Model-Based Design of Embedded Control Software for Hybrid Vehicles

Tizar Rizano
University of Trento
Trento, Italy
tizar.rizano@unitn.it

Roberto Passerone
University of Trento
Trento, Italy
roberto.passerone@unitn.it

David Macii
University of Trento
Trento, Italy
david.macii@unitn.it

Luigi Palopoli
University of Trento
Trento, Italy
luigi.palopoli@unitn.it

***Abstract*—In the last decades, model based methodologies have become the mainstay of research on embedded systems development. The availability of mature computer aided tools and of well–settled industrial practices has promoted the adoption of these methodologies in large companies, which are able to amortize the cost on a large volume of products. On the contrary, the cost of software licenses and of staff training often discourages their application in small and medium enterprises. In this paper, we present a model based methodology entirely based on the adoption of open source software tools. We have applied this methodology to a real case study provided by our industrial partner proving its effectiveness.**

***Keywords*-design methodology; digital control; automotive engineering;**

I. Introduction

Model-based design methodologies are emerging as some of the most convincing alternatives for the development of embedded software compared to the classical methods based on low-level languages and intense prototyping activities [1], [2]. In particular, model-based design promises to shorten the distance between high level control algorithms and their actual implementation. The idea is to start from abstract models, closer to those traditionally used by mechanical and electrical engineers, and to come to the actual code through automated or semi-automated transformations (refinements) which preserve already verified properties. Despite the higher productivity that can be achieved with these methods, there are large industrial sectors (especially the small and medium enterprises) whose production volumes do not justify the significant investment required for tools and training. This picture can potentially be changed with the emergence of no/low cost tools, which, if complemented by appropriate methodologies, could preserve the advantage of model-based development without its cost. In this paper, we describe a realistic case study conducted in collaboration with the Centro Ricerche Ducati (CRD), where the researchers have undertaken the design of one of the most crucial components of a vehicle: the Engine Control Unit. The tools employed are open source and include Scicoslab/Scicos[1] for modeling the system and the control algorithms, and the open source OSEK-compliant operating system Erika Enterprise[2]. This paper describes the modeling phase and shows the simulation results. We are currently conducting the implementation and test phase with the actual hardware.

[1]http://www.scicoslab.org/
[2]http://erika.tuxfamily.org//

A. Problem description

The system considered in this paper is a series hybrid vehicle. The most evident difference between a hybrid and a conventional vehicle is the presence of two motors (an electric motor and an internal combustion motor) instead of one. In a series hybrid vehicle, the internal combustion motor is exclusively used to re-charge a battery. The propulsion is provided by an electrical motor which utilizes the energy stored in the battery. The advantage of this approach is that the internal combustion motor control can be optimized with respect to power consumption and emissions. The drawback is the efficiency loss in converting the energy twice (chemical→electrical→mechanical) as opposed to the single conversion (chemical→mechanical) used in traditional automotive technology.

The problem we present in this paper is to design the controllers for the hybrid vehicle based on some design goals and constraints. The controllers that have to be designed are: a speed controller that sets the electrical motor at a reference speed, an ignition controller that controls the torque production and a lambda controller that manages the gas emission.

The design must satisfy three important constraints:

- **low target price of the vehicle**, which dictates the components used in the vehicle and it also influences the model of the vehicle;
- **low production cost** for Small Medium Enterprise (SME), which limits the cost of tool acquisition and staff training;
- **short time to market**, which emphasizes the need for rapid prototyping and/or simulation.

II. Methodology

Based on the constraints in the previous section, we decide to use a model-based methodology to design the controller for the hybrid vehicle. The methodology is inspired by the general principles of Platform-Based Design [3], [4]. It has three important phases, which are: **functional design**, **architectural design** and **mapping**.

