Model-based Development of Autosar Application Software

1 Introduction

With the aid of model-based function specifications and by using standardized interfaces for the ECU software the joint development of software for automotive electronics gets simplified. In addition, this approach reduces costs and time-to-market, enhances software quality, and provides mechanisms required for the design of safety relevant systems. To reach these goals, Autosar defines an architecture for automotive embedded software. It provides for the easy reuse, exchange, scaling, and integration of ECU-independent software components.

2 Model-based Function Development

With the Ascet development tool by Etas functions of the application software can be described in the form of a hardware-independent model. Compared with the purely textual specification making up the traditional engineering specification, a decisive advantage of this approach is the easy migration of proven function models from one vehicle project to the next. This capability positively affects product quality, significantly reduces development and maintenance costs, while supporting platform strategies.
3 Abstraction of Operating Functions

As cet is a software tool for the graphical programming of ECU application software. Using finite state machines and block diagrams, the development tool facilitates the graphical modeling of function properties independently of the vehicle’s E/E architecture. Unlike with typical simulation tools, the modeling in Ascet occurs at an abstraction level that is optimized with regard to the requirements of ECU programming:

- Finite state machines and block diagrams are translated into C code for the ECU by means of simple and logical transformation steps
- Explicit mapping of subroutines (processes) to operating system tasks makes it possible to define the order of execution of the control algorithms contained in the models
- Upon model execution, data is safely transferred through defined interfaces (messages) between subroutines.

As cet’s own code generators are capable of the direct translation of models into optimized series C code for specific ECU microcontrollers [1]. The code thus generated is highly readable and MISRA compatible [2], as well as well-structured. It is suitable for easy integration in the basic software of a series production ECU.

When it comes to mapping a model in C code, As cet provides developers with a high degree of control. The model is capable of allowing for microcontrollers-specific functions or memory areas. So-called modules and processes are used to subdivide both structures and sequencing of modeled functions. A function is represented by one or more modules. Messages are used to describe module interfaces, whereas processes define the ways and means by which messages are accessed by control algorithms. Because the processes are executed by operating system tasks, the As cet message concept ensures the independence of the process execution of the E/E architecture present – such as the communications matrix in use, or the integration of sensors and actuators in the ECU software.

The tool also opens the door to C code integration. For example, developers can import a network management algorithm from the basic software for combination with a function of the application. The resulting composite model is immediately available to the developer for testing on the PC, allowing specification faults to be identified early on in the design phase.

As cet manages the individual modules containing the function models in a database. In this way, one or more functions that shall be integrated on an ECU can be bundled for a given project. A special interface aids in versioning the function models in the As cet database by means of an external configuration management system.

4 The Significance of As cet

The approach of detaching the functions of the application from the basic functions of sensors and actuators and from those of the E/E architecture, can be fully exploited by means of As cet [3, 4]. As cet encapsulates control algorithms in the form of atomic software components that communicate through ‘ports’. These are differentiated by their interfaces. The internal behavior of a given software component is determined by the interaction of several executable, or ‘runnable’, entities. A runnable entity is activated in an operating system task on the ECU to perform control computations, while at the same time defining the access to the data elements of a port.

Electronic functions can be constructed from atomic software components. This is accomplished by interconnecting the ports of the atomic software components, and by adding proprietary ports to the software component – or composition – assembled in this way. The compositions’ ports can be used, for example, for receiving signals such as vehicle speed, which are available vehicle-wide. The mechanisms defined by As cet for the connection of ports ensure independence of the E/E architecture, especially for signal names.

The iterative assembly of compositions culminates in the so-called virtual function bus (VFB), which interconnects all of the software components of a vehicle or vehicle domain, such as the powertrain, for example. The VFB segments – the atomic software components – are distributed to the individual ECUs of a given vehicle type, Figure 1.

On an Autosar-compliant ECU, the runtime environment (RTE) handles the communications for the atomic software components. The RTE serves as the interface between software components, basic software (BSW), and operating system (OS) as shown in Figure 1. The operating system uses the RTE to call the software components’ runnable entities. The RTE replaces proprietary interfaces between function components, sensor/actuator components, as well as the communications matrix of the respective vehicle type.

