A Framework for Design, Test, and Validation of Electric Car Modules

Mehmed Yüksel, Mohammed Ahmed, Benjamin Girault, Timo Birnschein and Frank Kirchner

Abstract. This paper presents a practical framework and workflow for development and implementation of vehicle hardware and software components. It covers all activities from unit testing of single components to field experiments with the final constructed car. The described framework is based on the rapid control prototyping approach that is used for development and enables modularity in the design of subsystems. The framework was successfully used for the development and implementation of our new electric car concept (EO smart connecting car 2). Performance is analyzed through detailed simulations and experiments. The Framework reduces time and costs significantly for implementing component prototypes of the target system.

Keywords: Electric Vehicle, Hardware-in-the-loop, Software-in-the-loop, real-time control, rapid control prototyping, framework, path following, simulation, x by wire.

1 Introduction

In recent years, tremendous progress has been made in the field of intelligent vehicles for regular traffic. Research on vehicle automation for different automatic driving related tasks and assisted driving has increased during the last years.

Some early developments in electric vehicles include the "Lunar Roving Vehicle" (for Apollo programs 15, 16, and 17) for astronaut mobility on the moon [11].

M. Yüksel(✉) · M. Ahmed · B. Girault · T. Birnschein
DFKI GmbH - Robotics Innovation Center, Robert-Hooke-Straße 5,
28359, Bremen, Germany
e-mail: {mehmed.yueksel,mohammed.ahmed,benjamin.girault,timo.birnschein}@dfki.de

F. Kirchner
DFKI GmbH - Robotics Innovation Center, Department of Mathematics and Computer Science, University of Bremen, Robert-Hooke-Straße 1, 28359, Bremen, Germany
e-mail: frank.kirchner@dfki.de

In that vehicle, maneuverability is provided with all wheel steerable (AWS) and all wheel electric drive (AWeD) features. Hiriko [13] is another electric car, which also addresses urban mobility with its foldable construction (2.5–1.5 m), AWeD, and AWS features. Its design is a realization of the *CityCar* concept [12]. RObоMObil from the German Aerospace Center is one of the few autonomous electric car platforms with robotics features [9]. Another direction for autonomous car implementation is using a conventional car as technology carrier platform (e.g. Google Car [8] and BRAiVE car [7]).

In this work, we present a practical framework and workflow for development and implementation of vehicle hardware and software components. This covers all activities from unit testing of components to field experiments with the car. The presented platform was successfully used for the development and implementation of low, middle, and high level control layers.

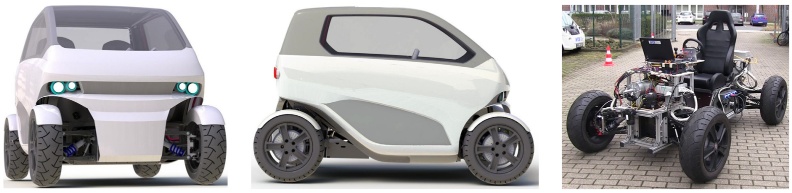

Fig. 1 EOscc2 (exterior design by David Grünwald) and its *SujeeCar* test platform [3]

2 EO Smart Connecting Car 2

EO[1] smart connecting car 2 (EOscc2) (Fig.1) is a high maneuverable all-electric micro-car designed for crowded cities with very confined parking spaces designed and constructed with robotic features. Its design is based on the fully functional technology demonstrator EO smart connecting car 1 (EOscc1) [1][2]. EOscc2 is developed to demonstrate the possibilities of a sideways driving, turn on the spot (Fig.3), shrink and dock to charging stations. EOscc2 is designed to be a *real* car with respect to practicality, safety, meeting the burden of regulatory compliance, and cost. Its chassis is a welded lightweight very high precision steel tube frame. The car has a modular active AWS. Four actuators are used for steering (between 32° to −92°) and changing the vehicle's ride height (within 16 cm). In addition, each wheel is equipped with a brushless DC (BLDC) wheel hub motor with integrated brakes for maximum efficiency. This results in a drive by wire vehicle that has none of the massive and bulky components. The shrink/fold feature results in a car with footprint of 3.4 m² (~36.6 ft²) through 1.4 m² (~15 ft²) for a minimum parking space. It is design for driving on test areas and public roads safely with a