978-1-61284-820-4/11/$26.00 ©2011 IEEE

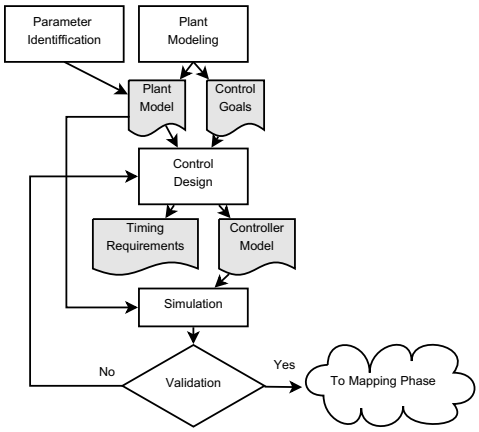

Figure 1. Overview of functional design phase

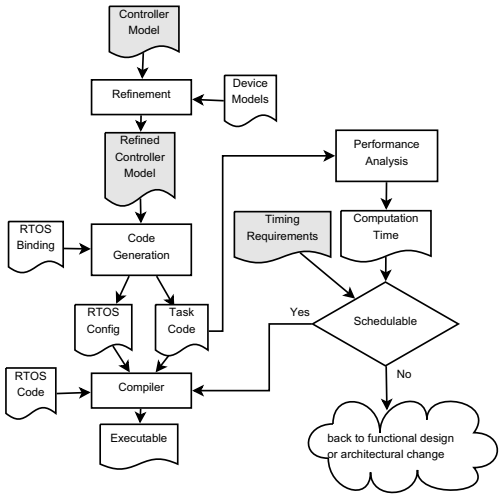

Figure 2. Overview of mapping phase

In the functional design phase we produce a mathematical model that describes the plant (the hybrid vehicle), the controllers (speed, ignition and lambda controllers) and the timing constraints. The mathematical model is normally called functional model.

For the plant model, we start by constructing a generic model of each component used in the vehicle (e.g., the motors, the intake manifold, the battery, etc.) based on models found in the literature. The next step is to identify parameters of the actual components for the generic model. This step is carried out by collecting data from the technical documentation of the components and by running experiments. The controller models are designed using control engineering techniques to satisfy certain control goals. To ensure all control goals are met, we run simulations on the functional model. Figure 1 shows an overview of the functional design phase. The boxes represent processes, whereas the shaded documents represent functional design artifacts.

In the architectural design phase, we decide on a platform on which the controllers will run. This phase produces the architectural model of the hardware and the real time operating system (real time tasks, timers, semaphores, etc.). The architectural design phase will not be described in detail in this paper.

The final phase is the mapping phase (Figure 2) where the functional model is mapped onto the architectural model. The non shaded documents represent models produced in the functional design phase and architectural design phase. In the mapping phase there are three main steps:

- *Refinement*. In this step, the controller models are refined by adding details about the device. For example, a direct connection between the controller model and the plant can be replaced with a particular communication method (e.g., RS232 or CAN bus)
- *Mapping*. In this step, the refined model is mapped onto the architectural model (real time tasks)
- *Code Generation*. In this step, the mapped model is converted into source code that can be compiled and executed on the target hardware
- *Performance and scheduling analysis*. In this step, we run an analysis on the tasks running inside the hardware to compute performance and scheduling metric (e.g., Worst Case Execution Time (WCET))

We need a set of tools that can support our methodology. The Matlab tool suite is the de facto standard for the development of mathematical models of hybrid systems, and can be easily enriched with additional features that carry out specialized operations required in distinct areas of application. This makes Matlab/Simulink the preferable tool suite for our methodology. However, in this paper we want to emphasize our methodology for SME. Therefore, we need a different tool suite that has lower acquisition cost.

For the rest of the paper we use Scicoslab/Scicos which is available as an open source software packages, as a tool suite for our methodology. This tool is suitable to evaluate the methodology. While it may have some drawbacks from a usability point of view, its strength lies in the rigorous mathematical underpinnings.

One of the challenges of using Scicoslab/Scicos is describing event-driven dynamics. Commercial tools, such as Matlab/Simulink, facilitate this task by providing a set of modeling primitives specialized for hybrid systems (Stateflow tool). In Scicos, we have to explicitly provide a "low level" modeling of hybrid dynamics. As an example, consider the problem of detecting state switches in a cylinder of a four-stroke engine. Events have to be generated every time the piston reaches the dead centers. In Scicos, the events can be manually generated based on the values of the angular position of the crankshaft.

III. Hybrid Vehicle

A very high-level view of the system is shown in Figure 3. We can recognize four macro-components:

- the internal combustion engine, which consists of the intake manifold, the bypass valve, and the 1-cylinder engine;
- the catalyst pipe;
- the electrical power-train, which consists of the alternator, the rectifier, the battery and the electrical motor;
- the vehicle dynamics.

The internal combustion engine receives as an input the throttle valve angular position and the duty cycle of the bypass valve and produces the torque on the crankshaft and the air-to-fuel ratio. The catalyst takes as an input the air-to-fuel ratio and produces the oxygen storage, which can be translated into the concentration of exhausted polluting gases. The torque is an input to the power train consisting of the alternator, the rectifier and the DC motor. The output of the electrical power train is the torque applied to the wheels of the vehicle. The torque, along with load torques from the environment, is an input to the vehicle dynamics and determines its motion. Modeling the vehicle dynamics is not in the scope of this paper.

