AutoMoDe – A Transformation Based Approach for the Model-based Design of Embedded Automotive Software

U. Freund, P Braun, J Romberg, A Bauer, D Ziegenbein, P. Mai

AutoMoDe – A Transformation Based Approach for the Model-based Design of Embedded Automotive Software

U. Freund[1], P. Braun[2], J. Romberg[3], A. Bauer[3], D. Ziegenbein[4], P. Mai [5]

1: ETAS GmbH, Borsigstrasse 14, 70469 Stuttgart
2: Validas AG, Lichtenbergstrasse 8, 85748 Garching
3: Technische Universität München, Boltzmannstrasse 3, 85748 Garching
4: Robert Bosch GmbH, Robert-Bosch-Strasse 2, 71701 Schwieberdingen
5: PMSF IT Consulting Pierre R. Mai, Ludwig-Thoma-Strasse 11, 87724 Ottobeuren

Abstract: The AutoMoDe approach manages the complexity of embedded automotive systems by employing a stream-based development paradigm which is specifically tailored to embedded automotive real-time systems. In this paper the tailoring process is explained by transforming a traction control system from a stream-based model to an embedded real-time software model and afterwards integrating the software model on an embedded automotive rapid development hardware.

Keywords: AutoFOCUS, ASCET, INTECRIO, Refinement, Embedded Software.

1. Introduction

The amount of automotive electronics has grown considerably in the last forty years. However, the resulting complexity has reached a level which can hardly be handled by current modeling means. The impact of the complexity to automotive software design has been foreseen more than a decade ago. Several projects have been started since then trying to define methods and tools to manage the complexity of embedded automotive control software by using high-level software structures. For example, ten years ago DaimlerChrysler started to tailor ROOM concepts for body electronic systems in the TITUS project [12]. In 1998, the French automotive industry started the AEE[1] project [13] which was finished in 2001. Both projects resulted in a common European research effort called EAST/EEA[11]. This project finished in summer 2004 and provided as one of the results an automotive architecture description language (ADL)[1]. Last but not least, the AUTOSAR development partnership [14] provides with the software component description a framework for high-level software description. The described approaches for high-level design, along with commercially available tools for detailed (control) software design, provide a wealth of proven abstraction mechanisms for all layers of the embedded software design chain. Unfortunately, each modeling approach is currently restricted to particular aspects of embedded automotive software design like networks, control-algorithms, or software architecture. An accepted and mature modeling framework integrating the different aspects of embedded software development is still missing. Hence, there still remains a lot of manual integration work to be done when designing an ECU-network. The manual integration work is centered around the questions how to integrate new functionality in existing E/E-architectures or how to optimize an E/E-architecture for a given number of functions. As a rule, designing an E/E-architecture appears as a mixture of both.

AutoMoDe is a joint research project consisting of members of the Software & Systems Engineering group at the Technische Universität München, Validas AG, ETAS GmbH, Robert Bosch GmbH, and BMW AG. The overall goal of the project is to tackle the integration challenge in model-based automotive development, and develop an integrated methodology for automotive control software based on custom, problem-specific design notations with an explicit formal foundation.

2. AutoFocus Concepts

The concepts developed in the AutoMoDe project are integrated and reflected in the modeling & development framework AutoFOCUS. In this section we briefly detail its semantics and the employed description techniques.

2.1 Semantics

Generally, the AutoFocus tool provides an embedded systems specific realization of the FOCUS framework[17]. Since its conception in 1996, it has been successfully used for modeling a large number of case studies, including, e.g., an automotive body-electronics system described in [7]. AutoFOCUS uses a so-called stream-based approach where a system is considered as components exchanging messages explicitly over streams. Basically, a stream is an ordered set of typed messages. Messages in turn are time stamped with

[1] Architecture Electronic Embarque

respect to a global, discrete time base, as indicated in the lower part of Figure 1.