[1] "EO" means in Latin "I go". The name belongs also to the fully electric vehicle family with robotic features, developed by DFKI since 2010, that includes EO smart connecting cars 1 and 2.

maximum speed of 70 km/h (~44 mph). To enter and exit the car in any folding position, the car is equipped with two *scissor-style* doors.

Autonomous functions are supported with several sensors and cameras. EOscc2 can dock to power outlets, other cars, or extension modules. EOscc2 is planned to park itself, dock to charging stations, undock, leave the parking space safely and realize a pick-up service for the driver within a parking lot. It is highly adaptive with a coupling mechanism for Car2Car, Car2Extender, or Car2ChargingStation [1][2][3].

To have a feasible system implementation for such a complex system, it should be kept as clearly arranged as possible. Therefore, the control layers "perception and planning" (high–level) and "actuation" (middle/low–level) of EOscc2 are separated. The actuation layer is developed in MATLAB/Simulink, tested on a rapid control prototyping (RCP) unit, and runs later on an embedded vehicle control unit (VCU). Precise multi-body dynamics simulation is used for rapid and cost efficient development [3][14]. The high-level control will run on the Robot Construction Kit (ROCK) framework[2].

3 Framework and Workflow Description

The presented framework used for the development and testing of electric cars consists of several tools. These tools are described as follows:

***Computation, modeling, and simulation software*:** it is used for the numerical computations (e.g. kinematics), control of the motors, modeling of the car's state machine. It is also used to analyze and visualize log data. For EOscc2 we used MATLAB/Simulink.

***Multi-body dynamics simulation software*:** to simulate the dynamics of the car and force/torque estimations. We used Adams/View that can also be interfaced with MATLAB/Simulink to perform co-simulations [4] using a detailed 3D model of the car.

***Rapid control prototyping solutions*:** For real-time applications, some software and hardware components are used to control different systems directly. In most of the commercially available RCP components, a Simulink generated model can be directly uploaded. The developer can monitor and record the parameters and modify them online. In our development, we used the dSPACE RTI-Libs and ControlDeskNG software components on the MicroAutoBox II RCP unit.

***Software development tools*:** used for the micro-controller programming. CooCox CoIDE was selected. It is specific for ARM Cortex MCU based microcontrollers and provides debugging tools.

***Documentation system*:** combining the use of *doxygen* for code documentation and the wiki and tracking platform *trac*.

[2] www.rock-robotics.org

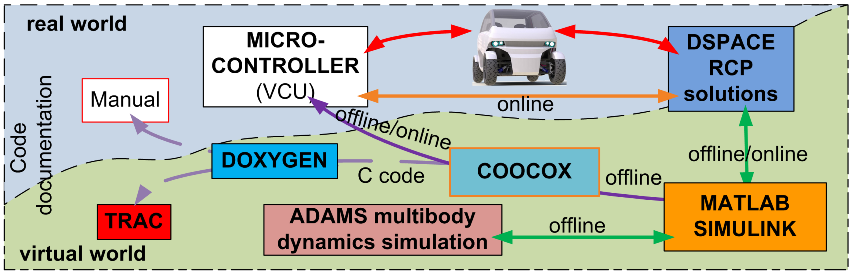

Fig. 2 Tools used within the framework divided into virtual (simulation) and real (hardware)

The main advantage of the chosen tools is their interactivity (Fig.2). They can re-use the code generated by other tools or allow online interaction between them or with the user/developer. Moreover, software-in-the-loop (SIL) tests can be performed. The code that will be later used for the car can be connected to simulation for qualitative validation and also for optimization. Hardware-in-the-loop (HIL) tests are made possible as well.

