# Hardware-independent Software Development with AUTOSAR

Stefan Bunzel, Khosrau Heidary (Continental); Simon Fürst, Andre Lajtkep (BMW Group); Jürgen Mössinger, Jürgen Cordes (Bosch); Stefan Schmerler, Christian Kühn, (Daimler); Frank Kirschke-Biller, Bernd Frielingsdorf (Ford); Robert Rimkus, Rick Kacel (GM); Alain Gilberg, Bertrand Delord (PSA Peugeot Citroën); Kenji Nishikawa, Hiroyuki Hirano (Toyota); Andreas Titze, Bernd Kunkel (Volkswagen)

AUTOSAR cooperation
Bernhard-Wicki-Str. 3
80636 Munich, Germany
admin@autosar.org

**Abstract:** This paper describes the development of application software with AUTOSAR. The AUTOSAR architecture and the development methodology enable a hardware independent development of application software, so that higher reuse and increased flexibility and scalability are possible.

## 1 Introduction

AUTOSAR (AUTomotive Open System ARchitecture) has become a de-facto standard for embedded software in the automotive industry. Since 2003 a cooperation of car manufacturers, suppliers, and other companies from the electronics, semiconductor, and software industry have been working on the development and introduction of this standard [AUTO]. By now many AUTOSAR members apply the standard in their product development and there are even first series products on the road [KG08], [Fü09].

AUTOSAR is a key enabling technology to manage the growing electrics/electronics complexity. It aims to increase the reuse of software components, in particular between different vehicle platforms, and between OEMs and suppliers. It enables the scalability of embedded automotive software to different vehicle and platform variants, the transferability of functions throughout the vehicle network, and the integration of functional modules from multiple suppliers. Therefore the AUTOSAR standard defines an architecture that separates application software from infrastructure related basic software, as depicted in Figure 1. The functional contents of the application software are different and related to the brand identity and the desired characteristics of the car manufacturer, or its system suppliers, whereas the functionality of the basic software is not visible to the customer and thus could been standardized by AUTOSAR in detail.

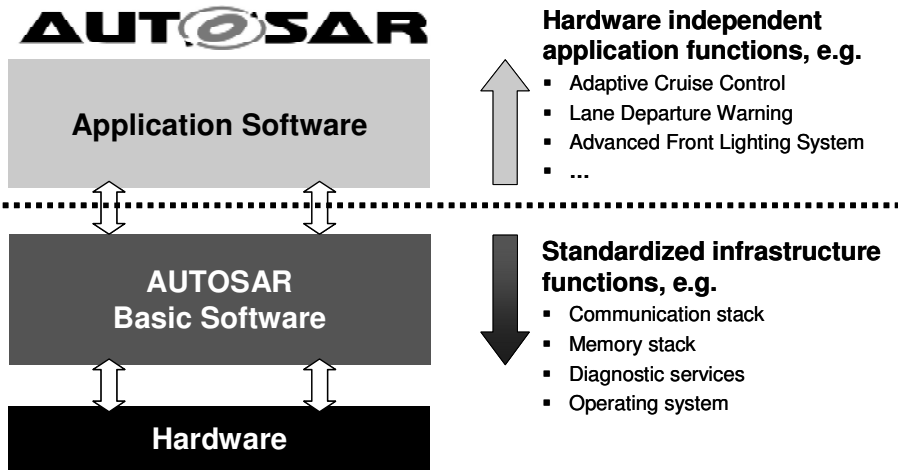


Figure 1: Application software separated from infrastructure functions

Beneath the architecture AUTOSAR has worked out a methodology how to develop software according to this architecture. A third topic of standardized interfaces for typical automotive applications completes the technical approach of the standard.

This paper emphasizes the benefits for application software development with AUTOSAR. Thus it highlights the methodology, explains the hardware independency of application development with the concept of the virtual functional bus (VFB), the assignment of application software to ECUs and the integration, and last but not least it describes the usage of standardized application interfaces.

## 2 AUTOSAR Methodology

The AUTOSAR methodology describes the major development steps of an overall AUTOSAR system, which means the entire software for a network of interconnected ECUs in a vehicle. In the release 4.0 the methodology is specified as a model yielding a set of HTML-documentation [AM40]. It addresses the wide range of software development from the system-level configuration to the generation of an ECU executable, and it supports a widely decoupled development and implementation of application functionality, as well as a seamless integration and configuration of both, the overall system and its individual ECUs. So the methodology is a guiding framework of how to use the AUTOSAR architecture [BF08].

Figure 2 depicts an abstract top-level view on the methodology. Firstly on the level of functional architecture there is the development of a so called virtual functional bus (VFB) system description. This description is independent of any network topology or deployment of features across multiple ECUs. It contains a component model of all application functions. This functional architecture means a partitioning of functions into components, which can even be a hierarchical model where components are clustered into compositions. The development on this level also yields a data model for the interaction of the components. In addition it optionally deals with timing constraints on VFB level. Subsection 2.1 describes further details on the VFB.

