AUTOSAR in the Development

Procedure for Introducing Model-Based AUTOSAR Function Development

Process

into Production Projects

The AUTOSAR standard defines a software architecture, software interfaces and interchange formats for creating function libraries that are independent of ECU suppliers. Daimler AG is introducing AUTOSAR in the fields of comfort and vehicle interior functions as part of its activities for standardizing function library software modules. This process includes using the autocoder TargetLink for modeling AUTOSAR software components and generating code from them.

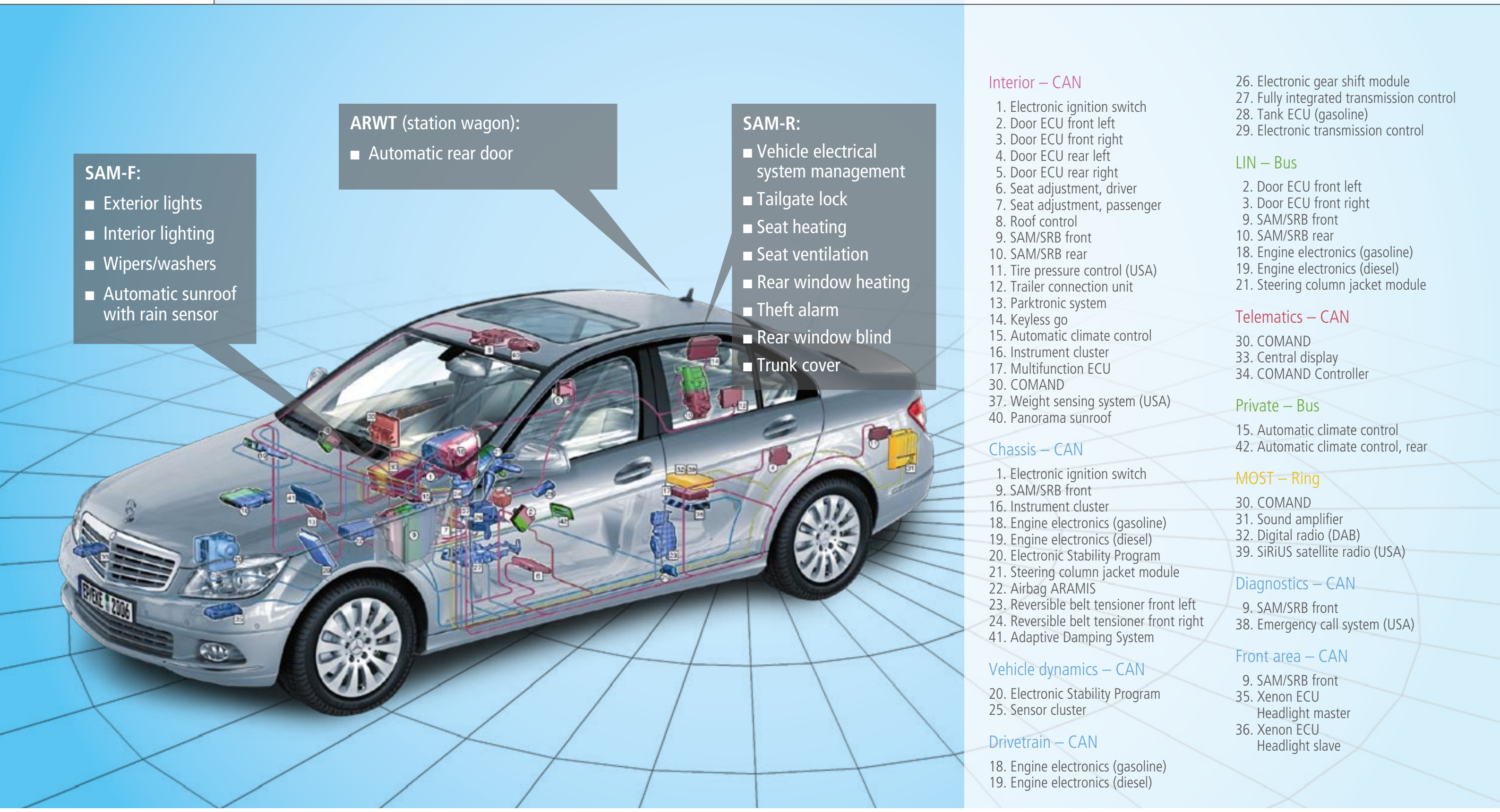


Fig. 1: Comfort functions for the C class (204 series) designed via modeling and automatically generated production code (SAM: central body computers ARWT: automatic hatch door).