Thanks to its various I/O interfaces, the RCP unit can be easily connected to motors or input devices. However, the car- in its final configuration- is so complex that the provided interfaces (especially CAN interfaces) are not enough to control the whole system. Despite its many advantages, it was decided not to use the RCP unit in the final version of the car but to use an in-house designed vehicle control unit (VCU) running on a microcontroller.

The chosen RCP (and other most commercially available) components generate code directly for microcontrollers. In theory, this would allow skipping the software development and program the microcontroller as it is done with RCP unit. However, the functionalities are limited and do not cover the hardware specific interfaces, which are the most demanding parts of the programming.

3.1 Workflow Description

Car development can be split into three major levels: the low–level software and driver modules that are close to hardware (e.g., motors and electronic devices), the middle–level software that controls the whole car and finally the high–level algorithms for the autonomous and intelligent features. Those levels do not necessarily happen sequentially. Thanks to the SIL and HIL possibilities, high fidelity tests can be performed, independent from each other. This makes the whole development time shorter and in addition reduces the costs.

3.1.1 Low–Level Control of Hardware Components

The low–level software modules manage the communication with the BLDC wheel hub motor controllers and the electric actuators used for the steering, lifting, morphology, or the docking interface. The interfaces, using different CAN protocols (device specific, CANopen), were modelled and developed in MATLAB/ Simulink with integrated RCP tools for each device separately and tested on a specific test bench. The required CAN database container (dbc) files were written and imported to the model. The properties of the hub motors and actuators were measured and their control parameters were tuned. Because of its modularity and scalability, the same software module could be re-used within the main model controlling the whole car. Thus, the RCP unit could be used for device control and parameter tuning. The last step is adapting the protocols for the micro-controller, reusing some parts of the RCP unit's code that were already written in C.

3.1.2 Middle–Level Control Components

The middle–level software components cover the computation of the wheel kinematics and the vehicle drive modes. Based on the steering wheel position, accelerator, and brake values steer angles and wheel speeds are computed according to desired drive mode (double Ackermann, diagonal steering, sideways or turn on the spot) (Fig.3).

The wheel suspension kinematics are computed for actuator values to reach the desired steer angle. The algorithms for kinematics and the drive modes were implemented in MATLAB/Simulink. The correctness of the algorithms was verified through co-simulation and later with unit testing. The Simulink blocks were connected for the simulation model of the whole car (in Adams/View) and driving trajectories were tested for each drive mode. These simulations validated the algorithms but were also the base for high-level simulations (e.g., path following) and evaluations of the dynamics of the vehicle. The developed Simulink blocks were then integrated in a larger model and tested directly on SujeeCar (Fig.3) test platform (using the RCP unit). Finally, the RCP unit code was ported to the micro-controller and verified using unit testing. The micro-controller will replace the RCP unit but there will be still the possibility to combine both of them, and having for example the RCP unit in *silent* mode for data logging.

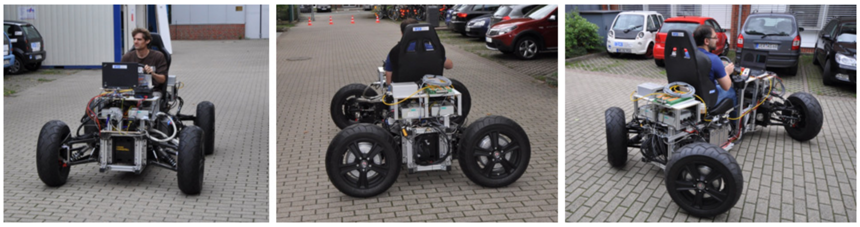

Fig. 3 Ackermann, sideways and turn-on-the-spot drive modes

3.1.3 High Level Algorithms

To provide the driver–assistance systems and go towards the car autonomy, several high level algorithms need to be developed and tested. This includes in particular a cruise control and path following system [3]. It can be directly deployed using the modules previously developed and interfaced with the multi-body simulation software.

