



# Collection of Professional Articles



Summary of AUTOSAR Competence



ENGLISH

PDF version. Print only for private use



## 2008 – Time for AUTOSAR

Dear reader,

AUTOSAR represents a breakthrough and a milestone in the development of modern automotive technology. This future-oriented standard for ECU software in the vehicle improves reusability and interchangeability for automotive OEMs and suppliers.

For years now, Vector has been an active participant in implementation of this forward-thinking technology. The expertise we have acquired has made us an experienced partner when it comes to AUTOSAR. This has resulted in successful evaluation of Vector AUTOSAR software in numerous practical projects and a production-mature AUTOSAR basic software package.

Much of our knowledge and experience from these AUTOSAR projects has been compiled in numerous technical articles. We are providing these articles to you in the form of a compact collection – we hope that you find them to be stimulating and informative in the context of your future projects.

2008 is the year of AUTOSAR. We look forward to supporting you in the implementation of your AUTOSAR projects.

I wish you much reading enjoyment!

Sincerely,



**Ralph Dürr**

Team leader: Embedded Software Sales



## >> Contents

AUTOSAR on its way to production	2 – 4
The Universal Gateway ECU	5 – 8
Early Migration creates Opportunities for Innovations	9 – 11
Successful Integration of Existing ECU Software in the AUTOSAR Architecture	12 – 17
Tool-Supported Integration of Microcontroller-Specific Modules	18 – 20

# AUTOSAR on its way to production

**To master the growing complexity of software in modern vehicles, automotive OEMs are increasingly developing their electronic systems based on AUTOSAR. The standards created by this development partnership simplify development processes and make ECU software reusable. Since its introduction in 2004, this innovative and pioneering technology has been tested in many evaluation projects and is now entering an implementation phase in production ECUs. AUTOSAR standard software covers the current state of technology and is undergoing continual advanced development in new releases.**

The automotive industry is being confronted with new times. Growth in the number of complex vehicle functions is making the development of automotive electronics increasingly more extensive and complicated. Customer wishes, e.g. for more variants and equipment variety in the vehicle, as well as non-functional requirements such as diagnostic capability and availability, are requiring more intensive ECU development processes. Vehicles, in particular luxury vehicles, may have more than approx. 1,000 software functions, several vehicle electrical networks and more than 70 ECUs. Due to the variety of hardware platforms used in this field, ECU software dependencies on the hardware and system configurations being used has become problematic. Each change to one of these constraints requires reprogramming or at least modification of the software.

To make the complexity of ECU software development manageable, the AUTOSAR development partnership has defined a practice-tested software architecture that serves as a foundation for developing reusable applications. This open standard for system architecture – created by automotive OEMs, suppliers and other companies in the global software, semiconductor and electronics industries – lets users avoid proprietary solutions that would result in increasingly high development cost and effort.

AUTOSAR subdivides the electronics architecture into a number of layers and modules. With the simultaneous definition of interfaces, AUTOSAR creates standards for easy exchange of software components or hardware platforms. The development partnership not only provides specifications for the base software modules. It also offers a methodology for developing applications and distributed systems. This methodology begins with a model-based description of the software applications and the distributed system, and ideally ends in an automatically generated and reproducible test. This formal approach simplifies the use of a universal tool chain.

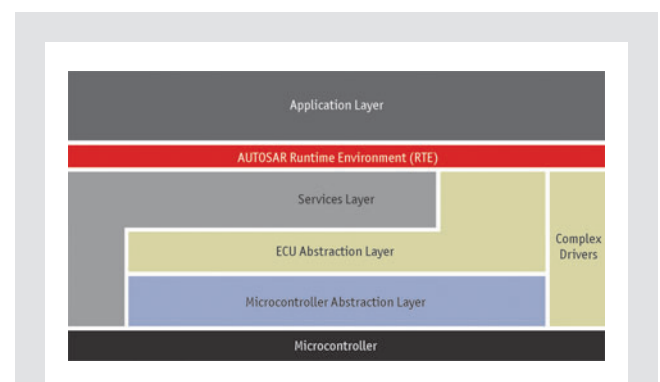
A full three years after its start, the AUTOSAR development partnership published Release 2.1 at the beginning of 2007. A stable level was reached with this release, and it has been tested in several validation projects for its practical utility. At many businesses, the action item of “AUTOSAR evaluation” has been successfully completed. Now it is ready to be implemented in concrete production ECUs.

## AUTOSAR architecture

To realize the objectives of AUTOSAR – separation of the application software from basic modules and functions – the vehicle electronics

is abstracted and subdivided into several layers (Figure 1). The connection to the actual microcontroller, i.e. the physical foundation, is represented by the Microcontroller Abstraction Layer, which maps the microcontroller’s functions and periphery. It defines interfaces for memory, the I/O driver and its communication connections. In this layer, features that the microcontroller does not offer may also be emulated in software. The second layer that lies above this is the ECU Abstraction Layer. It establishes die ECU-specific hardware layout and provides drivers for the periphery external to the ECU, for example. In another layer, the Services Layer, various basic services are represented 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 basic software (BSW) is made. The RTE handles integration of the application software and regulates the exchange of data between the application and the BSW. It is only at this layer that reuse of already written application software is possible, because the defined interfaces make it possible to develop the application software without special knowledge of the hardware to be used at a later time, and the software can be used for any other AUTOSAR-conformant ECUs.

The so-called Virtual Functional Bus (VFB) forms the basis for configuration of the layers. All components of the vehicle’s electronics communicate in an abstracted view via this bus. The software components use ports for this, whose interfaces can be defined as port interfaces. Connectors connect the ports. It is irrelevant to the VFB whether this is a connection within an ECU or a connection via an external bus. This is not decided until the final system layout is made and specific functions are assigned to a specific ECU.



**Figure 1:**  
**AUTOSAR layer model for ECU software.**

The software component itself does not require any knowledge of this later distribution, and it can therefore be developed independent of it. The component is subdivided into executable units, so-called Runnable Entities, which are executed like procedures when a specific event occurs. Such an event might be the arrival of a new sensor value or a periodic activation, for example. The description of the electronic system formulated from the perspective of the VFB defines the interfaces of the specific software components. As a result, the application components can be developed independent of the specific ECU.

The “counterpart” on the ECU is the RTE. It guarantees access to I/Os, memory and other basic services. Using the model-based description, the RTE is generated ECU-specifically, which means that it can be adapted to different requirements while economizing on resources.