Model-based design and automatic production code generation are used for passenger car development at Daimler AG. This article describes the step-by-step method of converting a model-based development process to the AUTOSAR architecture and reports on experience gathered so far.

Model-Based Development of Vehicle Interior Functions

For some years now, Daimler AG has used modeling to develop vehicle functions, especially for comfort and interior functions. Model-based development for the vehicle interior focuses on the main functions of the central body computers, each of which has integrated multiple functions. Model-based development was introduced on a large scale for the current C class (204 series) (fig. 1). The model-based approach is systematically applied to the series currently in production development, and the number of functions to be integrated is constantly increasing. Some of the functions we develop comprise the entire ECU application software. Others are what are called added value functions, i.e., extra functions added to existing ECUs to differentiate them from the competition. In each development cycle, the supplier is given the function model that we developed, plus a test specification, and is responsible for producing software for the model and implementing it on the ECU (fig. 2). Essentially, model-based development gives OEMs the following advantages:

- Early maturity validation by simulation
- Functions represented and validated by rapid prototyping in the vehicle, even before the ECUs are available
- Direct implementation by autocoding
- Supplier-independent function development and further development
- Reusable functions
- Protection of intellectual property

Integration into Software Architecture

At present, when functions are created via modeling, suppliers still have a high manual workload when integrating them into the ECU. The amount of work involved greatly depends on the software architecture used by each supplier, even when the OEM has specified the communication part of the basic software. In some cases, the software architecture has to be adapted or specially extended. There is no completely standardized software architecture, so sometimes extensive coordination meetings have to be

held with specific suppliers.
The need for coordination goes beyond just the software architecture, however. The OEM and the ECU supplier also have to jointly define the descriptions of metadata for integrating functions, such as the interface lists for the functions and the mapping of application signals to bus signals. Thus, the prerequisites for broader and process-safe use of model-based development are a uniform, supplier-independent software architecture and a standardized description of the metadata.