As cet already realizes essential Autosar concepts: Atomic software components are easily represented by modules, runnable entities by processes, and instances of port access by messages. In addition, As cet supports Autosar constructs such as interfaces, RTE events, services, and so-called exclusive areas.

In addition to an Autosar Authoring Tool, which provides initial descriptions of the system architecture and Autosar interfaces, As cet is therefore ideally suited to the definition and implementation of the behavior of Autosar-compliant vehicle functions, Figure 2.

In order to implement the communications between the various Autosar software components, As cet employs the RTE-API as part of its code generation. This approach ensures the consistent development of software components, from their structural design to behavior modeling. Thanks to the similarity of Autosar and As cet concepts, adapting the interfaces of existing As cet models to Autosar is not an issue. As has been demonstrated in the evaluation described below and in other examples [5, 6], the required modifications are far from exorbitant.

5 Virtual Integration and Validation of Autosar Software Components

The Autosar virtual function bus paves the way for the virtual integration of software components. With Autosar, software components can be integrated with greater ease, instead having to distribute them to ECUs beforehand because the VFB is dissolving the ECU boundaries. The virtual integration is accomplished with the use of the Etas-made tool Intecrio. This prototyping environment is capable of integrating behavioral models designed in Matlab/ Simulink or As cet, as well as modules that were manually developed in C. The new version 3.0, scheduled for rollout in late 2007, enables the integration of Autosar software components (SWCs) with legacy function modules. Figure 3. Intecrio thus provides for the reuse of existing models and C code during the migration of ECU software to Autosar architectures.

Intecrio already comprises the RTA- OSEK real-time operating system as an integral part. It ensures target-close behavior of control function prototypes. RTA- OSEK is available stand-alone for Microsoft Win-
dows providing support for virtual validation on the PC (virtual prototyping). In addition, Intecrio can also be used for testing the behavior of integrated components on a rapid prototyping system in the real-world vehicle environment by means of Autosar RTE and OS.

6 Autosar-compliant Reengineering of an Engine Management System

Starting from an existing Ascet project the migration from a purely Ascet-based development to Autosar was explored. The base point of this work [7] was an existing Ascet project of an engine management system (EMS) which demonstrated the applicability and benefits of model-based design and automatic code generation with the Etas tools [8].

The EMS application software, which was developed in various stages, was deployed on a series production ECU. The ECU was operated by an OSEK operating system (Ercosek) and installed in a BMW 5-series in order to control the engine, a 4.3 liter 8 cylinder 4-stroke gasoline engine with intake manifold fuel injection. The complexity of the real-world example emphasizes the relevance of the work.

The reengineering of the software comprised multiple steps:
- Partitioning of the existing functionality into Autosar SWCs
- Autosar-specific adaptations
- Generation of Autosar-compliant code
- Integration of the Autosar software components
- Configuration of the basic software (OS and COM stack)
- Generation of the Autosar runtime environment (RTE)
- Validation of the functions using virtual and rapid prototyping
- Generation of suitable basic software (especially operating system and COM stack)
- Compilation and linking of all generated source code files (SWCs, RTE, OS, COM stack)
- Deployment on a TriCore evaluation board.

For the reengineering project, Ascet was used as an integrated development environment for Autosar software components, offering the full range of modeling entities and flexibility of Ascet for this purpose. Intecrio was used for functional validation using virtual and rapid prototyping and as a mean to integrate the SWCs developed in Ascet into an ECU software system and configure the basic software (OS and COM stack). An appropriate Autosar XML description was generated directly from Intecrio which serves as input for RTE generation, OS generation, and COM stack generation. The complete software was deployed on a TriCore evaluation board. To validate the engine management, the board was coupled via CAN to a test system executing an engine simulation. Other basic software modules such as NVRAM management or diagnostics were out of the scope of this project.

7 Conclusion

In the course of this work, the mapping of concepts from the Ascet behavioral modeling tool and the Intecrio prototyping environment to Autosar has been demonstrated. Different packaging of functionality to Autosar software components can be provided to service specific requirements, such as the protection of intellectual property. The reuse of functions for different components helps to reduce development costs and to increase software quality.

References