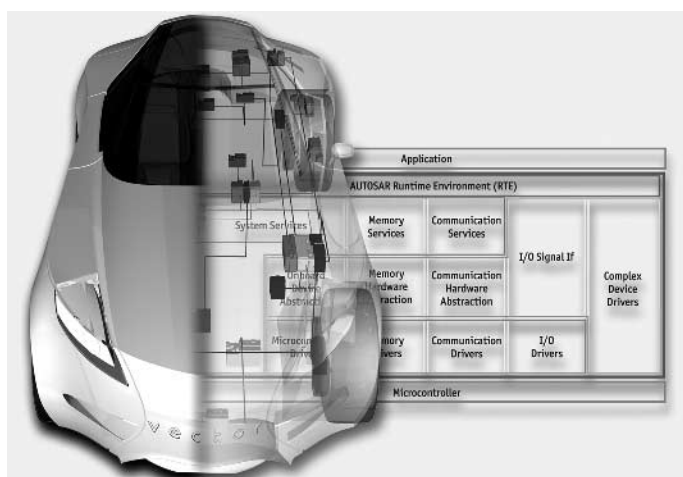


Einbindung bestehender Steuergerätesoftware in die Autosar-Architektur

Integration of Existing ECU Software in the Autosar Architecture



“Cooperate on standards – compete on implementation”, that is how the guiding principle of the Autosar standardization initiative reads. A long harboured wish of the automotive industry is now on an ideal path toward becoming a reality. The industry is prepared to implement the necessary paradigm change to accomplish this.

1 Introduction

In the past ECUs were proprietary solutions – highly customized components that were developed to be OEM and vehicle specific. Yet this control module centred approach to ECU development has disadvantages. A single change in requirements may make it necessary to conduct a completely new ECU development. As complexity in ECU functionality increases, this is associated with enormous expense. Further intensifying the need for a conceptually new approach are new constraints such as the need to represent an even greater number of variants and equipment options, offer options for upgrading architectures and handle non-functional requirements such as ECU diagnostics or availability.

Therefore, a number of years ago many OEMs and suppliers began to address the issue of how to replace this ECU centred development approach by a functionally

oriented approach. The Autosar standardization initiative has consistently advanced this approach. The goal of the initiative was to separate the customer-specific application from basic functions that can be integrated by standardized interfaces. This involved mapping the software to a layer model and defining uniform interfaces. In the future, together with a configuration description that is also standardized, it will be possible to develop, test and optimize functions independent of the underlying hardware environment.

The transition from customer-specific software to standardized base software and application software with uniform interfaces promises enormous cost and quality advantages to all parties involved in the development process. The driving factors are simpler integration and the ability to transport functions across vehicle and OEM borders, as well as greater flexibility in maintenance, scalability of functional features, and hard-

Authors:

Matthias Wernicke and Jochen Rein

ware independence of the software. A welcome side benefit here is the growing reliability of the electronic system.

Validity is also improved by the Autosar-specified development process. This process begins with a model-based functional description, continues with a consequent use of tools and also includes automatically generated and reproducible tests.

Release 1.0 was published in May 2005, and it already contained specifications for 31 base software modules (BSW). The functional capabilities of the modules were checked and validated in a subsequent implementation and validation phase (Validator 1). By May 2006 42 BSW specifications had already been published, among them the Runtime Environment (RTE). All specifications have successfully run through a test phase (Validator 2). At the end of 2006, the first full release (R 2.1) was introduced, which also contains the methodology specification.

As one of its first premium partners, Vector has actively participated in standardization activities of the Autosar Initiative and offers base software and tools for Autosar.

2 Autosar Architecture

To achieve Autosar goals related to partitioning of the application software from base modules and functions, the vehicle electronics is abstracted and subdivided into several layers, **Figure 1**. The connection to the actual microcontroller and thereby the physical basis is represented by the Microcontroller Abstraction Layer, which maps the microcontroller's functions and periphery. This layer defines interfaces for memory, I/O drivers and its communication connections. Features that the microcontroller cannot offer can also be emulated in software. Above this layer is the second layer, the ECU Abstraction layer. It defines the ECU's internal hardware layout and provides drivers for the ECU's external periphery for example. In another layer, the Services Layer, various basic services are provided such as network services, memory management, bus communication services, and the operating system. This layer is already largely independent of the hardware. The RTE represents the fourth layer. This is where the actual separation of application and BSW occurs. The RTE handles integration of the application software and controls the exchange of data between application and BSW. It is only because of this layer that it is possible to reuse application

software, since the defined interfaces allow application software to be developed and transferred to any other Autosar-conformant ECUs without special knowledge of the later hardware to be used.