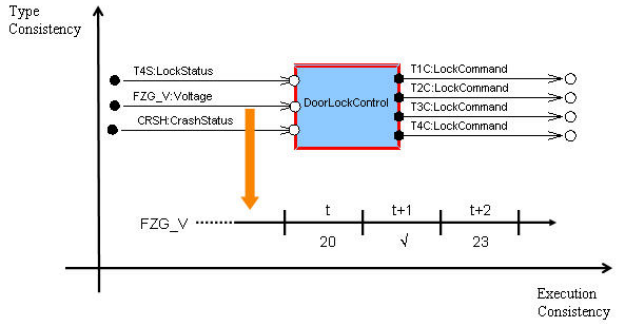

Figure 1: AutoFocus Execution Scheme

Because the discrete time base of AutoFocus abstracts from implementation details such as concrete timing or communication mechanisms, the use of additional, explicit timing information below the chosen granularity of observable discrete clock ticks is avoided. Examples of for such additional detailed assumptions include, for instance, the exact ordering of message arrivals within a time slot, or the precise duration and delays of transfer. Real-time intervals of the implementation are therefore abstracted by the logical-time intervals. In our opinion, this discrete-time and stream-based programming model, with dedicated timing and typing assumptions, suits the needs of automotive embedded software modeling better than modeling and programming approaches based on sequential languages, like C/C++. However, in order to support the dominating implementation platforms in the automotive field, stream-based models have to be transformed during the design process to OSEK-based real-time systems, with threads programmed in a sequential language.

Furthermore, AutoFocus implements the main properties of an architecture description language (ADL). These are components, connectors, typed interfaces, and hierarchies.

1. In AutoFocus, components are called in AutoFocus System Structure Diagrams (SSD). Each SSD has directed ports for the sending and receiving of signals. Signals, i.e. messages, are typed and communication between SSD components deliberately composed with an implicit delay. Additionally, an SSD may contain local variables. An SSD can be refined by either other SSDs or by using an STD.
2. Behavior is expressed in AutoFocus by State Transition Diagrams (STD) and Dataflow Diagrams (DFD).

STDs represent extended finite state machines. As such, they perform a transition if either the appropriate signal arrives and a given pre-condition is true, or, in case no signal is specified, the pre-condition alone is true. During the transition, internal variables of the assigned SSD can be manipulated as well as signals being sent via the associated ports.

Data Flow Diagrams (DFD), see Figure 3, Figure 4, and Figure 14 - Figure 16, define an algorithmic computation of a component. Graphically, DFDs are similar to SSDs. DFDs are built from individual *blocks* with ports connected by channels. Typing of ports is dynamic, using type inference properties of operators. A block may be recursively defined by another DFD. The behaviour of atomic DFD blocks is given either through a STD, or directly through a textual expression (*function*) in AutoFocus' base language. An example of such a function is shown in Figure 5. It is possible to define adequate block libraries for discrete-time computations with this mechanism.

In contrast to the delayed composition primitives in SSDs, the semantics of DFD composition is "instantaneous", in the spirit of synchronous languages[16]. In the AutoFocus tool, instantaneous communication primitives are accompanied by a causality check for detecting instantaneous (or causal) loops. Note that computations "happening at the same time" on the level of logical time are perfectly valid abstractions of sequential, time-consuming computations on the level of real-time implementation if the abstract model's computations are observed with a delay, that is such as the delays which are automatically introduced by SSD composition. The duration of the delay then defines the deadline for the sequential computation.

The message-based time-synchronous communication model does cater to both periodic and sporadic communication as required for a mixed modeling of time-triggered and event-triggered behavior. As shown in Figure 1, each channel in the abstract model either holds a message represented by an explicit value or the "√" ("tick") value indicating the absence of a message in that time instant. Thus modeling of event-triggered behavior is naturally covered by the AutoFocus notation by reacting explicitly depending on the presence (or absence) of a message. The extended AutoFocus notation is described in more detail in [5].

The frequency of signals as well as event patterns are represented in the AutoFocus notation as *clocks*: That is, each message flow in AutoFocus is associated with such a clock. For a given flow, the flow's clock indicates either the frequency of message exchange (periodic case), or a condition describing the event pattern (aperiodic case). Syntactically, a clock is simply a Boolean expression

evaluating to logical “true” whenever a message is present on the clock’s flow. For comfortable modeling, clocks are supported by an inference system, similar to type inference in programming languages. Communication between differently clocked partitions of the model is supported by appropriate sampling operators.

3. Example for AutoFocus modeling

3.1 A Simple Traction Control System

The AutoMoDe Methodology will be demonstrated by means of a Traction Control (TC) System. As a rule, a traction control system compares the wheel speeds of the driven wheels with the actual vehicle velocity. If the wheel speed is above the actual vehicle velocity, then there is slip which is introduced by the engine torque. In this case, two typical actions are taken.

1. If just one of the driven wheels currently experiences slip, the brake caliper might be actuated actively.
2. If both wheels are currently under slip, the engine torque will be reduced. In a gasoline engine, this might be achieved by manipulating the ignition or the throttle valve.

In this example, we consider the second case only and have chosen the throttle manipulation algorithm. Wheel- and vehicle speed signals are also used in Antilock-Braking-Systems (ABS) which also provides caliper interaction. Therefore, TC systems often come along with an ABS system, doing engine management interaction by means of a CAN-Bus.