The next level considers the physical architecture of the entire system, i.e. the activity 'design system' leads to a system description that defines the system topology of ECUs, the network, and the mapping of components to ECUs. Before the software for each ECU can be built, the information regarding to this ECU have to be extracted from the system description. This allows building and integrating the software for each ECU separately from the other ECUs. Of course, building the ECU software requires appropriate basic software for the ECU and all application software components mapped to the ECU. The delivery of basic software is out of scope in this paper. By now there is a broad variety of basic software implementations from different vendors for many hardware platforms on the market, so that the basic software and corresponding configuration tools can be regarded as off-the-shelf products. The development of the application software components with the definition of the internal behavior, coding, and implementation are independent from hardware and can be done separately for each component. Subsection 2.2 explains the component development in more detail.

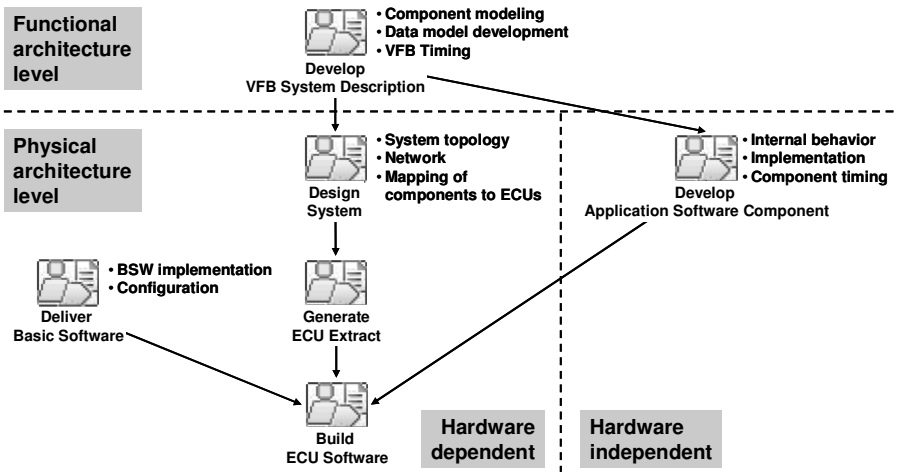


Figure 2: Methodology overview

## 2.1 Virtual Functional Bus

The Virtual Functional Bus (VFB) is a technical concept that enables the development of the functional architecture of the entire system independent from the actual hardware topology of ECUs and network. The functional architecture is a partitioning or clustering of the application functions into components, i.e. the so called AUTOSAR software components (SWC). In order to formally handle and model SWCs and their interaction each SWC needs a formal description: the SWC description. For the development on VFB level, the SWC description does not have to be complete. At least the data model, i.e. the used interfaces, data types, and services from the basic software must be defined, and in addition a component model with the top-level components is necessary. As mentioned before, components can be grouped hierarchically in arbitrary structure, but this kind of break-down can be derived iteratively. The development of the functional architecture on VFB level means a virtual integration of the applications – and this in a very early phase of the development process.

Figure 3 shows an example of the VFB level architecture. The components interact via ports. Ports can have different characteristics. For instance, they can receive or provide information, or they can implement different communication paradigms like sender-receiver-, or client-server-communication, which is indicated by a dedicated notation. The ports use interfaces and data types that are defined in the data model.

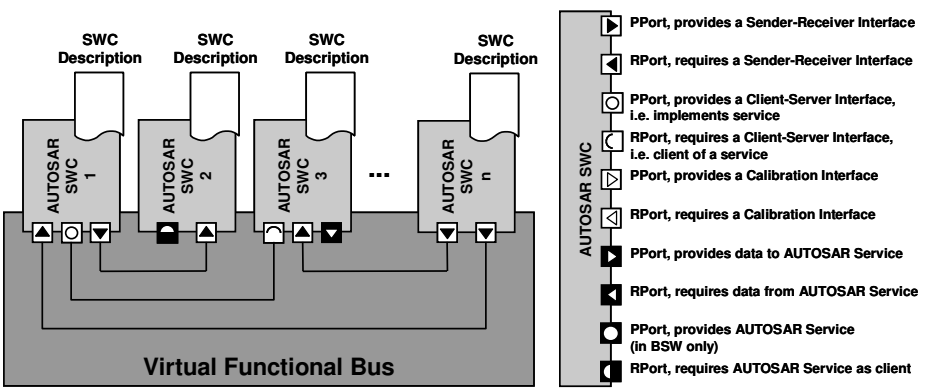


Figure 3: The "virtual functional bus" (VFB) supports a virtual integration

In practice the design of the VFB level architecture is done model-based with a tool. By now several members of AUTOSAR offer VFB level design capabilities in their tools. The design process with such tools is rather component oriented, which means that the data model implicitly will be extended, whenever a component will be extended with a new port, interface, or data type. Such tools furthermore can export the software component description of single components so that the further component development and implementation can run separately.