### Methodology

Besides defining the ECU’s software architecture, the AUTOSAR standard also defines a methodology to help in the development of AUTOSAR systems. Conformance to a structured creation process is recognized as an important precondition in creating error-free software. Deficiencies in the requirements list are detected early, reuse and porting of software components is simplified, and the overall system is more reliable. Nonetheless, this methodology also allows certain freedoms: for example, users can decide whether to use a top-down approach or a bottom-up development.

The AUTOSAR Initiative provides universal support for the software development process by tools. Mature tools are used for structured implementation of requirements and their consistent management; configurations are systematically created, and complex dependencies are automatically taken into account.

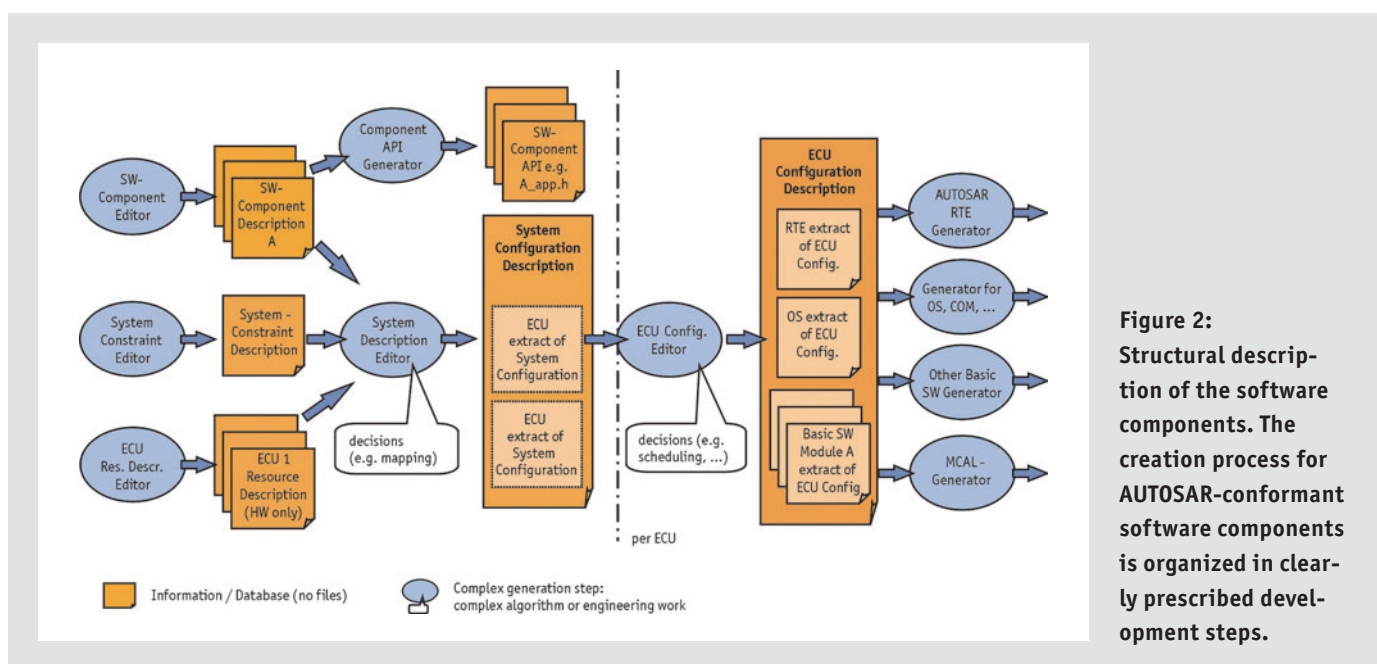
The first step involves formal description of the three primary constituents: Software (SW Components), ECUs (ECU Resources) and System Constraints.

Suitable editors are used to create a configuration description of the complete system (Figure 2). This system configuration serves as the basis for the ECU Configurations that the user creates for the individual basic software modules with the help of configuration tools. At the end of the process, multiple generators supply the ECU-specific implementation of the RTE and basic software.

All design and configuration data produced in the development process are described in a uniform format. AUTOSAR has defined an XML-based format for this purpose. On the one hand, it guarantees universality of the development process, and on the other it also simplifies seamless integration of necessary and auxiliary tools.

### Migration

The software architecture of an AUTOSAR ECU is not monolithic, rather it consists of a number of standard modules with well-defined interfaces. This enables implementation of migration solutions that offer a risk-free transition to AUTOSAR. Such a migration solution must typically be worked out project-specifically. In prac-



**Figure 2:** Structural description of the software components. The creation process for AUTOSAR-conformant software components is organized in clearly prescribed development steps.

tice, this may involve a mixed configuration of standard AUTOSAR modules and proprietary software.

To work out a migration solution, one begins by comparing the existing custom software and the AUTOSAR architecture. After an analysis of overlapping functionalities and integration options, a decision is made regarding which modules will remain and which ones can be replaced by standard software.

So, it is advisable, for example, to introduce a separation layer between the application and basic software. A potential scenario in this case is to prepare the applications as AUTOSAR software components already in early migration phase, and to integrate them via an RTE. Beneath the RTE, a specific adaptation layer is used for interfacing to the existing basic software (Figure 3).

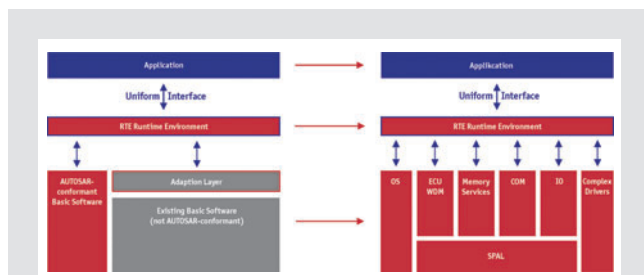
If parts of the existing basic software are to be replaced by AUTOSAR basic software, the emphasis should lie on the use of a uniform configuration tool for both worlds. Vector offers suitable tools

here, which are flexible enough to be used to configure proprietary basic software too. Non-AUTOSAR modules can now be replaced by AUTOSAR modules in successive steps, without putting the overall architecture at risk or requiring reprogramming to other modules.