3.2 The Traction Control Model

Figure 14 shows the traction control algorithm as AutoFOCUS model in its functional environment, namely the ABS system components as well as the throttle controller. The ABS-components shown are reference-velocity (Referenzgeschwindigkeit) determination, wheel-slip and acceleration-control (Bremsschlupf and Beschleunigungsregelung) as well as a coordination of both control strategies. The result of the coordination is a pressure request (Druckanforderung) to the hydraulic valves which manage the fluid supply to the calipers.

From the wheel-speed signals a reference velocity will be calculated against which the actual wheel speeds is compared. The resulting slip is normalized w.r.t. the vehicle velocity and then classified. In this example this means that two different slip values are reached. The slip-determination is shown in Figure 2 while the slip-classification is shown in Figure 3. The rightmost block of Figure 15 influences the current throttle position by a certain amount if the wheel slip of both wheels is outside the limits.

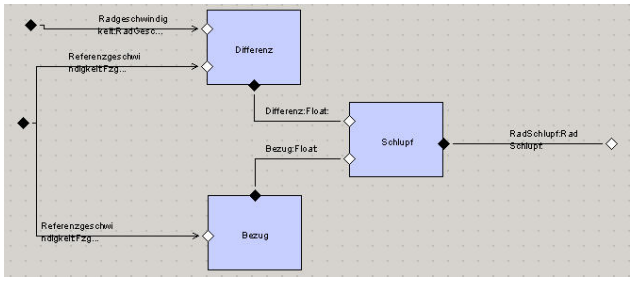

Figure 2: Inner View of the Slip Determination DFD

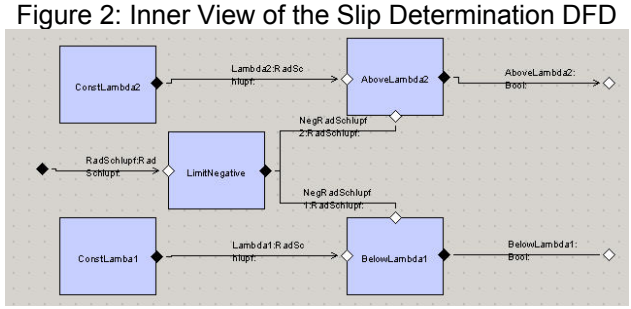

Figure 3: Inner View of the Slip Classification DFD

The amount of throttle actuation is calculated in the throttle control block as shown in top-leftmost block of Figure 16. This amount is compared with the actual throttle position and the driver wish (target-position). The result of this comparison drives the throttle-valve. The throttle-valve is itself a dynamic system which has to be controlled. This is done by a PID-control algorithm which limits the throttle-angle.

The actual control value for the throttle drives a motor via PWM-signals. PWM-signal generation is part of the hardware abstraction and is delivered in this example as a graphical representation of a rapid-development system’s I/O-boards. The same is done for the wheelspeed-sensors and the throttle-valve sensor. According to the vehicle dynamics, the traction control system is sampled with different rates. For example, the wheel- and vehicle speed determination might run in 6 ms, while the traction control might run in 12 ms. Throttle-control is done every ms. In the stream-based model, the rates are described as different clocks while in the embedded software-model, the differently clocked model partitions will be assigned to disjoint preemptive tasks, each task triggered at a rate corresponding to the clock in the model.

4. Refinement to ASCET/INTECRIO

Using the stream-based design approach of AutoFOCUS and applying model based checks to the control algorithm is just one crucial part of embedded

automotive software development. The properties shown on model level should be retained in the real-time execution scheme too. This execution scheme consists of Interrupt-Service Routines (ISR), Tasks which run so-called processes in a sequential manner, so-called messages as inter process communication (IPC) mechanisms, clusters, modules, and sequence calls. Furthermore, there are I/O devices realizing the interface with the pysical world.

To transform the TC-system from the stream-based world to the real-time world an AutoMoDe refinement chain is introduced which incorporate the ETAS tools ASCET [8], RTA-OSEK [10], and INTECRIO [9]. This refinement chain is shown in Figure 4. On the left side, there is the stream-based TC model defined in AutoFOCUS. Its components are transformed to real-time modules, processes, messages and clusters in ASCET, shown in the top-middle. The ASCET code-generator will transform the clusters to C-code which are then integrated on a Rapid-Development target (as shown in the lower right part). Target integration comprises the implementation of an OS-schedule. The OS-schedule will be derived from the AutoFocus clock-scheme but taking into account furthermore the interrupts as generated by the I/O-boards. The schedule is checked with the analysis feature of the RTA-OSEK configuration tool. The target integration step is shown in the center of Figure 4.