As an example, a road-tire interaction model is used with Adams/View to obtain a realistic movement of the car and be able to perform closed-loop testing. Once the algorithm was evaluated and tuned, thanks to the co-simulation, it can be tested on SujeeCar. The first step is an open-loop test in which steering wheel and wheel speed values are generated and loaded on the RCP unit. The scenario can then be run on the car. Later, the algorithm will be tested in closed-loop on the car. The results of the drive tests can as well be recorded and replayed with the simulation software, to verify for instance the accuracy of the dynamics model.

3.2 *EOscc2 Test Platform (SujeeCar)*

In addition to the several test benches for the wheel hub motors and brake system, a prototype of the car was built to be able to experiment with a real system at an early stage of the development (Fig.1). This was possible as the axles were early finished and fully functional thanks to their modular design. Thus, constructing the demonstrator platform mainly consisted of connecting these two axles with T-slot aluminum profiles. The length of the prototype corresponds to the case where EOssc2 is unfolded. Batteries are placed under the driver's seat and the electronic devices are gathered above the axles. To control the car, a laptop is used to interact with the RCP unit.

4 Performance Analysis Tests

4.1 *Drive Mode Change*

During drive mode changes or folding, it is important that the wheels roll at the same time to follow the movement. Since the axis of rotation of the steering movement is not perfectly in the middle plane of the wheel, it performs a circle with a so-called *scrub* radius (91 mm for EOscc2). Simulations were done to estimate the actuator forces while folding or switching between Ackermann and Turn-on-the-spot modes. It was tested with and without the wheels rolling at the same time. The results (Fig.4) were used to verify the correctness of the equations of movement.

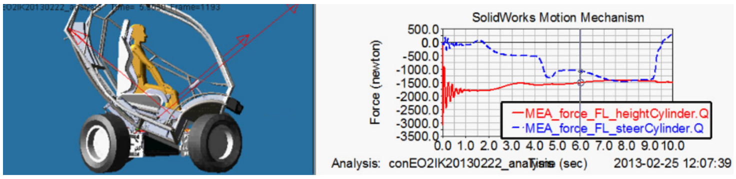

Fig. 4 Estimation of the forces applied to actuators while folding or switching between Ackermann and Turn-on-the-spot steering modes

4.2 Path Following Algorithm

A path following module was designed and implemented [3]. It consists of a proportional input-scaling feedback controller that uses the forward velocity and angular acceleration of the vehicle as control inputs. For this controller, a kinematic vehicle model is used to map from the path curvature specified by the given path to the vehicle's actual steering angle. This controller was interfaced to the rigid-body simulation software and tested in closed loop for different trajectories.

In this simulation, EOscc2 kinematic parameters are: 1.9 m for the wheelbase, the axle track equals 1.35 m, and the wheel radius is 0.325 m. The controller parameters were empirically tuned. The car is commanded to move with constant forward speed of 2.8 m/s. A desired path for the standard test track described in ISO3888-2 [5] is used. From the results (Fig.5), it can be seen that the system converges to the reference trajectory asymptotically. Once the vehicle converges, the vehicle follows the trajectory very closely. The convergence can also be seen in the error graph. These results show that the controller does not attain extremely large values, and is bounded which are essential properties for the real systems.

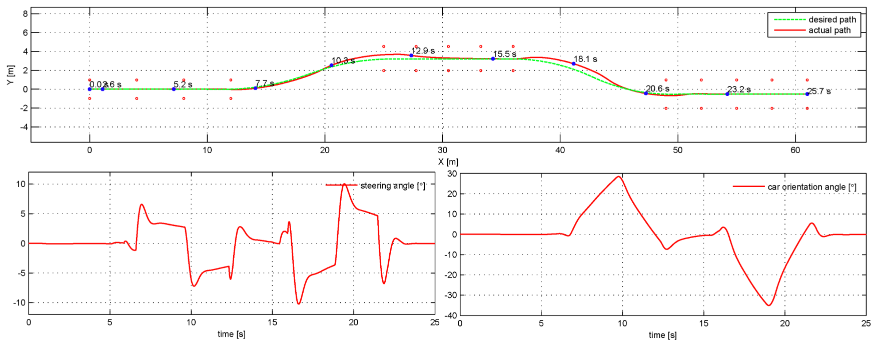

Fig. 5 Path following controller in Adams/View – MATLAB/Simulink cosimulation results for ISO3888-2 double lane change test track