The so-called Virtual Functional Bus (VFB) serves as the basis for configuration of the layers. All vehicle electronic components use this layer to communicate in an abstracted view. For this purpose the software components use ports that are specified by port interfaces. Connectors interconnect the ports. It does not matter to the VFB whether this is a connection within an ECU or a connection via an external bus. That is not decided until the final system layout is set and the components are assigned to a specific ECU.

The software component itself does not require any knowledge of this later partitioning and can therefore be developed independently. It is subdivided into execution units, so-called Runnable Entities, which are executed like procedures when a certain event occurs. Such an event might be the arrival of a new sensor value or a periodic activation. The description of the electronic system formulated from the perspective of the VFB is used to define the interfaces of the specific software components. This makes it possible to develop application components independent of the specific ECU.

The "counterpart" in the ECU is the RTE. It ensures access to I/Os, memory and other base services. The RTE can be generated ECU-specifically from the model-based description and can thereby be adapted to different requirements in a resource-efficient manner.

3 Methodology

Now that Phase 1 has been completed and Phase 2 has begun, Autosar also starts as planned to formulate the methodology of the development process. Conformance to a structured creation process is recognized as an important precondition for creating error-free software. Deficiencies in the requirements list are detected early, reuse and porting of software components are simplified, and the system is more reliable overall. Nonetheless, the methodology offers certain freedoms: For example, users are free to select a top-down approach or a bottom-up development.

The Autosar Initiative provides for seamless tool support of the software development process. Mature tools facilitate structured implementation of requirements and their con-

sistent development, systematic creation of configurations and automatic consideration of complex interdependencies.

First a formal description is made of the three main parts: Software (SW Component), ECUs (ECU Resource) and System Constraints. Suitable editors are used to create a configuration description for the total system, see **Figure 2**. This system configuration serves as a basis for creating the ECU configurations that the user creates with the help of configuration tools for the individual base software modules. At the end of the process, a number of generators are used to generate the ECU-specific implementation of the RTE and base software.

All design and configuration data created in the development process are described in a uniform format. Autosar has defined a format based on XML for this purpose. First, it guarantees consequently the data consistency during the development process, and secondly it simplifies seamless integration of required or supplemental tools.

As early as 2003 Vector Informatik introduced a line of tools based on a structured and modular development approach that supports users in designing distributed functions, allocating software components, and integrating the code of the networked ECUs. Today the DaVinci Tool Suite turns the Autosar idea into reality and conforms fully to the specification. As one of the first and only tool providers in this area, Vector now supports the specified universal and consistent Autosar development process.

The DaVinci Tool Suite is used to handle all design steps systematically and manage the complex interrelationships. Tool support begins with the design of the software architecture and covers tasks ranging from network configuration to code generation. Well-functioning data management and configuration management are essential in the development process for a complex system [1]. The eA-SEE tool environment performs this task in background as well as data integration and user administration.

After requirements have been established, the development process begins with formal description of the system topology and software components. The DaVinci System Architect is used for this purpose. It allocates software components to the individual ECUs and thereby defines which communication is to be performed locally and which will occur via a bus system. Afterwards the interface data of the software components are mapped to the bus signals.

DaVinci Network Designer defines the layout and time behaviour (Scheduling) of the network messages. The special properties of vehicle-specific CAN, LIN and Flexray bus systems are considered here. As a result, the developer obtains the System Configuration Description, which contains all information on the system architecture including the communication matrices. As an Autosar-conformant XML file the System Configuration Description is used for ECU development.

DaVinci Developer takes this XML file as a basis and uses it to configure the RTE. GENy, osCAN Configurator and MICROSAR. EAD are now coming into play as configuration tools for the base software. They sup-

port the generation of a comprehensive set of BSW modules such as the operating system, communication stacks and I/O drivers, **Figure 3**.

Tools support both implementation of software components and initial testing. When suitable model-based tools are used, functionality can be formulated using state diagrams for example. This enables later simulation of one or more software components and testing already conducted on the functional model. As an alternative, direct implementation of the software components in C is also possible. CANoe serves as a PC-based test platform for software components and is used to simulate all software components as DLLs and emulate the VFB.