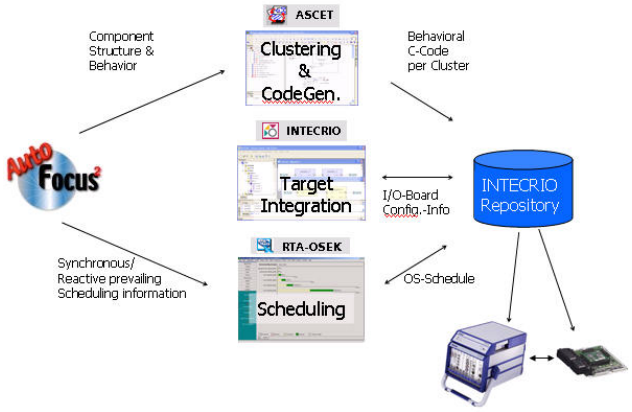

Figure 4: AutoMoDe Refinement Toolchain

In the remainder of this section, the applied transformation steps are described in detail:

- Module-Identification
- Sequence-Call-Generation
- Cluster-Definition
- Software-System Construction
- Target-Integration
- OS-Configuration.

ASCET modules group processes and messages. Processes perform the algorithmic work by executing sequence calls which are to some extend equivalents to assignments in sequential programming languages, i.e. there are operations and variables. To exchange information with other processes (either of the same or an other module) they read and write to messages. After code-generation messages are realized as consistent IPC variables. Processes are linked to operating systems tasks therefore performing the actual work.

4.1 Module-Identification & Sequence-Call-Generation

The currently used AutoMoDe refinement algorithm transforms the hierarchically lowest level of AutoFocus DFDs to ASCET modules. The notation used for showing the bottom-level DFD clusters in AutoFocus is called Cluster Communication Diagram (CCD), and is detailed in [5]. Within each DFD, the elementary AutoFocus blocks which are described as terms in a textual language, will be transformed to ASCET sequence calls. An example for this transformation is the function which tests whether the actual slip is above the limit lambda2 as shown in the upper right block of Figure 3. The function of this block is shown in the lowest line of the block properties of Figure 5. The corresponding sequence call in ASCET is shown in Figure 6. The slip-classification component of Figure 3 is shown as ASCET block diagram in Figure 7. ASCET needs just three sequence calls instead of five because the parameter constructing blocks shown in the left part of Figure 3 are represented by specific blocks in ASCET.

Name	AboveLambda2
Clock	
Function	rad_schlupf(RadSchlupf)<=rad_schlupf(Lambda2)
Comment	{function=rad_schlupf(RadSchlupf)<=rad_schlupf(Lambda2)}

Figure 5: Properties of the Above-Lambda2 Elementary DFD Block

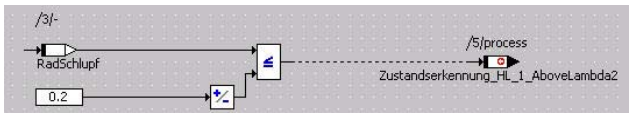

Figure 6: ASCET Sequence Call for Above-Lambda2 Function

As a rule, the number of the module internal variables is determined automatically. Each port of a DFD is translated to an ASCET-message. Per module, there is one process. More sophisticated clustering algorithms based e.g. on the port's clock information are currently under developement.

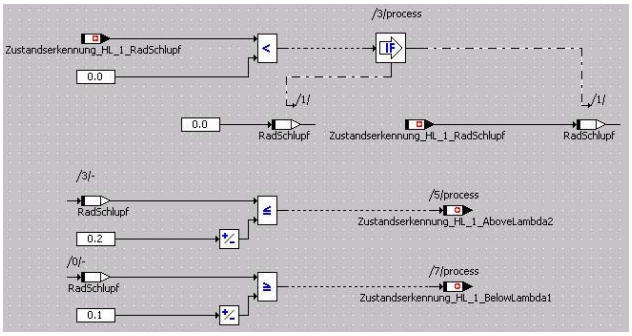

Figure 7: ASCET representation of the Slip Classification Component

These clustering algorithms as well as other model transformations are defined in AutoFOCUS with the help of the Operation Definition Language (ODL) [15]. The ODL is a first-order logic language which allows the definition of tests and transformations. Within an ODL expression user interaction is possible. For example a particular clustering algorithm may ask the user to provide a clock and some components using this clock. Afterwards the algorithm transforms the AutoFocus model by inserting appropriate clusters including the necessary rerouting of the communication. These clusters are the base for the next step.