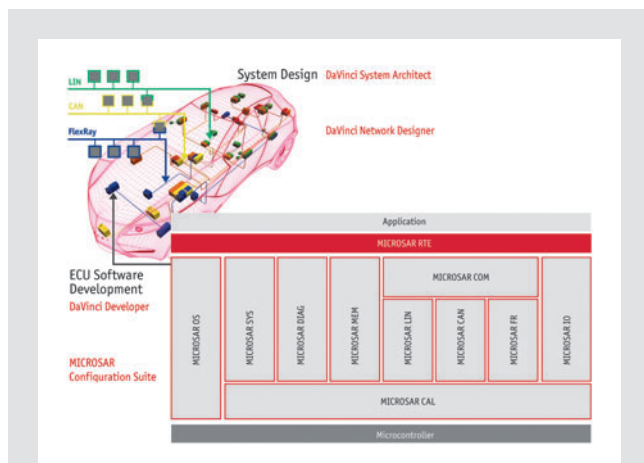
### Outlook

The current AUTOSAR Release 3.0 represents another step taken to enhance the quality of the AUTOSAR standard. The continuing involvement of participating companies clearly demonstrates commitment to the goals of the AUTOSAR development partnership. Ideas are currently being introduced that should become a reality in the framework of the future Version 4.0 of the AUTOSAR standard.

New ideas related to AUTOSAR are also being generated by tool suppliers. Current developments at Vector are aimed at making AUTOSAR-based ECU development as convenient and efficient as possible. One example is a PC-based test tool for AUTOSAR application components, which serves as an emulator for the AUTOSAR ECU environment on the level of the VFB. This makes it easy to test the implementation code of the AUTOSAR software components on a development PC. Widely used standard tools such as Vector's CANoe may be used for test execution, visualization during or after testing, and generation of test reports. Vector supports all phases of the ECU development process with its full set of AUTOSAR basic software and a universal tool chain of design and development tools (Figure 4). The available Vector solution has been tested in practice through its use in evaluation projects, and it is production-mature for AUTOSAR Release 2.0 or 2.1 (3.0 too starting in the 2nd quarter of 2008).



**Figure 3:**  
Early introduction of a uniform interface and RTE simplifies migration.



**Figure 4:**  
The Vector AUTOSAR solution: From system description to standardized ECU software.

### Summary

AUTOSAR is becoming a reality. Various OEMs have concrete plans for implementing AUTOSAR in upcoming vehicle production programs. Vector offers a comprehensive product lineup for this, as well as basic software and tools related to AUTOSAR. Not only does this enable the development of pure AUTOSAR systems; it can also support a stepwise migration toward AUTOSAR.



#### Matthias Wernicke

(Graduate engineer) is responsible for product management of the DaVinci Tool Suite and is also actively engaged in AUTOSAR standardization work.

# The Universal Gateway ECU

## Flexible post-build configuration of AUTOSAR gateways

High flexibility of gateway ECUs is achieved by the post-build principle, since it permits a later configuration at any time – even in late development phases during ECU integration or even in the field. This results in universally deployable ECUs. Based on the AUTOSAR standard, a method is presented here that describes how the gateway functionality in the finished ECU can be adapted to new requirements.

Embedded systems can be configured at different points in time: Before the source code runs through the build process, the so-called pre-compile configuration occurs. This enables efficient implementation of configurations, e.g. variants, by macros or C pre-processor switches. The link-time configuration is typically used to generate a library and to link it with ROM constants. Furthermore, there is the option of configuration during the run-time of the ECU (run-time configuration), e.g. by calibration or diagnostic commands. In such cases, the configuration parameters must be stored in RAM. In contrast to the configuration types already named, post-build configuration is performed on ECUs that have already been built by loading the configuration data into the ECU via a flash bootloader (Figure 1).

The AUTOSAR standard [1] defines three so-called Configuration Conformance Classes (CCC), which cover these different configuration times:

- > CCC0 refers to a Pre-Compile Configuration
- > CCC1 Link-Time Configuration
- > CCC2 Post-Build Configuration.

Just which of these configuration options is in principle available for a specific basic software module depends on the character of the module. The RTE (Runtime Environment) supports only CCC0, because it is closely linked to the applications and consists of fully generated code. The operating system is also configured only at pre-compile time. Figure 2 shows – in the context of the AUTOSAR layer model – which configuration options exist for CAN communication modules.

For the most part, the modules belonging to the communication stack beneath the RTE support the post-build configuration per CCC2. Nonetheless, these modules have some pre-compile parameters such as the number of channels, use of data buffers (queues) and activation of debug modes. These settings must be known at

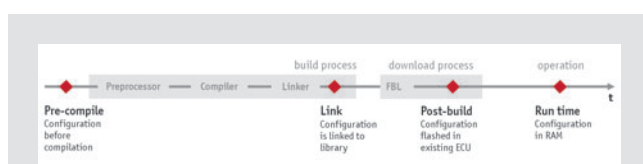


Figure 1: Configuration concepts in ECU development

the time the library is generated. In designing an ECU, a sensible strategy must be developed for using the post-build capabilities of neighboring modules. For example, it would make little sense to provide post-build capability to an individual module such as the PDU Router (PDU-R), while only granting pre-compile configuration to all other modules.

### When is a post-build implemented for gateways?

If the functionality of individual ECUs changes, e.g. during a vehicle facelift, often changes must also be made to the communication matrix of one or more networks. This is where post-build comes into play and enables simple adaptation of the basic software responsible for communication between all ECUs of the affected network.