4.3 Line Change Experiment

Because of lack of standard tests for autonomous vehicle driving skills, we borrowed tests from the conventional automotive industry as the Carnegie Mellon Red Team did for DARPA Grand Challenge [10]. We chose the same test method for EOscc2, in order to be able to compare the future results.

To measure maneuverability and stability of the vehicle with influence of different drivers, a lane change test is performed. A test track was constructed according to ISO 3888 Part 2 (Fig.6). For three different drivers, the test scenario was that each driver begins the run in the right lane, swerves into the left, and then immediately cuts back into the right. During the experiment, car and driver data were logged by the RCP unit (67 parameters in total) and the drives were recorded with four cameras (Fig.6). The data can be plotted or exported to the mechanical simulator for replay. Hence, the dynamic parameters of the car can be estimated and the accuracy of the multi-body simulation can be verified. A sample of the results is shown in the Fig.6 for trajectories of three drivers test run.

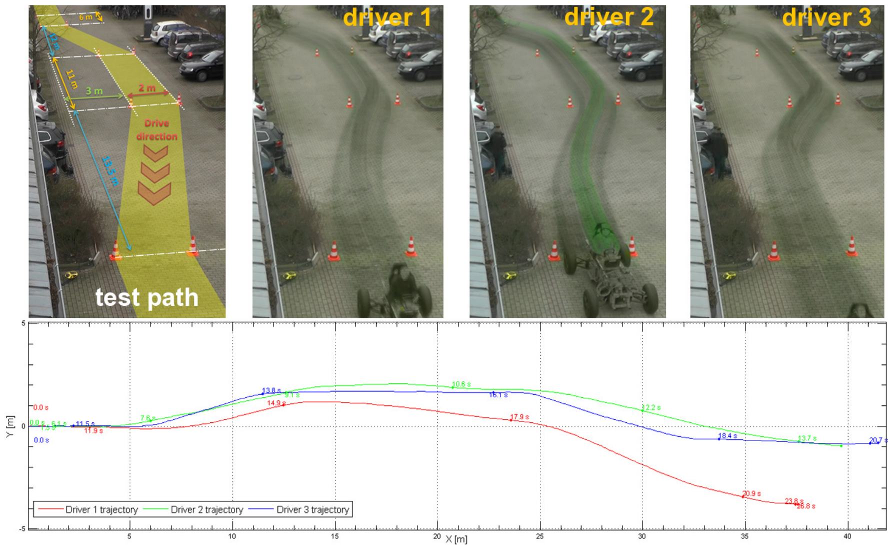

Fig. 6 Lane change experiments for three drivers and corresponding trajectories from simulation replay

5 Conclusion

In this paper, we present a practical framework and workflow for development and implementation and evaluation of vehicle software components. It covers all activities from unit testing of single components to field experiments with the real car. The described framework is based on RCP approach that is used for system development and enables modularity in the design of subsystems. The framework was successfully used for the development and implementation of actuators and motors low–level control in addition to middle–level algorithms. The performance of the designed modules is demonstrated through detailed simulations and experiments. In the conducted simulations, the car software modules are used in a SIL fashion and in the real experiments, the software modules are interfaced to the hardware, tested and verified.

From the presented performance analysis tests of the framework to validate hardware and software components and systems of the car, it is verified that this framework is an effective and adaptive solution as a development and test environment. It reduces effort, time, and costs significantly for implementing component prototypes of the target system. Because of the benefits of this framework and its capabilities as a realtime hardware interface, it is selected for the integration phase of the EOscc2 car as well as the development and optimization of most software components for the car control. The system will be used for future experiments on autonomous driving and for driver assistance modules especially addressing users (driver) modeling and adaption.