In this paper we would like to show how to perform the functional design for a particular component in the system. We selected the battery as an example since it is one of the most complex components in our model.

The vehicle uses lead-acid batteries due to their low cost. The battery pack consists of eight 12V batteries arranged as two parallel series of 4 batteries each. The total capacity of the battery pack is about 84Ah, whereas the output voltage is in the order of 48V. We developed our battery model based on the model by Ceraolo and Barsali [5], [6]. The considered battery model describes the dynamics of a lead-acid battery when it is charged and discharged in different operating conditions.

One of the most important parameters of a battery is its State of Charge (SOC). In fact, the SOC is an indicator of how full the battery is compared to its maximum capacity at temperature θ. It is given by the following equation:

$$SOC = 1 - Q_e / C(0, \theta)$$

where $C(I, \theta)$ is the function that computes the battery capacity and $Q_e = \int_0^t -I_m(\tau)\,\mathrm{d}\tau$. Based on the mathematical model of the battery we construct the Scicos model that is shown in Figure 4. In the battery model, the charge/discharge current is regarded as an input (connected to both the alternator output rectifier and the inverter powering the electrical motor), whereas the voltage is regarded as an output (connected to the rectifier). Positive current values indicate that the battery is being charged and negative current values indicate that it is discharged. The parameters for the battery model are identified from the data-sheet of the lead

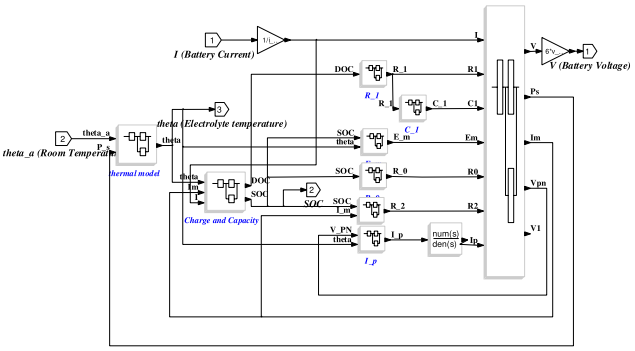

Figure 4. Scicos Model of the battery

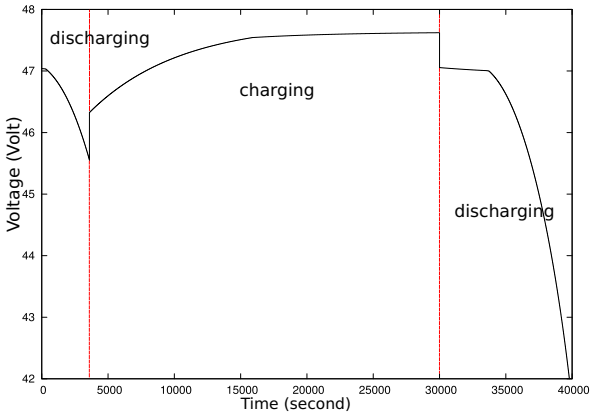

Figure 5. Simulation of charging and discharging the battery. The battery is discharged at -25.2 A and charged at 7.14A for 500 minutes and finally discharged at -25.4A

acid battery and by running experiments. Figure 5 shows simulation results for charging and discharging the battery.

Even though we only presented the battery as an example, we have built the complete model describing the hybrid vehicle and the three controllers. Having a complete model of the system and the controllers gives us flexibility of simulating the complete system or only parts of the system. Experiments shows that the simulation of the complete system for 10 seconds requires 1.4 hours. However, the simulation of the internal combustion engine model for 60 seconds requires around 5 seconds. The long experiment time for the complete model is due to the complexity in the electrical model, especially the alternator and rectifier that requires very frequent switches. We can improve this by selecting simpler models for the alternator and rectifier with the drawback of lowering the quality of the models.

Figure 6 shows the simulation results of the electrical motor model and the battery model. The motor is set to run at 590 RPM for 9 seconds and at 300 RPM afterwards.