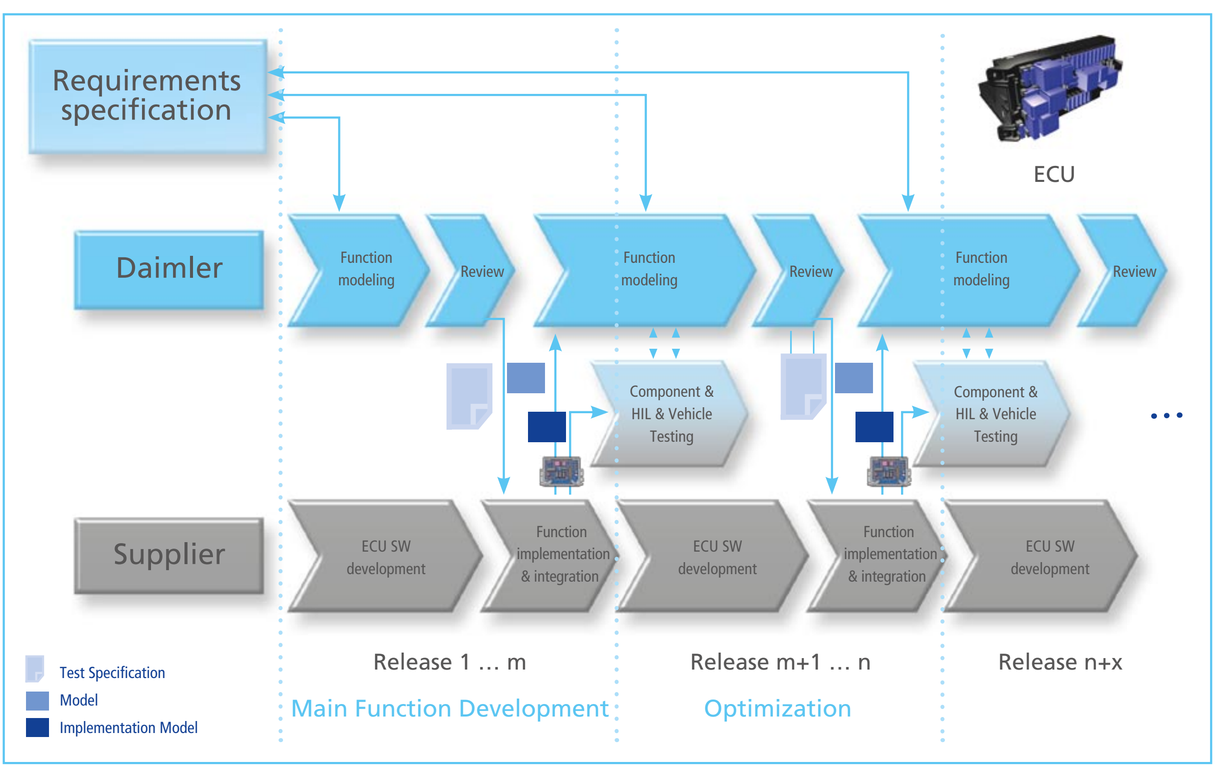
The AUTOSAR Software Architecture

The AUTOSAR standard defines a software architecture for ECUs, an integration method, and the interchange formats that these require. In other words, AUTOSAR largely addresses the requirements for the process-safe integration of model-based functions described above.
AUTOSAR divides the application software of an ECU into several software components (SWCs), which communicate with one another via middleware (the run-time environment, or RTE). SWCs encapsulate the software and give it type definitions, allowing data exchange only via well-defined interfaces.
Two mapping steps are needed for integration on an ECU: First the SWC instances are mapped to ECUs, and then the data elements are mapped to network signals for communication across ECUs.

Strategy for Introducing AUTOSAR

For the next generation of vehicle interior ECUs, Daimler AG is taking the first step towards an AUTOSAR architecture. The introduction process starts at the top of the layer model for AUTOSAR software architectures and works its way down, i.e., it begins with the application software components (SWCs) and the RTE (fig. 3). This will involve systematically dividing the software architecture into application parts and basic software parts, which will communicate via a defined interface. The basis for defining this interface is the AUTOSAR standard, though this is partially restricted here and has also been more precisely formulated at some points. In this first step, the standard software will still be based on the established Daimler Standard Core, to which selected AUTOSAR software services (such as memory management for NVRAM data) will be added.
In this early phase, the ECUs developed in this way will still be network-compatible with ECUs

Fig. 2: Typical development cycle with multiple iterations.



developed by the classic method. This means that AUTOSAR technology can be introduced step by step.

Modeling at Behavior and Architecture Level

With AUTOSAR, modeling is performed at two levels: Beside the behavior level, where the behavior of the functions is modeled, there is an architecture level where the interfaces of the SWCs and their connections have to be formally described.

In a top-down strategy, when new vehicle functions are developed it is useful to first subdivide their functionality into several SWCs and then to define their interfaces at architecture level (fig. 4). Then the behavior of the resulting SWCs is modeled.

For previously existing function models, a bottom-up procedure can be used to generate the SWC descriptions from the model interfaces. The resulting SWCs are then connected with one another at architecture level.

Stepwise introduction means that it is not possible to produce a complete top-down design of a whole vehicle. It also means that at the level of single ECUs, not all functions are initially available as models. A meet-in-the-middle strategy has therefore proved useful, with the following steps being performed for each ECU:

- The SWC descriptions are derived from existing behavior models.
- For new vehicle functions, the interfaces are first defined at architecture level via SWCs.
- Sensor/actuator SWCs are generated automatically according to fixed rules.

The resulting SWCs are then collected together in a composition at architecture level and networked with one another. The remaining unconnected ports are led through to the outside, which turns them into ports in the composition. The ports now represent the communication interface of the ECU. Data elements referenced via the ports can be mapped to the signals specified for the ECU by the communication matrix. This makes it possible to create the SWC structure of an ECU at a reasonable cost.
The next section explains these activities in greater detail.

Defining the AUTOSAR Interfaces

The basis for developing software components is a shared database or common object pool (COP) of interface and data type definitions that the SWC developers can use when defining SWC ports. The pool was derived from the signal definitions in the communication matrix and must be constantly reconciled with this (fig. 8).