4.2 Cluster-Definition

The next refinement step is the clustering of ASCET modules to ASCET projects. ASCET project clustering is a step that is performed semi-automatically and requires user-interaction. For example, the reference speed calculation being mandatory for the anti-lock-bracking system as well as for the traction control is composed of five clusters in AutoFocus and hence are represented by five modules in ASCET. As can be seen in Figure 8, each connector is represented by messages which are shown conceptually in ASCET. Since after appropriate grouping to clusters, all the DFD blocks defining a given cluster are on the same clock, and because the vehicle velocity is used by other functions as well, it is sensible to cluster these components together which means they have to run on the same µC in an ECU and its processes are scheduled in the same OS-task.

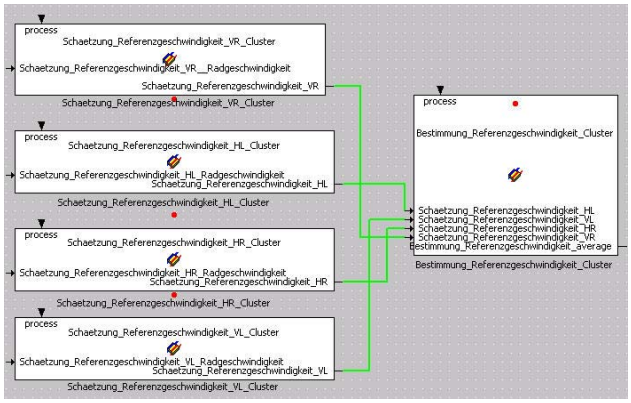

Figure 8: Reference Speed Calculation Cluster in ASCET

4.3 Software System Construction

After all ASCET modules have been clustered to ASCET projects, the ASCET-code generator using the INTECRIO target is applied to all clusters. The result is the C-code per cluster as well as a code description using the SCOOP-IX format. Both description establish an INTECRIO module, not to be mistaken for an ASCET module. For software system construction in INTECRIO, all clusters are imported as INTECRIO modules to INTECRIO and might be further clustered by INTECRIO functions. The result of all clustering steps is shown in Figure 9 . Sensor- and actuator modules are shown in the left- and right side of the software system directly connected to ports. E.g., the wheel speed calculation is done by edge detection. These ports will interface the modules representing the I/O-boards of an ES-1000 rapid prototyping system. Such a system is shown in Figure 10.

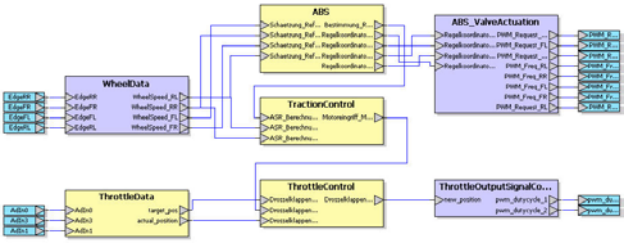

Figure 9: The Traction-Control System in INTECRIO

Figure 10: ES 1000 Rapid Prototyping System with I/O-Boards

4.4 Target Integration

The ETAS rapid prototyping system ES 1000 is VME-bus based. For this traction control example it consists of a microprocessor board (ES 1135), an A/D-converter board (ES 1303), a PWM board (ES 1330), and, for the edge detection of the inductive wheelspeed-sensors, a sophisiticated digital I/O-board (ES 1325). The software interface representation of the A/D-converter board and the digital I/O-board as INTECRIO modules is shown in Figure 11 on the left side whereas the PWM modules for the hydraulic-valve interaction and the throttle-motor are shown on the right.

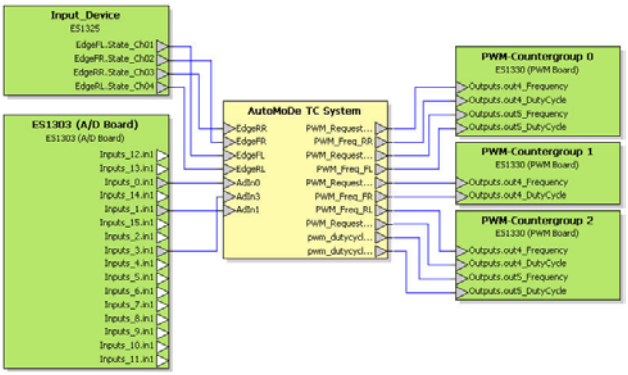

Figure 11: The Traction Control System in INTECRIO refined to run on a RP-System

4.5 OS-Configuration

The AutoMoDe traction control model employs streams running on different clocks. From the AutoMoDe model based point of view, the throttle control algorithm runs 6 times as fast as the anti-lock-bracking algorithms and 12 times as fast the traction control algorithm.