## 2.2 Component Development and ECU Integration

An AUTOSAR software component encapsulates an application which runs on the AUTOSAR infrastructure. After the VFB level design yielded the initial software component description, the further component development is independent from other components or from system design steps.

The major task of the component development is the implementation, i.e. the coding of the functionality or algorithms. This could be done either as direct coding in a programming language which is in the AUTOSAR context typically C, or more comfortable by means of a model-based design tool and automatic code generation.

In addition to the implementation, the component developer has to complete the software component description with the definition of runnable entities, events and interrunnable variables. Runnable entities are the smallest code parts schedulable to tasks of the operating system. And a component can consist of multiple runnable entities. With this information in the software component description the interface of the component towards the highest layer in the basic software – towards the Runtime Environment (RTE) – is defined. From the viewpoint of the component, the RTE implements the VFB functionality on a specific ECU.

In case of model-based design the software component description will be completed within that tool, and exported e.g. together with the code generation. The generated code includes even the header towards the RTE so that the component can be compiled.

In case of conventional development, the software component description has to be completed with a specific template editor, which in practice often is part of basic software configuration tools. Such tools commonly can generate the RTE header as well.

As mentioned before, building the ECU software requires appropriate basic software for the ECU and all application software components mapped to the ECU. Although the basic software can be regarded as an off-the-shelf product, it needs a configuration which of course depends on the application. Therefore the basic software configuration tool has to read all software component descriptions of the concerned applications. For a proper configuration it furthermore has to read the ECU extract of the system description and the ECU description. Then all the ECU software, i.e. the configured basic software and all application software components can be compiled and linked.

## 2.3 Standardized Application Interfaces

AUTOSAR specifies the interfaces of typical automotive applications from all domains regarding syntax and semantics [AI40]. This specification is aggregated in a common table, which contains by now nearly 2500 different ports and more than 500 interfaces. These are clustered into about 40 different compositions, and they use 770 data types and 26 units. The following example shall illustrate the context of these elements.

A composition, e.g. “Mirror Adjustment” contains different ports, e.g. “MirrorPosition” or “MirrorMoveStatus”. Both ports apply the interface “MirrPosnSts1” for describing the actual mirror position, either for status inquiries or for requests. The “MirrPosnSts1” consist of only one data element, i.e. here the data record “MirrAxisPosn1”. The data record has three elements: mirror type, axis type and position. Each element can be either a continuous value or an enumeration. In this example the position is a continuous value of type “perc7” – an unsigned integer with recommended length of 14 bit, unit percent, a resolution of 0,1, a physical range of 0,0 to 1001,0. In contrast the enumeration axis type means 0 no axis, 1 horizontal, and 2 vertical axis.

The entire specification of application interfaces serves as a standard for application software. The usage of these interfaces limits the development effort to adopt application software components to for example, different vehicle platforms. Since AUTOSAR focus is set to mature application interfaces innovation in software functionality is not limited. New functions shall stay proprietary. This enables the OEM and software component developers to keep competition alive. In consequence the number of standardized application interfaces will grow over time.

## 3 Conclusion

Applying AUTOSAR to embedded automotive software development yields several important advantages. Due to the architecture that provides standardized infrastructure functions for application software, the application software itself is independent from the hardware. The VFB concept allows for developing the functional architecture of the application functions without any need of considering the underlying hardware with ECUs and network. This virtual integration means a frontloading of development work, so that integration issues could be identified and resolved early in the design process. The AUTOSAR approach provides very high flexibility in the software development. Roles and responsibilities can be easily split and therefore software can be handled as a product.

## References


[AUTO] AUTOSAR: Official Website of the partnership, http://www.autosar.org.

[AI40] AUTOSAR R4.0 Specification: Table of Application Interfaces

[AM40] AUTOSAR R4.0 Specification: Methodology Model

[BF08] Bunzel, S.; Fennel, H.: The AUTOSAR Methodology. In (VDI): FISITA World Automotive Congress, Springer Automotive Media, September 2008; FISITA 2008/F2008-10-023.

[Fü09] Fürst, S; et al.: AUTOSAR – A Worldwide Standard is on the Road. 14th International VDI Congress Electronic Systems for Vehicles 2009, Baden-Baden

[KG08] Kinkelin, G.; Gilberg, A; et al: AUTOSAR on the Road. In (CTEA): SAE Convergence, October 2008; 2008-21-0019