The derivation rules state that one interface with a data element must be generated for each communication matrix signal. This facilitates the transition from the current communication matrix signal world to the AUTOSAR world, since the same names are used and the corresponding interfaces are therefore easy to find. AUTOSAR's structural options for collecting several data elements together in one interface are used

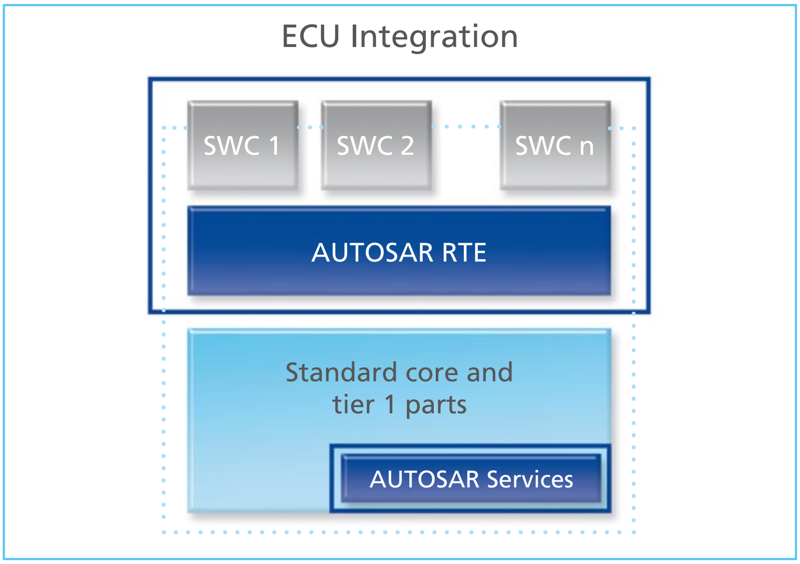


Fig. 3: In stepwise AUTOSAR introduction, first the application software (SWCs) and the middleware (RTE) will be developed in compliance with AUTOSAR. The basic software will still consist of the Daimler Standard Core, extended by selected AUTOSAR software services.

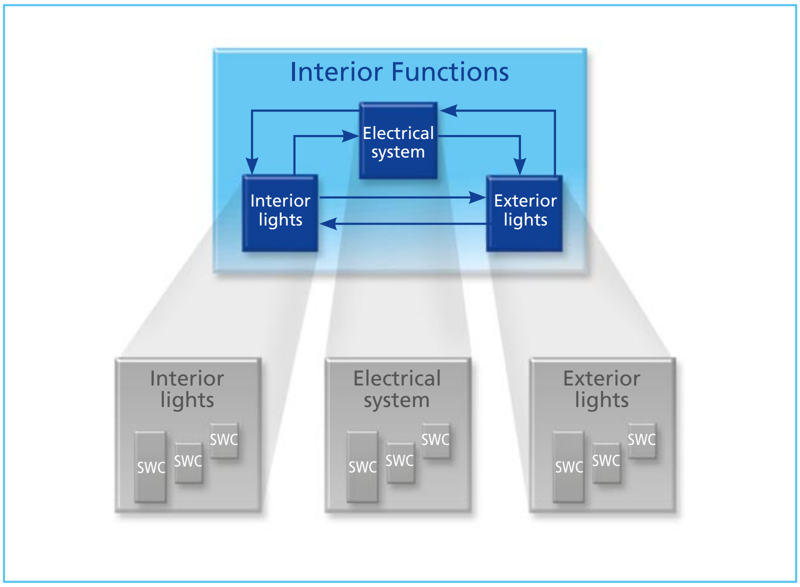


Fig. 4: In the top-down approach, first the functionality is divided into several SWCs, and then their interfaces are defined at architecture level.

Glossary

Composition – In a composition, several SWCs are grouped and connected to form one superordinate component.

DBC file – Database container file. Communication matrix description file for CAN.

Communication matrix – Describes the signals/data exchanged between individual ECUs.

NVRAM – Nonvolatile random access memory, used for storing diagnostic information, etc.

Runnable – An executable element in an AUTOSAR SWC, comparable to a function.

System architecture tool – Used for planning and integrating complex ECU architectures. It can import, export, and modify AUTOSAR files.

for new interfaces. The derived interfaces will not profit from this for the time being.
To ensure that the SWCs are compatible with one another, modeling rules are used to ensure that the only interfaces used are the ones in the pool. If necessary, a coordinator must add further interfaces to the pool.

Creating New Vehicle Functions with the Top-Down Strategy

The modeling of new vehicle functions begins at architecture level, with the descriptions of the SWC interfaces in the COP database described above. This involves creating ports with types defined by interfaces. Runnables and their subsets of the SWC interface are also defined. RTE events are used to define the runnables' cycle times and, if necessary, event triggers such as the receipt of a new signal. Finally, an AUTOSAR SWC description is generated from this.