This clock scheme will be translated to a real-time system where the throttle-controller cycle time is set to 1 ms. In combination with the clock-schemes, there will be a 1ms, a 6ms and a 12ms task. The processes of the INTECRIO modules will be allocated to the appropriate tasks. This is shown in Figure 12. Scheduling algorithms typcially need the worst case execution time (WCET) as input for their calculations. Though there are some promising results for the WCET estimation, the AutoMoDe project still uses heuristic approaches based on the process allocation to obtain very first coarse grain WCET estimates of the tasks. In [3], a correct-by-construction method for implementation of time-synchronous AutoFocus programs based on rate-monotonic scheduling is described. The approach uses a double buffering technique for communication from low-frequency to high-frequency tasks. For analyzing this default configuration in the context of the TC example system, we use the planner feature of the tool RTA-OSEK [10] which implements algorithms described in [2]. The result of this analysis using rough estimates of the WCET is shown in Figure 13**.** In the case that the simple top-down, rate-monotonic approach of [3] is not sufficient for a particular situation, the algorithms described in [4] can ensure in a bottom-up fashion that the multiple clock scheme is appropriatly implemented by some given real-time OS schedule. The basic idea of this algorithm is to check whether all signals are read in the appropriate cycle and that a writer is not overtaken by the reader. Nevertheless, it is common automotive design practise to support this analysis by measurements on the real executing system[6] using dedicated measurement and tracing tools such as RTA-TRACE

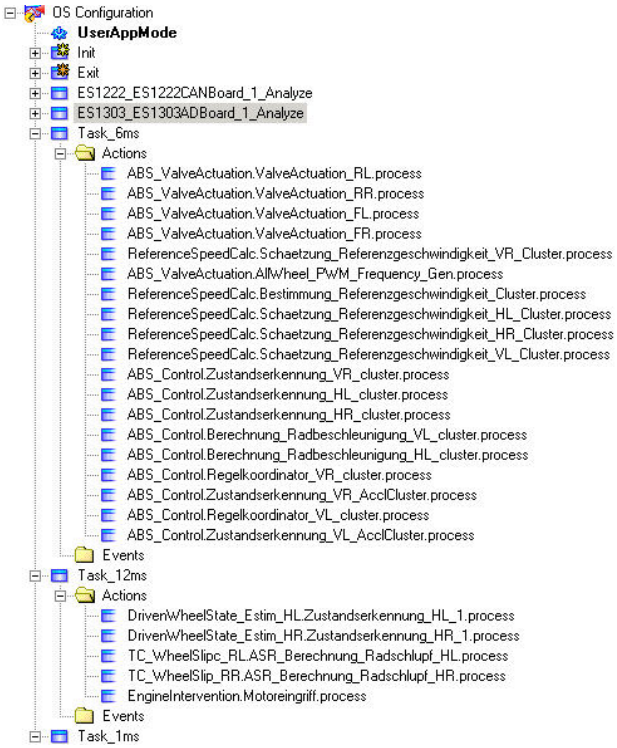

Figure 12: The resulting OS-Schedule in INTECRIO

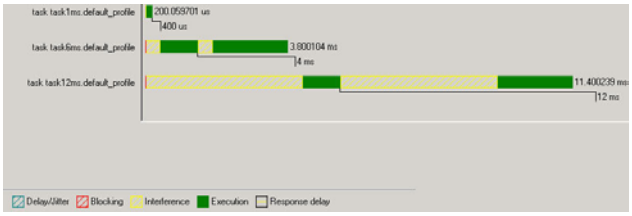

Figure 13: OS-Schedule Analysis

5. Conclusion

This paper shows how embedded automotive control systems can be modeled in a crisp manner by using a stream-based approach. Systematic application of refinement steps incorporate model transformations that result in executable embedded software.

6. Acknowledgement

The AutoMoDe project is partly funded by the German ministry of research and education (BMBF) within the "Forschungsoffensive Software-Engineering 2006" program.

7. References

[1] Freund, U., et al.: The EAST-ADL: *A Joint Effort of the European Automotive Industry to Structure Distributed Automotive Embedded Control Software*. 2nd Workshop on Embedded Real-Time Systems, Toulouse 2004.