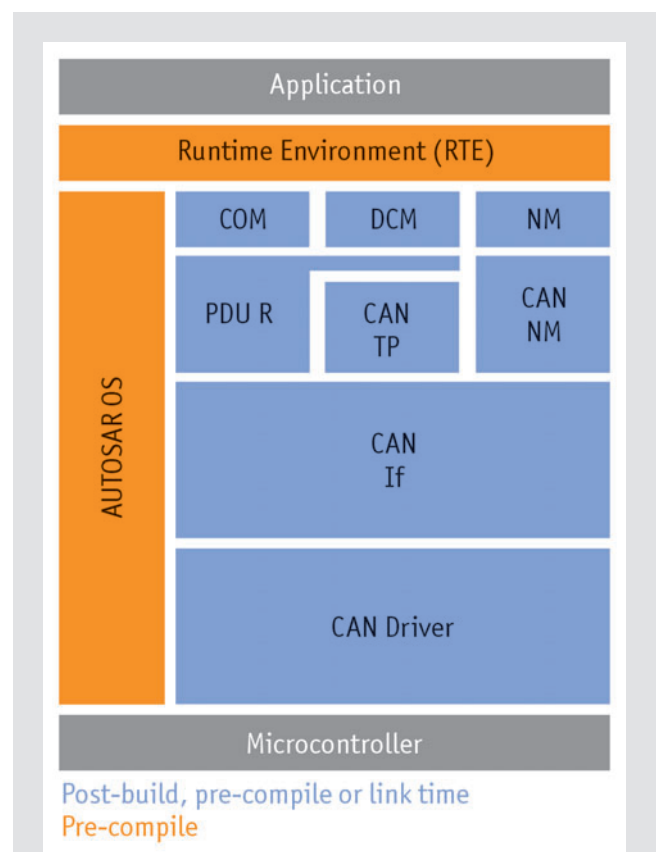


Figure 2: Configuration classes in the layer model of an AUTOSAR ECU

This eliminates the need for recompiling and linking the code. If a gateway ECU is designed to be post-build capable, it is easier to fit it in as a standard ECU in different vehicle models. Gateway ECUs perform a majority of their tasks via the communications basic software. A vehicle facelift for such ECUs often consists of just new or modified routing relationships, for which all that is needed is a reconfiguration of the basic software.

The primary motivation for using a post-build configuration is the fact that it avoids the need for a new build process, and the configuration can be performed in late development phases during ECU integration or even in the field. This approach is of special interest for gateway ECUs.

### The gateway ECU as flexible switchboard

The main purpose of a gateway ECU is to distribute communication data among the individual networks in a vehicle. According to the AUTOSAR standard, various basic software modules of the ECU perform this task. Which modules are used depends on the specific gateway functionality that is needed:

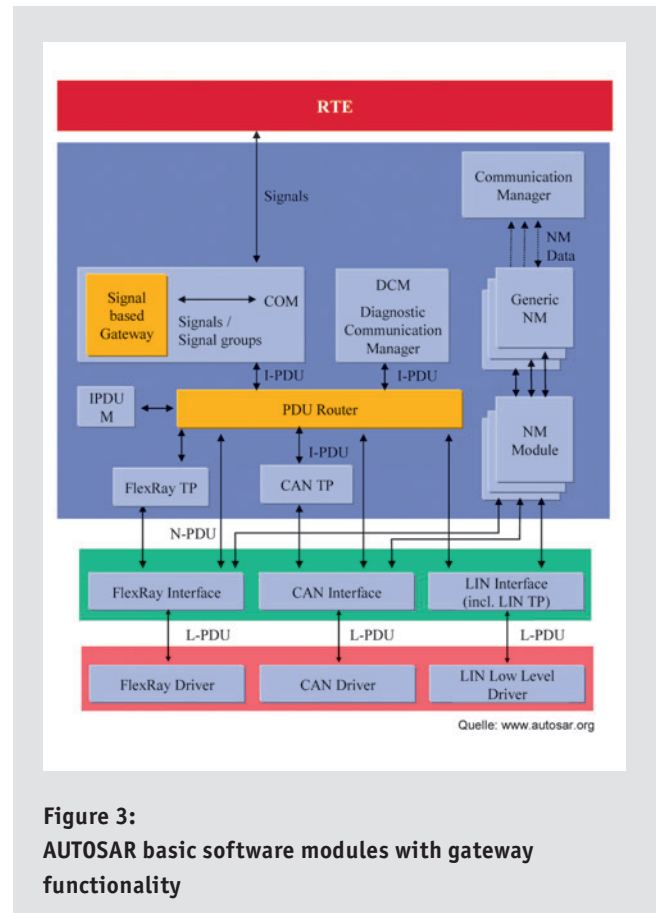
#### PDU gateway

The PDU gateway is a part of the PDU router (Figure 3). It routes entire data packets, known as Protocol Data Units (PDUs), from one network to another. This principle assumes that the PDUs are defined identically on both the source and target networks, i.e. they must agree in length and contents. This means that data can also be exchanged between different bus systems such as CAN, LIN or FlexRay.

Nonetheless, it should be noted that in accordance with the AUTOSAR specification the PDUs must be routed immediately after they are received. Consequently, the PDU router cannot perform send type or cycle time conversions. In some cases, however, such a conversion is necessary. An example of this might be a FlexRay-CAN gateway that routes a PDU from a FlexRay cluster to a CAN network as a CAN message. In this case, the gateway ECU must for example conform to minimum transmission delay requirements (debounce time) for the CAN message. In such cases, the PDUs are therefore routed directly to the COM layer, which then assumes this task.

#### TP gateway

Another task of the PDU router is to route transport protocol data. This comes into play, for example, if the extensive diagnostic data of an ECU need to be automatically transferred from one network to another. This involves receiving the data via the Transport Protocol (TP) and resending it. At first, the transmission occurs above the TP (Layer 4 in the ISO/OSI layers model), and this enables a conver-



**Figure 3:**  
**AUTOSAR basic software modules with gateway functionality**

sion to different addressing methods and various bus systems. To keep delays and RAM requirements in the gateway as low as possible, the TP gateway supports so-called “on-the-fly routing”: The gateway does not wait until all TP data have been received, rather it already begins to resend the data at an earlier time point. Consequently, it receives and sends simultaneously.

#### Signal gateway

Often just individual signals are needed on the other network. In this case, the gateway does not transfer the entire PDU, but only sends individual signals to the other bus. To do this, it first disassembles a received PDU into individual signals, so that it can then assemble them into one or more send PDUs. Besides modifying the signal composition and signal positions within a PDU, the send type and cycle time can also be changed. This method is also used when a PDU should contain both routed signals and signals generated directly in the gateway ECU.

### Technical aspects of post-build configuration

Data structures for the post-build configurable data essentially have two different types of layouts (Figure 4). In the non-fragmented variant 1, the data structures are arranged one directly after another in memory. The individual data structures are accessed via an indirection table located at a static position. The data structures on the other hand may vary in size depending on the post-build configuration. The remaining memory is a contiguous block that is available for other purposes. However, the basic software modules are highly dependent with respect to their implementation. If the basic software originates from different software suppliers, this variant generates a high coordination needs and is therefore impractical.

In the fragmented variant 2, the data structures are always placed at a static position in memory. Here, at the time of design, an assumption is made concerning the largest possible memory usage of individual data structures. In practice, some unused memory between the data structures remains. With regard to runtime behavior, variant 2 is more efficient, since no indirection is needed in data access. Despite the fragmentation, this variant also offers advantages in terms of memory efficiency, since an indirection table is unnecessary.