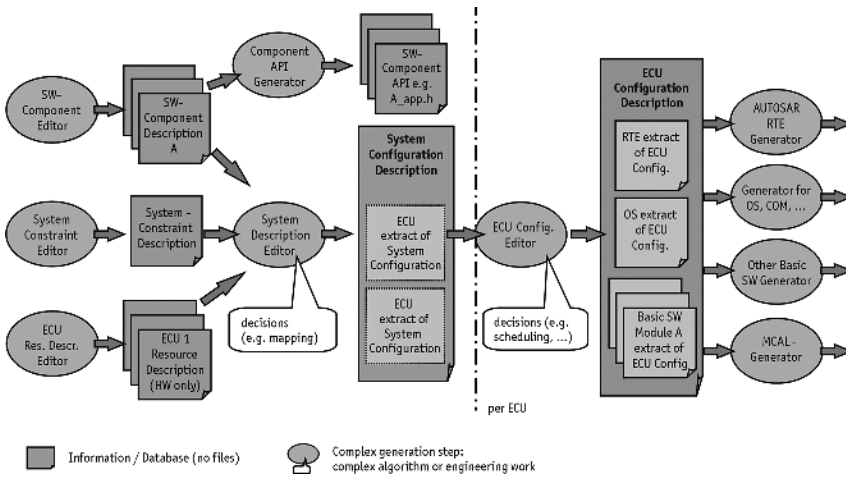


Figure 2: Structural description of the software components. Creation of Autosar-conformant software components is subdivided into clearly specified development steps

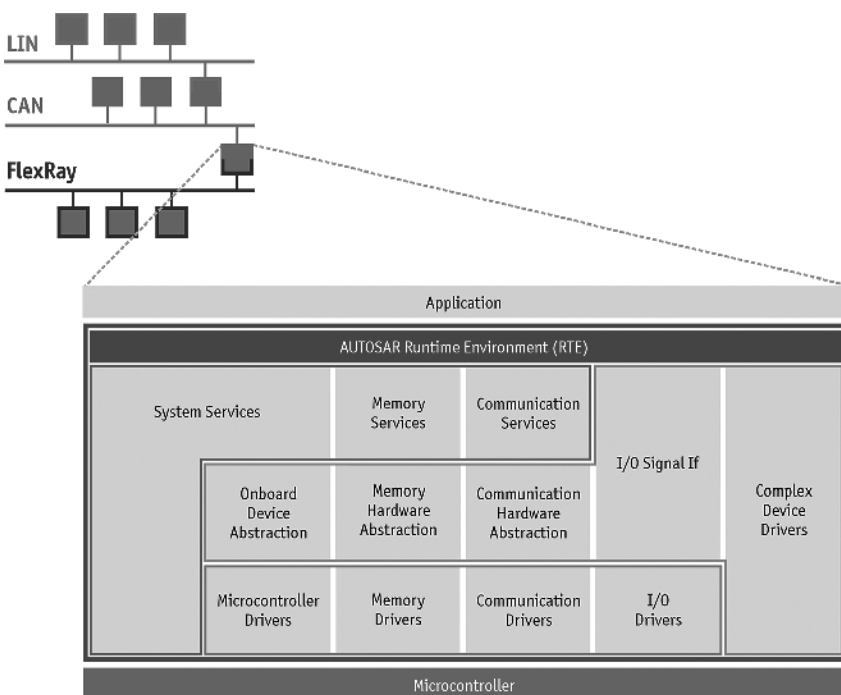


Figure 3: The Vector Autosar solution: From system description to the standardized ECU software

4 Migration

Theoretically, automotive OEMs could attempt to integrate a fully Autosar-conformant electronic system in their next generation of cars. The cost and effort required to redevelop all components would be enormous, however, and the process would not be without risk. Since the standard is still in a validation phase, this would also be associated with planning uncertainties. In developing the Autosar specifications, working partners gave priority to incorporating proven technologies in software development. Furthermore, there are areas where optimizations are necessary in software. Modifications of conventional technology are therefore unavoidable when newly developed Autosar components are added. Stepwise introduction of Autosar-conformant software components into the overall architecture addresses this issue and is preferred by automotive OEMs since it offers better handling and greater validity.

Figure 4 shows the development of component integration in car generations at Daimler-Chrysler [2]. Each new model series and software generation comes along with new functions that need to be integrated. Only the modules shown in red are taken from the predecessor system; their technical aspects are quasi-frozen. It is apparent that in spite of identical functionality some modules are continually redeveloped. Therefore at Daimler-Chrysler a strategy has already been developed to establish the extent to which vehicle electronics and thereby its components should conform to the new standard. Autosar-conformant modules will already be added to the proven vehicle electronics for the first time in the next generation of software. Gradually, existing modules will also be migrated to Autosar, until finally a complete Autosar-conformant system of ECUs is achieved.

Today OEMs and suppliers can already begin a migration of existing software functions to Autosar. Release 2.0 paves the way for such a conversion. Different approaches are conceivable here. Modularized base software already provides the foundation for Vector's CANbedded software components today. Vector customers can therefore replace individual modules by Autosar components with little effort. This applies to the Transport Protocol, bus interface and diagnostics for example.