This AUTOSAR description is the basis for behavior modeling. The SWC descriptions are imported into the Data Dictionary and a modeling frame is generated as the starting point for modeling a behavior function. The modeling frame contains the relevant AUTOSAR interfaces and runnables, using the TargetLink AUTOSAR blocks (fig. 5). The frame is then filled with standard blocks for modeling the actual algorithms. Any changes to the interface are always performed in the database.

Migrating Existing Models in the Bottom-up Strategy

Existing behavior models were migrated, using the interface definition for deriving and typing the ports. We were able to automate virtually 90% of model migration, since we already had modeling conventions for using the names of the associated communication matrix signals for input and output variables.

Fig. 5: Frame model created with TargetLink.

First the SWC was created in the Data Dictionary, then the appropriate AUTOSAR blocks were inserted in the model (fig. 6). Finally, the XML file (SWC description file) created during code generation was loaded to the SWC database. The interface can now be maintained in the database.
The modeled functionality (behavior model) was migrated in its entirety and with no modifications; only the model interface had to be adapted to the standard (fig. 7).

AUTOSAR Tool Chain

An existing tool chain was extended for developing AUTOSAR-compliant vehicle functions (fig. 8). The common object pool for storing AUTOSAR interfaces and SWC descriptions is a major component of the development environment. It was implemented as an extension to an existing database application used for creating communication matrices. The pool not only contains AUTOSAR modeling elements; in it, parameters and NVRAM variables can also be described and given values. Because of its similarity to the communication matrix, AUTOSAR interfaces can be generated elegantly from the data in the communication matrix and subsequently updated. The tried and tested release concept used for communication matrix development can also continue to be used.
The SWC descriptions can be imported and exported via the AUTOSAR interchange formats. This supports the connection to the Data Dictionary and to system architecture tools. The model-based development of AUTOSAR-compliant function models is supported by model-in-the-loop (MIL) and software-in-the-loop (SIL) tests, both integrated in the development environment. Finally, AUTOSAR-compliant production code together with a consistent AUTOSAR SWC description file is generated.
The system architecture tools are used to network the SWCs. This is usually done in graphical editors, which provide better visualization than purely form-based database applications. The system architecture tool is now being used to map the non-local data elements to network signals. The resulting AUTOSAR system template is used for ECU integration at the supplier.

Conclusions and Outlook

AUTOSAR is answering a long-standing need to standardize the description formats and interfaces for model-based function development. The type-safe AUTOSAR descriptions make it possible to ensure consistency between separately developed function models at the very early stage in the development process when we as the OEM hand over the function models to the ECU supplier. The expectation is that this will make the function integration by the supplier much more efficient. Coordination meetings held between the OEM and the supplier to discuss software architecture are considerably more productive because both sides can use terms that are standardized by AUTOSAR. The current necessity of using two development tools (for modeling behavior and for describing interfaces) in developing AUTOSAR-compliant function networks poses new challenges, as each system has to be divided into manageable and logically useful software components. In the future, the transitions between different modeling tools, often from different vendors, will have to be made more efficient to ensure a 'round trip'. Moreover, the current division of AUTOSAR development environments, into tools for architecture modeling and system integration and tools for behavior modeling with their own

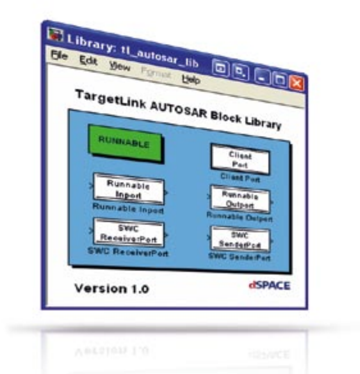


Fig. 6: The functions are modeled with TargetLink's AUTOSAR blocks.

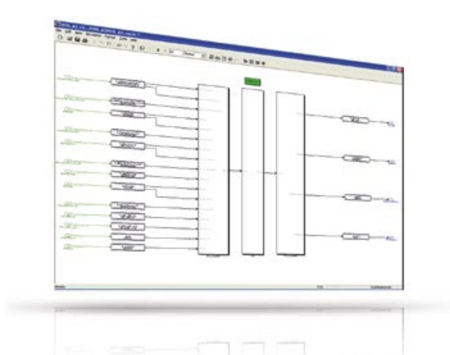


Fig. 7: Modeled runnable with appropriate interface blocks after AUTOSAR migration.