The use of post-build data structures has an enormous effect on the design of the basic software modules. In the case of the pre-compile configuration, for example, the separation of code and data is

not required. This makes it easier to implement configuration settings very efficiently with macros or preprocessor switches and to generate optimized C functions with the help of code generators. The principle of the post-build configuration, on the other hand, requires strict separation of code and data for post-build parameters. A generator is only available for generating constants tables. The C functions are static. Figure 5 shows how the different configuration concepts appear based on code examples.

A gateway ECU requires knowledge of large numbers of signals or messages. This information exists in the form of data structures in the ECU's memory. As a result, in gateway ECUs the communication layers in the software architecture take up a considerable share of memory and runtime resources. Even when the more efficient variant 2 is selected, the post-build capable layout of the ECU typically causes increased resource requirements.

### A tool chain for post-build configuration of gateway ECUs

Besides the aspects of memory and runtime resources in the ECU, the post-build principle also requires new processes to coordinate configuration parameters between participating development partners and exchange them reliably. One important source of support here is a well-functioning tool chain. Figure 6 shows the tool chain from Vector Informatik, which can also be used for post-build configuration of gateway ECUs.

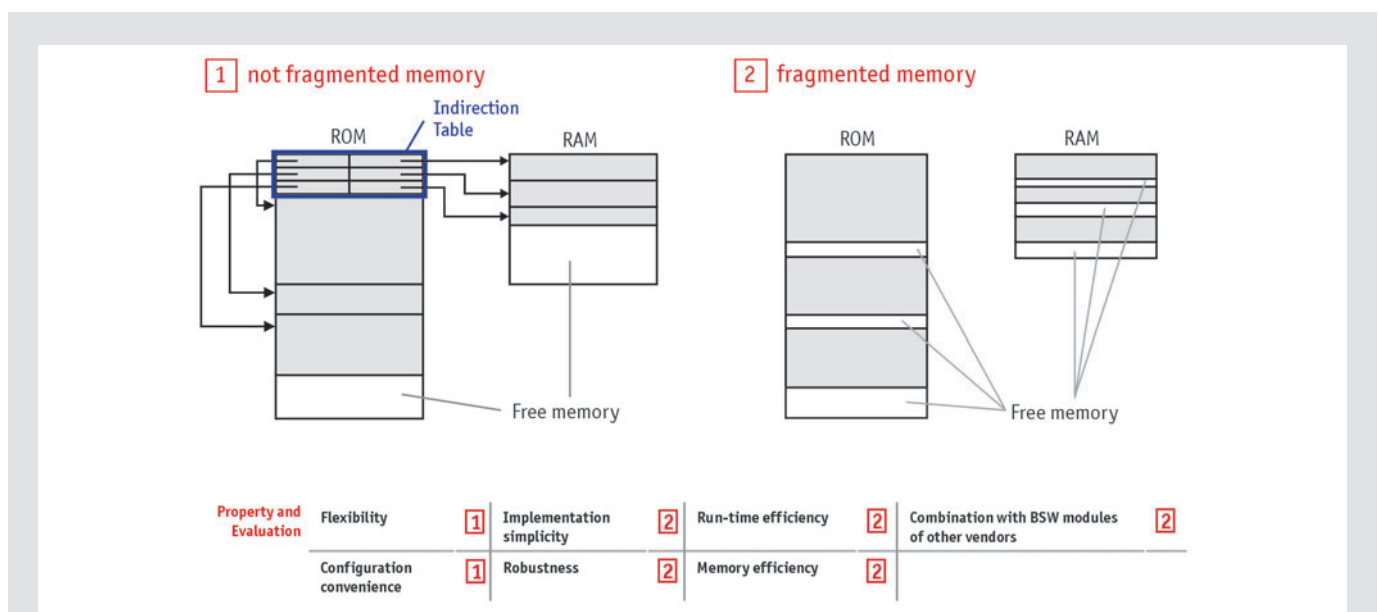


Figure 4: Data structures for post-build configuration

	Pre-compile time	Post-build/link time
Enabling Features	<pre>#define FEATURE_ENABLED #if defined( FEATURE_ENABLED ) ... #endif</pre>	<pre>static boolean FEATURE_ENABLED; if ( FEATURE_ENABLED == true ) { ... }</pre>
Scalar Values	<pre>#define VALUE 8</pre>	<pre>const uint8 value = 8;</pre>

**Figure 5:**  
Code examples for different configuration concepts

At the beginning of a development project, the ECU supplier sets up the project based on the Pre-Config1 parameter set. Vector provides this parameter set along with the base software delivery for the ECU, and it contains pre-compile parameters that do not have any effect on the post-build configuration. Examples of this are general settings and use of the Development Error Tracer (DET).

In coordination with the ECU supplier, the automotive OEM provides the Pre-Config2 parameter set and an initial network description (database). The Pre-Config2 parameter set contains those pre-compile and link-time parameters that have an effect on the post-build process and need to be defined in advance. Examples of this are the addresses of post-build data in the ECU, compiler options and maximum memory size (Flash and RAM). The initial network description, which the automotive OEM might create with the DaVinci Network Designer tool, for example, contains all signals relevant to the ECU. In the case of a gateway ECU, the routing relationships between the networks are also described there.

The ECU supplier uses the GENy tool to configure and generate the basic software using this input information. The routing information is prepared in the form of generated data structures (tables)

for the individual basic software modules. This provides a foundation for developing the ECU based on the initial network description.

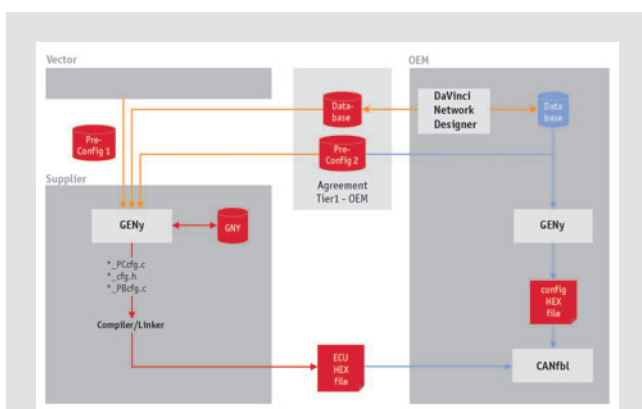
During the actual post-build process, these tables must be regenerated and exchanged in the ECU. The basis for this is a modified network description by the automotive OEM. If the updated tables now exist in binary format – and indeed in precisely the same form as the compiler would have created them – then another compiling and linking run is obsolete. The knowledge about the compiler behavior when generating the binary format is stored in GENy. This binary file is now converted to a standard hex format and is loaded into the ECU via the CANfbl flash bootloader. If the pre-config information is known to the automotive OEM, the OEM itself can also perform the post-build process directly.

