further safety checks the existing battery model has to be combined with a thermal model, so battery overheating scenarios could also be investigated.

6 Acknowledgement

We are very thankful for the Federal Ministry of Education and Research, Germany for supporting the ViERforES project (01 IM/08003) within their Hightech-Strategy IKT 2020 framework.

We would also like to express our appreciation to our partners at the RRLab of the Technical University of Kaiserslautern, who provided detailed data about their RAVON platform.

7 References

[1] C. Armbrust, T. Braun, T. Föhst, M. Proetzsch, A. Renner, H. Schaefer and K. Berns: RAVON – The Robust Autonomous Vehicle for Off-road Navigation; Proceedings of the IARP International Workshop on Robotics for Risky Interventions and Environmental Surveillance 2009 (RISE 2009); 12-14. January 2009 - Brussels, Belgium

[2] Modelica – http://www.modelica.org

[3] P. Fritzson: Principles of Object-Oriented Modeling and Simulation with Modelica 2.1; John Wiley and Sons, 2004, ISBN 0-471-471631

[4] Dymola – http://www.dynasim.se

[5] T. Juhasz, U. Schmucker: Automatic Model Conversion to Modelica for Dymola-based Mechatronic Simulation; Proc. of 6th International Modelica Conference, 3rd-4th March, 2008, Bielefeld, Germany, Vol. 2, pp. 719-726

[6] Rill, G., "First Order Tire Dynamics", Proceedings of the III. European Conference on Computational Mechanics Solids, Structures and Coupled Problems in Engineering, Lisbon, Portugal, 2006

[7] U. Riefenstahl: Elektrische Antriebssysteme; Teubner Verlag, 2006, ISBN 3-8351-0029-7

[8] O. Tremblay, L.-A. Dessaint, A.-I. Dekkiche: "A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles," Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE 9-12 Sept. 2007, pp. 284-289

[9] http://www.mathworks.com/access/helpdesk/help/toolbox/physmod/powersys

[10] UniTek GmbH – http://www.unitek-online.de