Acknowledgment. We thank all team members of the ITEM project [6] (in alphabetical order: Christian Oeckermann, David Grünwald, Haci Bayram Erdinc, Janosch Machowinski, Roman Szczuka, Sujeef Shanmugalingam, Sven Kroffke, Yong-Ho Yoo) who provided valuable comments, ideas, and assistance, which were essential to this study. This work is developed for the EO car which is evolved in the subproject ”Innovative Technologies Electromobility (ITEM)” of main project ”Model Region Electric Mobility (PMC)” – Module 2 ”Intelligent Integration of Electric Mobility” and funded by the German Federal Ministry of Transport, Building and Urban Development (Grant Nr. 03ME0400G). Program coordination is carried out by the National Organization Hydrogen and Fuel Cell Technology (NOW GmbH).

References

[1] Jahn, M., Schröer, M., Yoo, Y.-H., Yüksel, M., Kirchner, F.: Concept of actuation and control for the EO smart connecting car (EO scc). In: Su, C.-Y., Rakheja, S., Liu, H. (eds.) ICIRA 2012, Part I. LNCS (LNAI), vol. 7506, pp. 87–98. Springer, Heidelberg (2012)

[2] Birnschein, T., Kirchner, F., Girault, B., Yüksel, M., Machowinski, J.: An innovative, comprehensive concept for energy efficient electric mobility - EO smart connecting car. In: ENERGYCON 2012. IEEE (2012)

[3] Ahmed, M., Yüksel, M.: Autonomous Path Tracking Steering Controller for EO Smart Connecting Car. In: Proceeding of the World Congress on Multimedia and Computer Science 2013 (ICIAR-13). IEEE (2013)

[4] Ahmed, M., Yoo, Y.-H., Kirchner, F.: A cosimulation framework for design, test and parameter optimization of robotic systems. In: Joint Conference of the 41st International Symposium on Robotics and the 6th German Conference on Robotics (ISR/ROBOTIK 2010). VDE Verlag (2010)

[5] Lundahl, K., Åslund, J., Nielsen, L.: Vehicle dynamics platform, experiments, and modeling aiming at critical maneuver handling. Technical report, Linköping University (2013)

[6] ITEM–Project Web Page, http://robotik.dfki-bremen.de/en/research/projects/item.html (accessed January 08, 2014)

[7] Broggi, A., et al.: Autonomous vehicles control in the VisLab Intercontinental Autonomous Challenge. Annual Reviews in Control (2012)

[8] Guizzo, E., How Google’s Self-Driving Car Works. IEEE Spectrum, http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/how-google-self-driving-car-works (accessed January 24, 2014)

[9] ROboMObil-System Architecture and Safety - DLR, http://www.dlr.de/rm/desktopdefault.aspx/tabid-8001/13698_read-34737 (accessed January 27, 2014)
[10] Urmson, C., Whittaker, W., Harbaugh, S., Clark, M., Koon, P.: Testing driver skill for high-speed autonomous vehicles. Computer 39(12) (2006)
[11] Wright, M., Jaques, B., Morea, S.: A Brief History of the Lunar Roving Vehicle. NASA: Marshall Space Flight Center (2002)
[12] Mitchell, W.J., Borroni-Bird, C., Burns, L.D.: Reinventing the Automobile: Personal Urban Mobility for the 21st Century. The MIT Press (2010)
[13] Hiriko, driving mobility, http://www.un.org/esa/dsd/susdevtop-ics/sdt_pdfs/meetings 2012/statements/espiau.pdf (accessed January 26, 2014)
[14] Ahmed, M., Oekermann, C., Kirchner, F.: Cosimulation Environment for Mechanical Design Optimization with Evolutionary Algorithms. In: International Conference on Artificial Intelligence (ICAI 2014). IEEE (2014)