### Summary

The post-build process provides flexibility when changes are made in the communication description. Configuration is even possible in late development phases during ECU integration or in the field. This approach is especially useful for gateway ECUs, since it is possible to adapt them to modified network conditions without having to change the complete application code. However, the increased resource requirements must be taken into account. In any event, a gateway ECU is an interesting candidate for the post-build process, at least during the development phase.

### Reference:

[1] AUTOSAR specifications: [www.autosar.org](http://www.autosar.org)



**Figure 6:**  
Vector tool chain for the post-build configuration



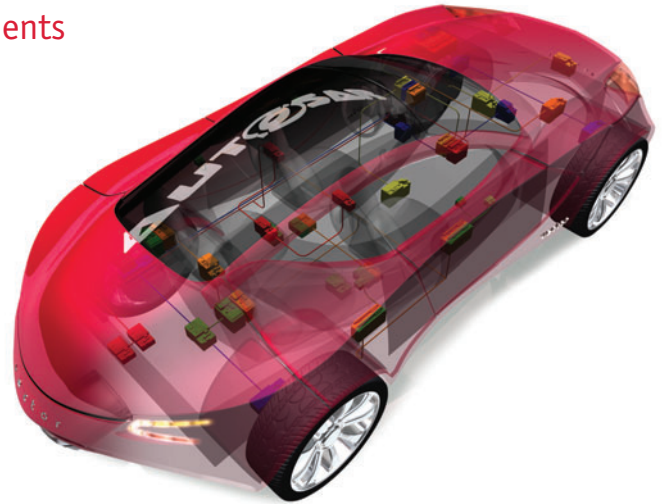
### Hartmut Hörner

studied Electrical Engineering at the University of Stuttgart from 1987 to 1992. Afterwards, he worked as a software developer for ATM Test Systems. In 1998, Mr. Hörner came to Vector where he is the team leader responsible for the development of embedded software components. Hartmut Hörner represents Vector on various standards working committees (OSEK, ISO, AUTOSAR).

# Early Migration creates Opportunities for Innovations

## Mix of Individual Software and AUTOSAR Components in Vehicle Electronics

ECU development in the motor vehicle is evolving rapidly. This article sheds light on one important aspect: The introduction of standardized basic software defined by the AUTOSAR development partnership. If AUTOSAR software components are added to the overall architecture in a stepwise and differentiated manner, this assures quality enhancements.



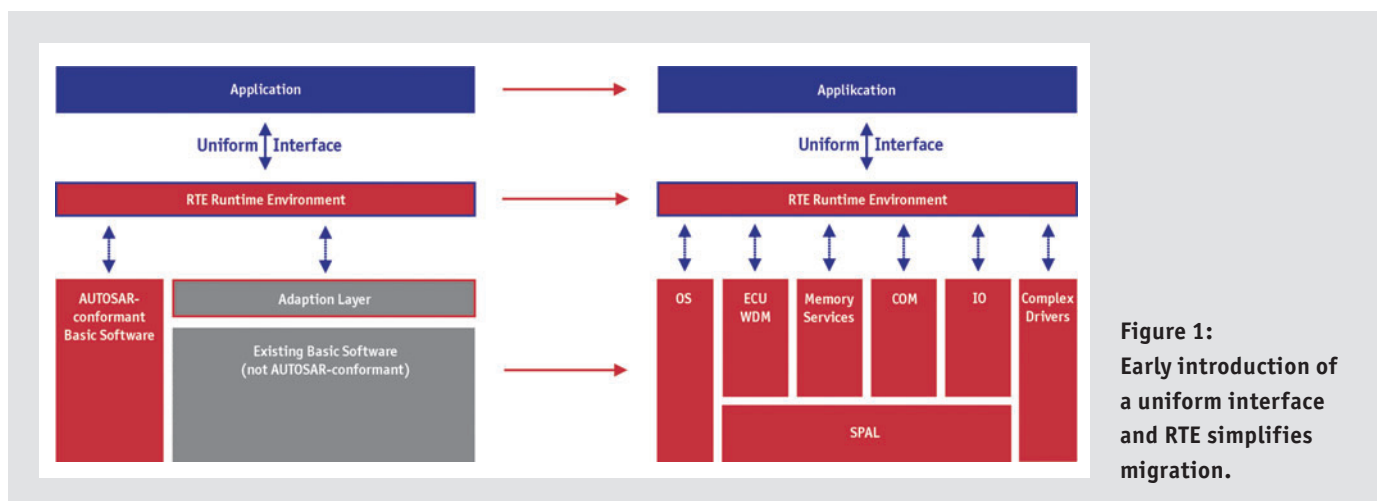
At the beginning of 2007, the AUTOSAR development partnership defined, in its Release 2.1, a practice-tested software architecture that is used as a foundation for developing reusable applications. With publication of these specifications, in the future it will be possible for all companies belonging to the AUTOSAR development partnership to develop AUTOSAR-conformant components. However, its practical implementation is not trivial. The transition from individual software to standard software has to be well-planned and is certainly only conceivable if a stepwise approach is taken. The AUTOSAR philosophy and description language create a diverse environment for using standardized software. In practice, this may be a mixed installation of AUTOSAR and non-AUTOSAR components or an integration of basic software for specific platforms by a number of different software suppliers. For both types of implementation, it is necessary to clarify and define the relevant constraints from various perspectives.

### Vehicle perspective

From the OEM perspective, basic platforms, technological platforms, etc. are being developed, from which the next generations of cars will be derived. The underlying goal here is to integrate one and the same ECU in many vehicles and thereby reduce costs. At the same time, the quality and the stability of the vehicle electronics should be improved. This results in a dilemma between the imperatives of a newly introduced technology and the stability of the product.

### Architectural perspective

From the point of view of an ECU individual software components are discernible. At the same time, two contradictory approaches are being followed with regard to the base software of today's ECUs. On the one hand, many OEMs require specific software components or at least they specify them. On the other hand, control module producers are striving internally to always use the same architecture for a control module platform. Added to this is the fact that the degree of standardization of the software is not as comprehensive as



described by AUTOSAR. The goal here is to apply a standard to software that does not serve the purpose of competitive differentiation, thereby creating space for new innovations. Optimally, low investment costs would be incurred by new tools, since those tools already in service can for the most part be reused.