[2] Tindell, K., Clark, J.: *Holistic Schedulability Analysis for Distributed Hard Real-Time Systems, Microprocessing and Microprogramming* - Euromicro Journal, vol. 40, pp. 117-134, 1994

[3] Bauer, A., Romberg, A.: *Model-Based Deployment in Automotive Embedded Software: From a High-Level View to Low-Level Implementations.* MOMPES 04 Int'l workshop, Hamilton, Canada, June 2004

[4] Manucci, L., et al.: *Correct by Construction Transformations across Design Environments for Model-based Embedded Software Development*, DATE' 05, Munich 2005

[5] Bauer, A., et al.: *AutoMoDe – Notations, Methods, and Tools for Model-Based Development of Automotive Software*, SAE-Paper 05AE-268, Detroit, 2005

[6] Schäuffele, J., Zurawka, T.: *Automotive Software Engineering.* Vieweg Verlag, Wiesbaden, 2003.

[7] Braun, P., Slotosch, O.: *Development of a Car Seat: A Case Study using AutoFocus, Doors, and the Validas Validato*r, in OMER – Object-oriented Modeling of Embedded Real-Time Systems, Lecture Notes in Informatics (LNI), Volume P5, GI 2002.

[8] ETAS GmbH: *ASCET User Guide V 5.1*, Stuttgart, 2005 (www.etas.de)

[9] ETAS GmbH: *INTECRIO User Guide V 1.1*, Stuttgart, 2005 (www.etas.de)

[10] LiveDevices: *RTA-OSEK User Guide V 4.0*, York, 2005 (www.livedevices.com)

[11] Thurner, T., et al.: *The EAST-EEA project – a middleware based software architecture for networked electronic control units in vehicles.* In: *Electronic Systems for Vehicles* (VDI Berichte 1789), p 545 ff. VDI-Verlag, Düsseldorf, 2003.

[12] Eisenmann, J., et al.: *Entwurf und Implementierung von Fahrzeugsteuerungsfunktionen auf Basis der TITUS Client Server Architektur*; VDI Berichte (1374); pp. 399 – 425; 1997; (in German).

[13] Migge, J., Elloy, J.P.: *Embedded Electronic Architecture*, 3 rd. OSEK/VDX Workshop, Bad Homburg 2000.

[14] www.autosar.org

[15] Schätz, B., et al. : *Checking and Transforming Models with AutoFocus*. In 12th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS), IEEE Computer Society, 2005

[16] Benveniste, A., et al.: *The Synchronous Languages Twelve Years Later. Proceedings of the IEEE*, 91(1):64–83, 2003.

[17] Broy, M., Stolen, K.: *Focus : Specification and Development of Interactive Systems*. Springer, Berlin 2001

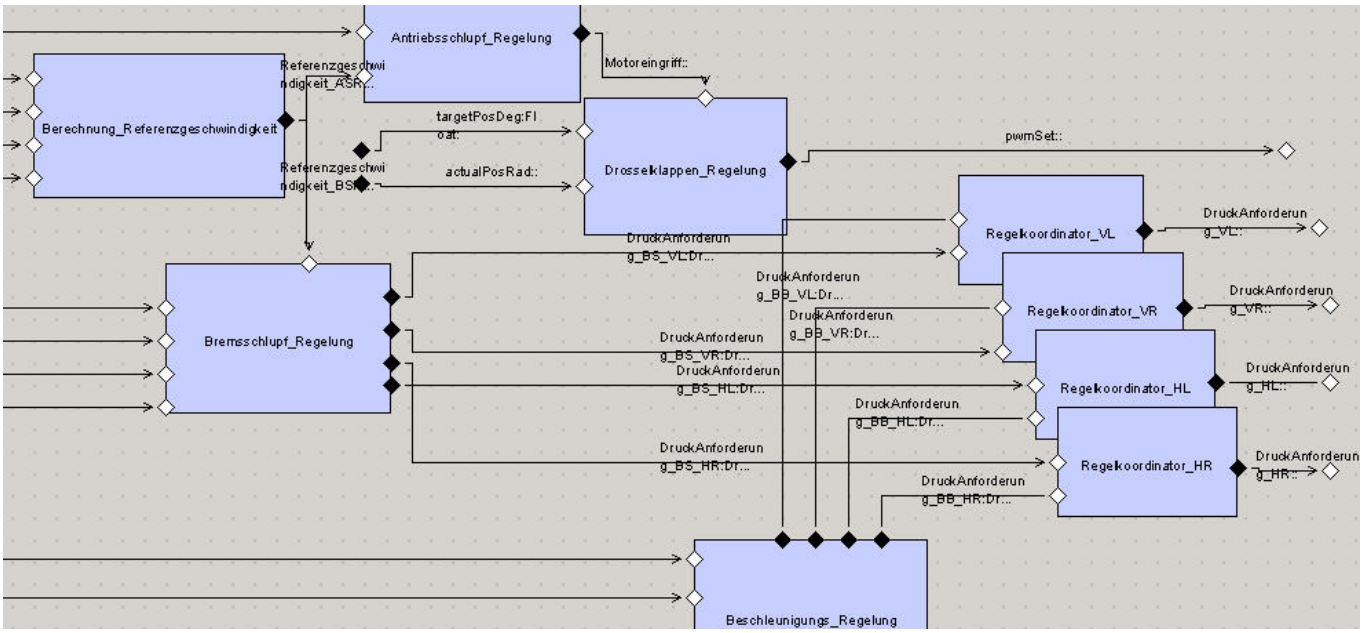

Figure 14: AutoFocus DFD showing the Traction Control System with its Functional Environment