OSEK Network Management (NM), which is widely used in the automotive industry, can be replaced by Autosar-NM. Nonetheless, it should be noted that synchronization is not automatically done

when mixing ECUs with Autosar-NM and OSEK-NM within a network. An ECU containing both network management variants could handle this task.

In a transition phase, newly written application software will support both prior and new standard components. A certain overhead must be accepted here. The great advantage though is universal reusability in different projects, because different OEMs will not necessarily strive for similar migration strategies. Mixed forms are conceivable: While one OEM might want to just guarantee compatibility to Autosar on the bus but otherwise wants as little change as possible, another OEM might preserve components with a lot of bus interaction in the first stage to maintain compatibility with existing ECUs.

For suppliers it is advisable to switch over to the RTE early on. This makes the application independent of underlying layers, and it may be developed decoupled from hardware requirements, **Figure 5**. However, it cannot be denied that this approach includes additional effort for the RTE. The RTE must support the different variants of underlying base software, as long as these variants do not provide an Autosar-conformant API to the RTE. Moreover, standard formats used today for configuration data such as DBC, LDF or FIBEX must be converted to uniform Autosar XML formats.

The migration is advisable, however, because suppliers can decide to go for a standardized API even before the OEM requires it. At an early stage this spares the supplier further migration steps.

5 Next Steps: IDE as Basis

Networking of components in automotive electronics must go hand in hand with close tool coordination. Only seamlessly complementary tools can make the development process consistent and the complexity manageable. Additional benefits arise by integrating the tool chain into Integrated Development Environments (IDE) that are accepted industry-wide, such as Eclipse or Visual Studio. Extensions to this IDE enable coding, debugging and testing of Autosar components within a homogeneous environment, which may also serve simultaneously as an execution and simulation platform for Autosar components. This makes it possible to test real executable software functions while auxiliary functions are simply added as models via a simulation interface, **Figure 6**.

Simple test scenarios can be created quick-

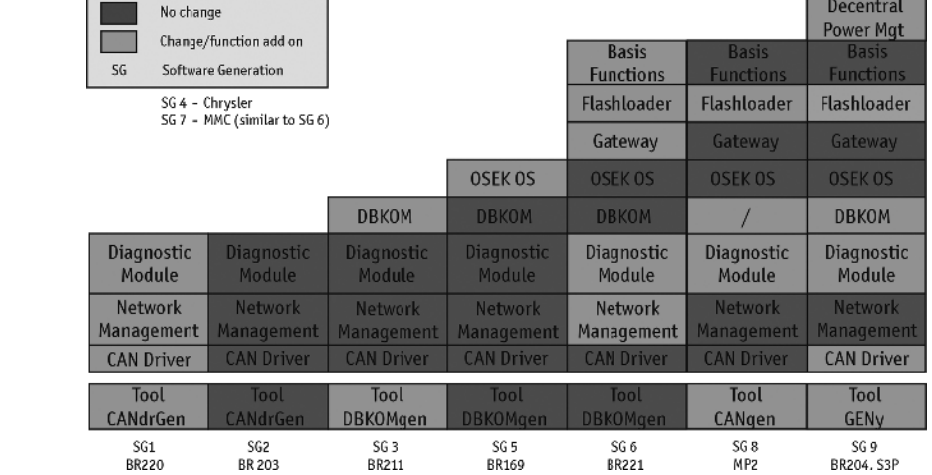


Figure 4: Autosar layer model of ECU-specific software

ly. Different execution speeds can be used for different types of testing. Fast execution of test cases delivers high repetition rates and enables a large number of automated tests, whereas slow execution facilitates user-friendly debugging of the software.

Possible supplements to such an IDE include test case generators and static code or coverage analyses. The better the linkage of such add-ons to the development platform, the simpler and more economical are the tests, and the product will operate more reliably later.

6 Summary

The complexity of vehicle electronics continues to grow. Autosar's definition of interfaces and description of the system by abstraction layers does not reduce the volume of requirements and interdependencies of functions. To master them requires powerful and capable authoring tools.

Decoupling of the base software from the application is a prerequisite for running functional code on different hardware platforms. This leads to a qualitative improvement of the system since proven applications in product quality may be reused. Moreover, the development capacities may be used to focus on innovations and new challenges. The development process prescribed by the specification supports a structured approach and points out potential error sources at an early stage.

Vector Informatik has consistently implemented the Autosar concept and supports the development process in all phases. This begins with structured design of the Autosar software components and

their distribution to ECUs, continues with definition of communication, and finishes with configuration of the base software. OEMs and suppliers can already benefit from the advantages of Autosar projects today. Especially for projects, mixing conventional components with new Autosar technology Vector is the right implementation partner. By calling upon its expertise over the entire range of Autosar software, integration experience and suitable tools for integrating third-party components, Vector helps in all steps on the path to the Autosar ECU.