### Clear migration strategy as factor in success

When these two perspectives are applied to a decision on introducing AUTOSAR, it makes sense to select a multistage approach.

#### Stage I – Set up the architecture and expand:

The first step is to compare the existing custom software and the AUTOSAR architecture. After analyzing overlaps and integration potentials, a decision is made regarding which modules will be preserved and which ones can be replaced by standard software. At this stage it is recommended that a separating layer be introduced between application software and base software as well as a standardized interface. The so-called Runtime Environment (RTE) serves as the link for the necessary data exchange, and as a buffer with defined interface it enables modular programming without dependencies. This is how AUTOSAR components can be integrated without making additional changes to the custom and application software. The custom software is linked to the system architecture via an Adaptation Layer to enable data exchange with the AUTOSAR components via the RTE; see Figure 1.

To minimize cost and effort and arrive at an optimal overall solution, it is helpful at this point to integrate the custom software in the configuration tools.

#### Stage II – Replace:

Non-AUTOSAR components can now be replaced gradually by AUTOSAR components without putting the overall architecture at risk or requiring reprogramming of other modules. The goal here is to set up an AUTOSAR architecture and use the appropriate tools. Initiated in individual ECUs, in the end the entire vehicle is conceptualized with AUTOSAR software, starting with system design and ending with integration.

#### Using the AUTOSAR architecture in designing new ECUs

As implied in Figure 2, essential parts of the individual software can also be reused in the framework of an AUTOSAR architecture. They are then linked to the application via an adaptation layer as a complex device driver.

An overlap matrix shows the portions in which AUTOSAR software can be introduced without great risk. Primarily, the memory section and the IO hardware abstraction can be migrated smoothly. Cluster memory management in particular has very clear and easy-to-use interfaces that enable its early migration to new ECUs.

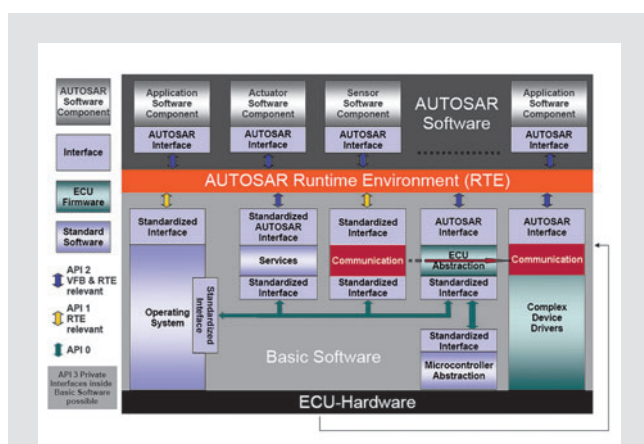
In communication and diagnostics, on the other hand, there are considerable overlaps between proprietary vehicle software and the standard modules of the AUTOSAR basic software. In the interest of stability in the vehicle, a more conservative approach is required here. Many OEMs utilize platform ECUs, in which existing software modules are transferred to new vehicle models. One implication is that the network and communications strategy cannot be changed over the short term. In the case of an immediate migration, ECU calibration and off-board diagnostics would also need to be adapted, which in practice would lead to significant problems.

Therefore, the simplest path is to use the existing communication stack in the AUTOSAR environment too. This stack can be linked to the RTE via an adaptation layer.

Vector has the necessary expertise in this area and can propose the right solutions for creating such mixed architectures or supply them to the customer. For example, the familiar XCP protocol can be integrated in a migration architecture so that existing measurement and calibration tools such as CANape can be used.

The described approach is not a pure top-down approach, since at many points AUTOSAR software can even be integrated on lower hardware-related layers. Its modular structure and defined interfaces help in implementing the standard software on the SPAL level without affecting the upper layers. This offers an enormous advantage with regard to reuse.

Vector Informatik utilizes the concept of Product Clustering here. Based on AUTOSAR specifications, the products offered range over a number of layers and provide total solutions for memory, commu-



**Figure 2:**  
Integration of custom software in the AUTOSAR architecture.

nication, diagnostic and system areas. These are independently functioning areas, some of which can also be implemented without AUTOSAR architecture. Cluster memory for example can be integrated quickly and easily in many ECU applications; see Figure 3.

### Support by tools

An important pillar in the introduction of AUTOSAR relates to the tools. They must be able to operate the AUTOSAR interfaces, yet they must remain open to integration of third-party components. Above all, configuration tools should be able to master this challenge and also support the user in validation of the system configuration.

The tool world servicing AUTOSAR can be subdivided into three categories: Design, configuration and test/simulation. Above all, suitable test instruments are an essential component for successful development. An ECU operates as a part of a whole. Checking for and assuring consistency in the overall system requires a high-performance simulation tool. Vector Informatik GmbH has come to grips with these requirements and can make a contribution to the success of AUTOSAR with its comprehensive tool solutions such as the DaVinci Tool Suite, MICROSAR Configuration Suite and CANoe. Support in the context of project work and consultation complement the products offered.

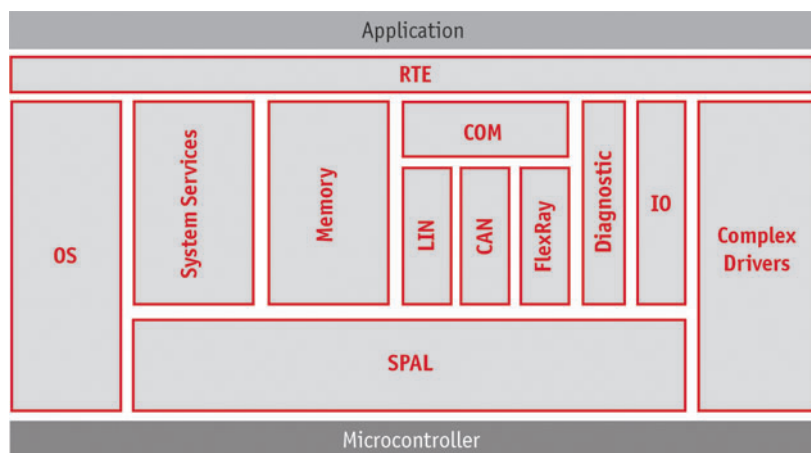
### Summary

When considered from different points of view, a stepwise introduction of the standard components defined by the AUTOSAR development partnership into an individual company's software architecture appears to be the correct path. This approach guarantees quality and consistency. Proper tools support this process. Gradual migration instead of an immediate total conversion leads to an overall AUTOSAR solution in the vehicle that minimizes risks. Vector's expertise and many years of experience lend support to this process.



