Model-Based Development of In-Vehicle Software

Extended Abstract

Mirko Conrad, Heiko Dörr
DaimlerChrysler AG, Germany
{mirko.conrad | heiko.doerr}@daimlerchrysler.com

Mathematical modeling, which already is established for a long time in many engineering domains, now also gains strongly in importance in the development of embedded software. In the automotive sector [9], modeling is used on the one hand for the conceptual anticipation of the functionality to be realized (open/closed loop control, monitoring) and, on the other, for the simulation of the behavior of real physical systems (plant, environment).

Modeling is usually carried out with commercial modeling and simulation packages such as MATLAB/ Simulink/Stateflow [6]. They support the development and definition of system/software components, their connections and interfaces by semiformal graphical models using editable, hierarchical block diagrams and extended state transition diagrams (state charts) and provide the necessary means of description, computation techniques and interpreters/compilers. Graphic editors permit an intuitive development and description of complex models. Hierarchically structured modularity is used in order to control complexity. A model consists of function blocks with fixed in- and outputs. Function blocks are connected within the block diagram by directed edges between their interfaces, which describe signal flows. With this, they represent equations in the mathematical model, which relate the interface variables of different components. The connection lines represent causally motivated directions of action, which define the outputs of one block as the inputs of another. Components within the hierarchy can aggregate other components or be elementary (cf. [5,7]).

Such models can be simulated, i.e. executed. During simulation the calculation causality follows the defined directions of action until the entire model has been processed. For solving the equations described by the model at certain instants of time there is a range of different predefined solvers available. Variable-step solvers are of special importance for the thorough modeling of physical systems and are used primarily for modeling the plant and the environment. For the development of embedded software fixed-step solvers are used, which represent a necessary prerequisite for an efficient code generation.

The modeling style described is used extensively within the scope of model-based development of embedded in-vehicle software [4,8]. Typically, both an executable model of the control software (functional model) and a model of the surrounding system (plant model) and its environment (environment model) are created early in the development cycle and are simulated together. This way, it is possible to model even highly complex automotive systems with a high degree of detail at an acceptable calculation speed and to simulate their behavior closely to reality. While the plant/environment model is gradually replaced during the course of development by the real system and its real environment, the functional model serves as a blueprint for the implementation of embedded software on the control unit through code generation.

One characteristic of the model-based development paradigm is the fact that the functional model not only specifies the desired function but also provides design information and finally even serves as the basis of the implementation by means of code generation. In other words, such a function model offers specification aspects as well as design and implementation aspects. In practice, these different aspects are reflected in an evolution of the functional model from an early specification model via a design model to an implementation model and finally its automatic transformation into C code (model evolution). In comparison to traditional software development with a clear separation of phases, in model-based development a stronger coalescence of specification, design and implementation phases can be noted. Moreover, one and the same graphical modeling notation is used during the consecutive development stages. The seamless utilization of models facilitates a highly consistent and efficient development.

In central vehicle domains, model-based development has become established as the standard paradigm for the development of control unit software. Significant advantages can be found in the consistency of the modeling notations and tools used, in the gains in

3-9810801-0-6/DATE06 © 2006 EDAA

efficiency due to using code generation [10] and in the fact that testing can be started already on model level [2]. According to user reports there are increases in efficiency of 20-50% due to model-based development with code generation in comparison to traditional software development as well as a more rapid increase of the maturity level of developed functions. Current research topics include the model-based development of safety-related in-vehicle software [3] as well as an tool-supported integrated trace-management for the different development artifacts [1].

In general, the paradigm of model-based development does not depend on the existence of the model type mentioned above. Alternatively, UML-models, for example, can be used as well. This tutorial, however, focuses on the form of model-based development with MATLAB/ Simulink/Stateflow.

References

[1] Altheide, F., Dörr: Integrating Simulink/Stateflow into a System Development Environment. 6. NASA-ESA Workshop on Product Data Exchange - Open Standards for Model-Based Development, Friedrichshafen, 2004

[2] Conrad, M.: Modell-basierter Test eingebetteter Software im Automobil - Auswahl und Beschreibung von Test-szenarien. Deutscher Universitätsverlag, Wiesbaden, 2004

[3] Conrad, M., Dörr, H.: Einsatz von Modell-basierten Ent-wicklungstechniken in sicherheitsrelevanten Anwendungen - Herausforderungen und Lösungsansätze. 2. Workshop Model-Based Development of Embedded Systems (MBEES'06), Dagstuhl, 2006 (to appear)

[4] Conrad, M., Fey, I., Grochtmann, M., Klein, T.: Modell-basierte Entwicklung eingebetteter Fahrzeugsoftware bei DaimlerChrysler. Informatik Forsch. Entw. (2005) 20:3-10

[5] Jersak, M., Cai, Y., Ziegenbein, D., Ernst, R.: A Trans-formational Approach to Constraint Relaxation of a Time-driven Simulation Model. Proc. 13. Intl. Symposium on System Sythesis, Madrid, 2000

[6] MATLAB, Simulink, Stateflow, The MathWorks Inc., www.mathworks.com/products/matlab, www.mathworks.com/products/simulink, www.mathworks.com/products/stateflow

[7] Merz, R., Litz, L.: Objektorientierte Mathematische Modellierung - Generische Methoden bei komplexen dynamischen Systemen. Informatik Spektrum, 23 (2000) 2: 90-99

[8] Rau, A.: Model-Based Development of Embedded Auto-motive Control Systems. Dissertation, Universität Tübingen, 2003.

[9] Schäuffele, J., Zurawka, T: Automotive Software Engineering. Vieweg Verlag, 2003

[10] Stürmer, I., Weinberg, D., Conrad, M.: Overview of Existing Safeguarding Techniques for Automatically Generated Code. 2. Intl. ICSE Workshop on Software Engineering for Automotive Systems (SEAS'05), St. Louis, 2005