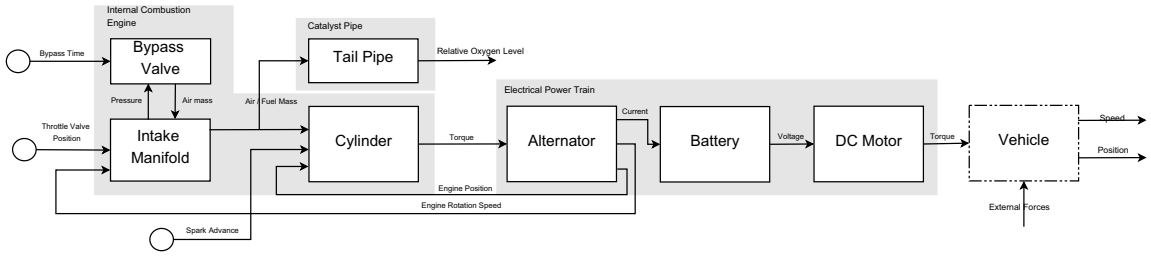

Figure 3. Overview of The Model

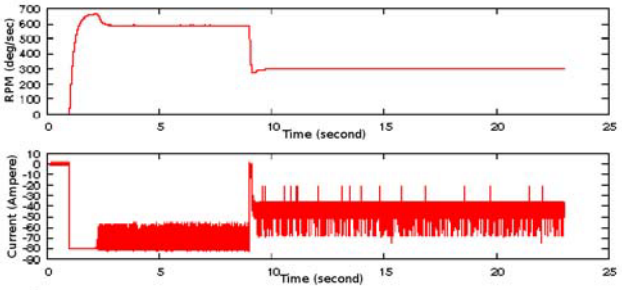

Figure 6. Scicos Simulation of The Motor and The Battery. Initially, the motor is set to run at 590 RPM. After 9 seconds, the motor will run at 300 RPM

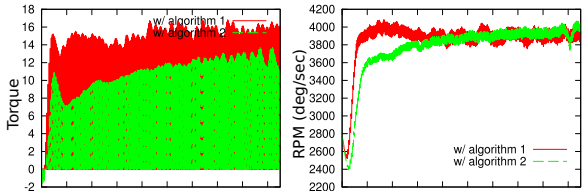

Figure 7. Comparison of Scicos simulation result between two control algorithms. The RPM set point is set at 4000 RPM

In the bottom figure we observe the current drain of the battery. Negative current means that the current is flowing out of the battery, i.e. into the electrical motor. Reducing the motor speed also reduces the current required to run the motor.

The model based methodology gives us flexibility in designing and testing different control algorithms and choosing a control strategy based on the simulation of the system. Figure 7 shows a comparison between two control algorithms. Algorithm 1 is provided from CRD whereas algorithm 2 is a non-linear control scheme proposed by Balluchi et. al. [7]. As we can see in Figure 7, the steady state precision of the latter scheme is better and the transient is more regular. However, the price to pay is the use of additional sensors (i.e., the pressure sensor in the intake manifold), which for the considered application is not an affordable cost.

IV. Conclusions

In this paper we have presented a model based methodology for embedded systems development. The methodology is based on the use of open source software tools, and it holds the promise of being applicable by small and medium enterprises for its limited cost. In order to show the concrete applicability of the methodology, we have modelled the electromechanical components of a hybrid vehicle and the digital controller for the RPM control and for emission control. This model was of challenging complexity for its hybrid dynamics and for the different time constants. The next steps of this case study will be to generate the code and map it onto different architectures. This step is crucial to show the ability of our methodology to perform architectural exploration and to port a project to a different architecture with a very small effort.

References

[1] B. Selic, “The pragmatics of model-driven development,” *Software, IEEE*, vol. 20, no. 5, pp. 19–25, 2003.

[2] S. Edwards, L. Lavagno, E. Lee, and A. Sangiovanni-Vincentelli, “Design of embedded systems: Formal models, validation, and synthesis,” *Proceedings of the IEEE*, vol. 85, no. 3, pp. 366–390, 1997.

[3] A. L. Sangiovanni-Vincentelli, “Defining platform-based design,” *EE Design of EE Times*, February 2002.

[4] K. Keutzer, S. Malik, A. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli, “System-level design: Orthogonalization of concerns and platform-based design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 12, pp. 1523–1543, Dec. 2000.

[5] M. Ceraolo, “New dynamical models of lead-acid batteries,” *IEEE Transactions on Power Systems*, vol. 15, no. 4, pp. 1184–1190, 2000.

[6] S. Barsali and M. Ceraolo, “Dynamical models of lead-acid batteries: Implementation issues,” *IEEE Transaction on Energy Conversion*, vol. 17, no. 1, pp. 16–23, 2002.

[7] A. Balluchi, L. Benvenuti, M. Di Benedetto, T. Villa, and A. Sangiovanni-Vincentelli, “Idle speed control–A benchmark for hybrid system research,” in *Proc. of the 2nd IFAC Conference on Analysis and Design of Hybrid System (ADHS06), Alghero, Italy*, 2006.