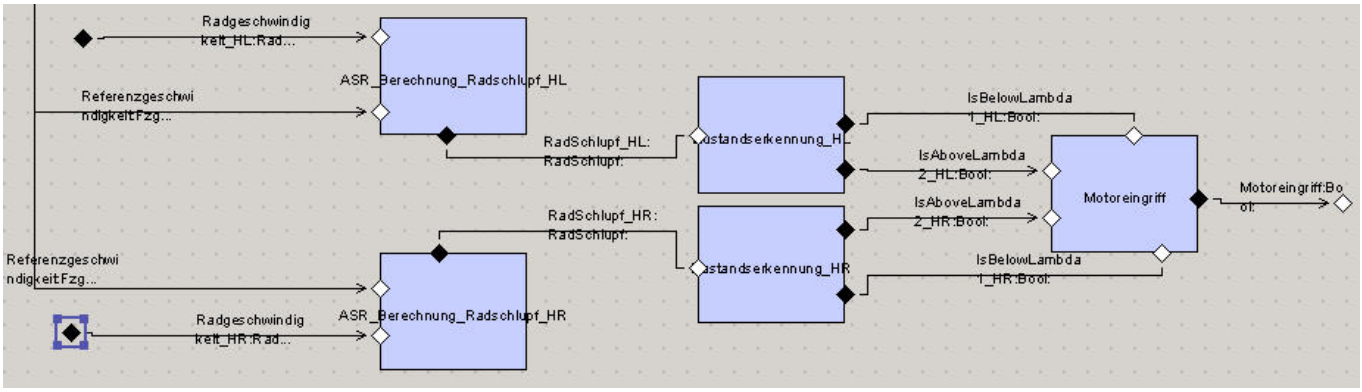

Figure 15: Inner View of the Traction Control System DFD

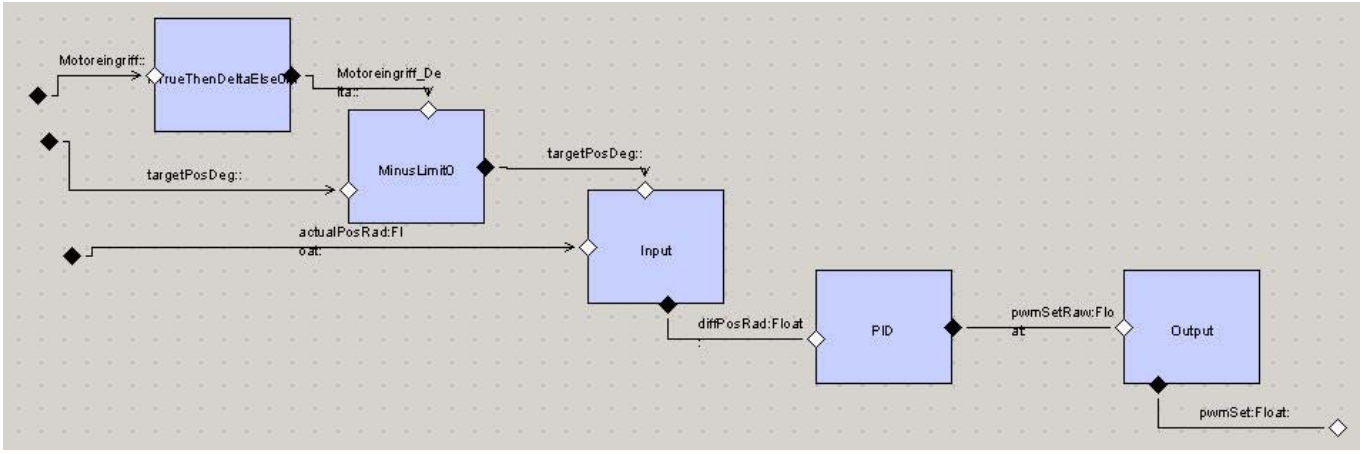

Figure 16: Inner View of the Throttle Controller DFD