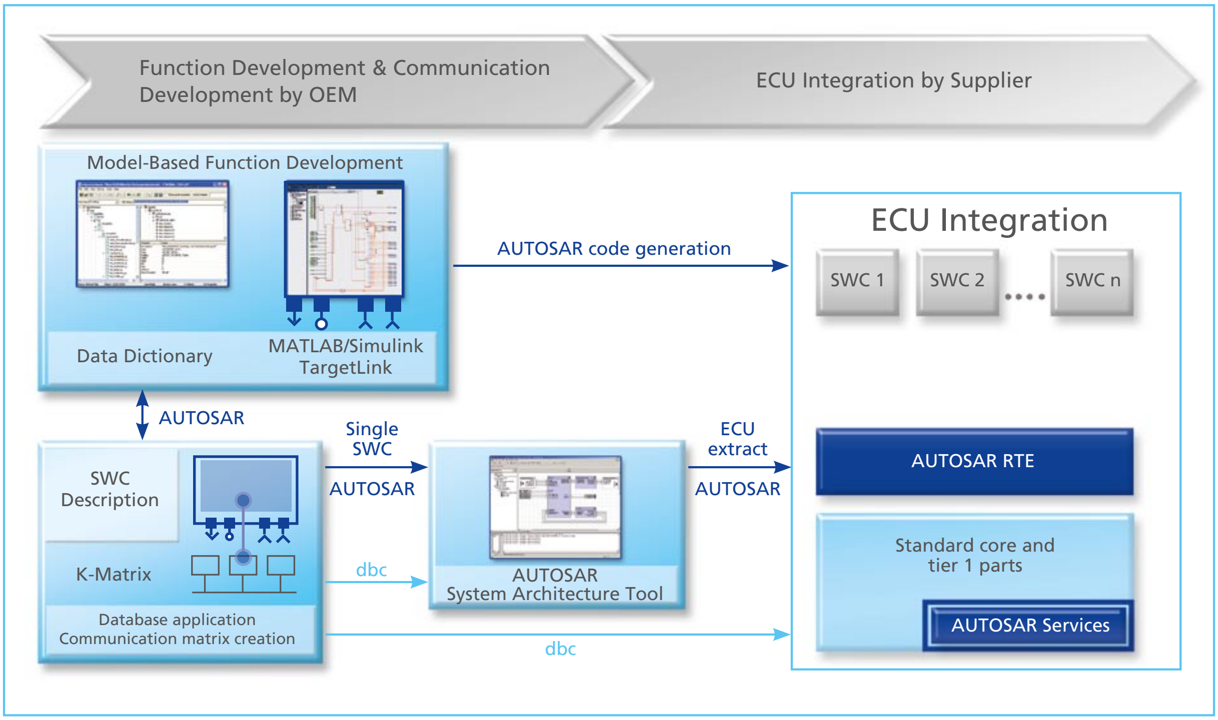


Fig. 8: AUTOSAR tool environment for generating AUTOSAR software components (SWCs). Data exchange between tools is based on AUTOSAR XML and DBC files.

autocoders, is to a large part due to the tool domains. This limits the potential for tool-independent system modeling and resource optimization. Architecture and design decisions therefore need to be thought through carefully at the outset. Because the AUTOSAR description is so extensive, and because individual tools do not yet provide complete equivalents, developers have to begin by deciding on a subset of the standard via suitable application profiles.
As we see it, AUTOSAR is capable of evolutionary introduction into the development process as it is known today. With a stepwise approach to introducing AUTOSAR, we can continue using existing function models as well as proven processes and tool chains, and optimize them step by step.

Due to its success, using function modelling to develop comfort functions and interior functions has become a standard method at Daimler AG. Much of the experience gathered in these projects can be transferred to AUTOSAR-compliant development and will make it possible to replace manual steps by automated and standardized process steps in the future.

Since the definition of the AUTOSAR standard is not yet complete, close cooperation with the standardization groups and tool manufacturers continues to be necessary to ensure that investments made in converting to AUTOSAR are future-proof. ■

Christian Dziobek, Dr. Florian Wohlgemuth
Mercedes-Benz Cars Development
Dr. Thomas Ringler,
Group Research & Advanced Engineering
Daimler AG, Germany

Summary

- Daimler AG is introducing AUTOSAR in a production project for vehicle interior ECUs.
- Existing tools and processes were extended for AUTOSAR.
- Existing models can be reused with the AUTOSAR-compliant code generator.