#### Peter Schiekhofer

(Graduate Engineer) has been employed at Vector since May 2006. He manages Vector's Regensburg subsidiary and is responsible for advanced technical development of AUTOSAR solutions. As an active working participant on Working Packages WP 1.1.2 and WP 20 of the AUTOSAR Development Partnership he is directly involved with the new technology.

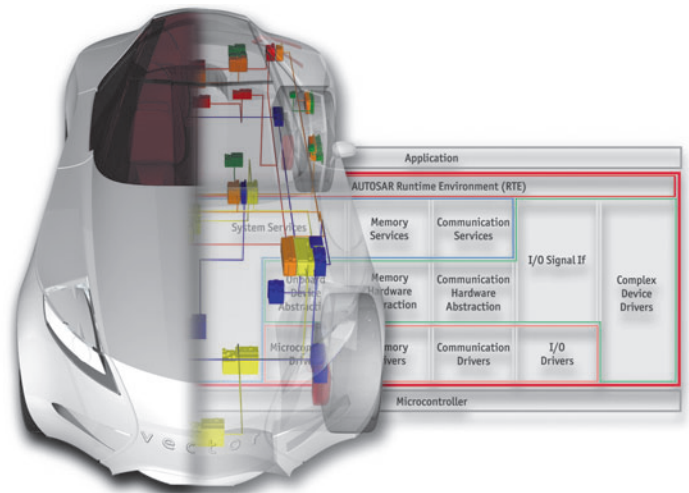


**Figure 3:** Vector offers MICROSAR, which contains the entire range of AUTOSAR-BSW including RTE.

# Successful Integration of Existing ECU Software in the AUTO

## Early migration creates opportunities for innovations

“Cooperate on standards – compete on implementation”, that is how the guiding principle of the AUTOSAR standardization initiative reads. A long harbored 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 centered 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 nonfunctional 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 centered 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 hardware 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 also 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, see Figure 1. The

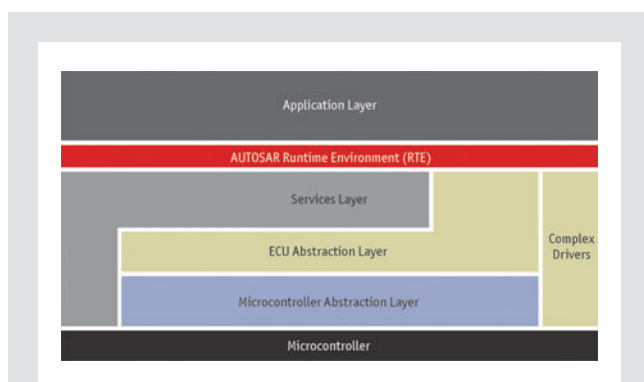


Figure 1: AUTOSAR layer model of ECU-specific software.

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 consistent 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 eASEE 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 behavior (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 support the generation of a comprehensive set of BSW modules such as the operating system, communication stacks and I/O drivers, Figure 3.

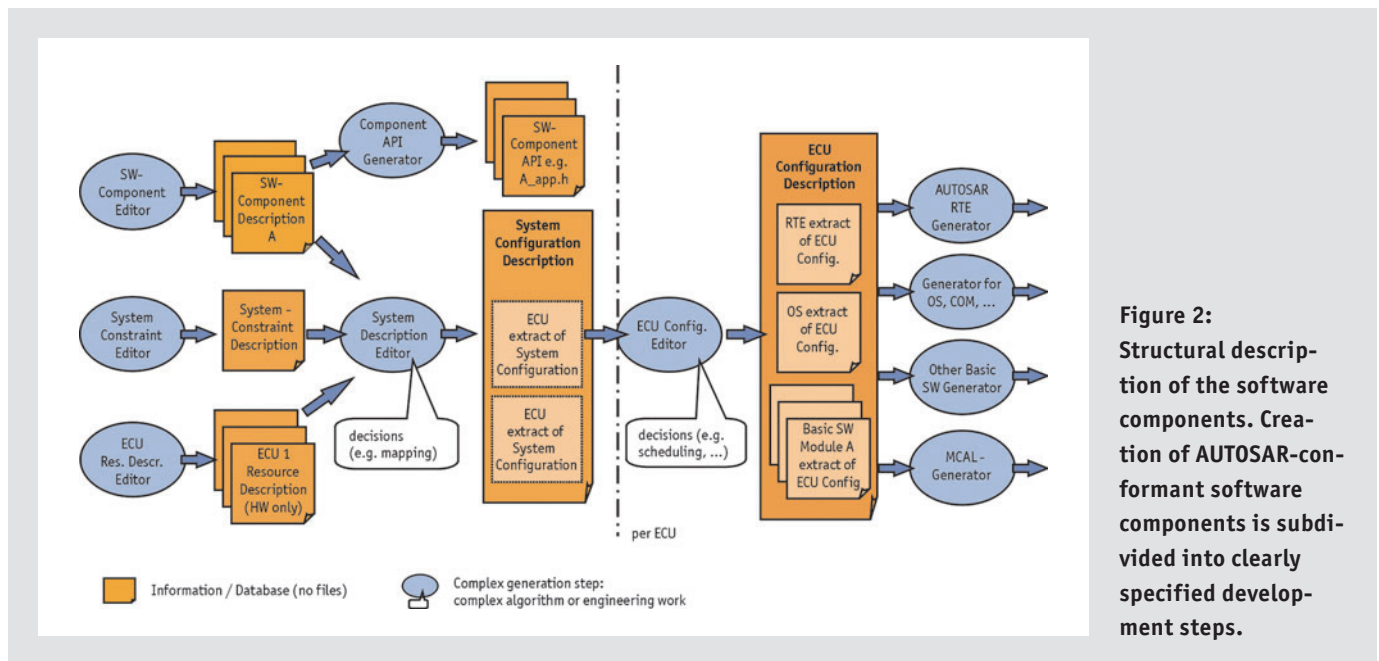
Tools support both implementation of software components and initial testing. When suitable model-based tools are used, functionality can be formulated using state diagrams for example. This enables later simulation of one or more software components and testing already conducted on the functional model. As an alternative, direct implementation of the software components in C is also pos-

sible. 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.

#### 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. Step-wise 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 DaimlerChrysler [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



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

continually redeveloped. Therefore at DaimlerChrysler 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.

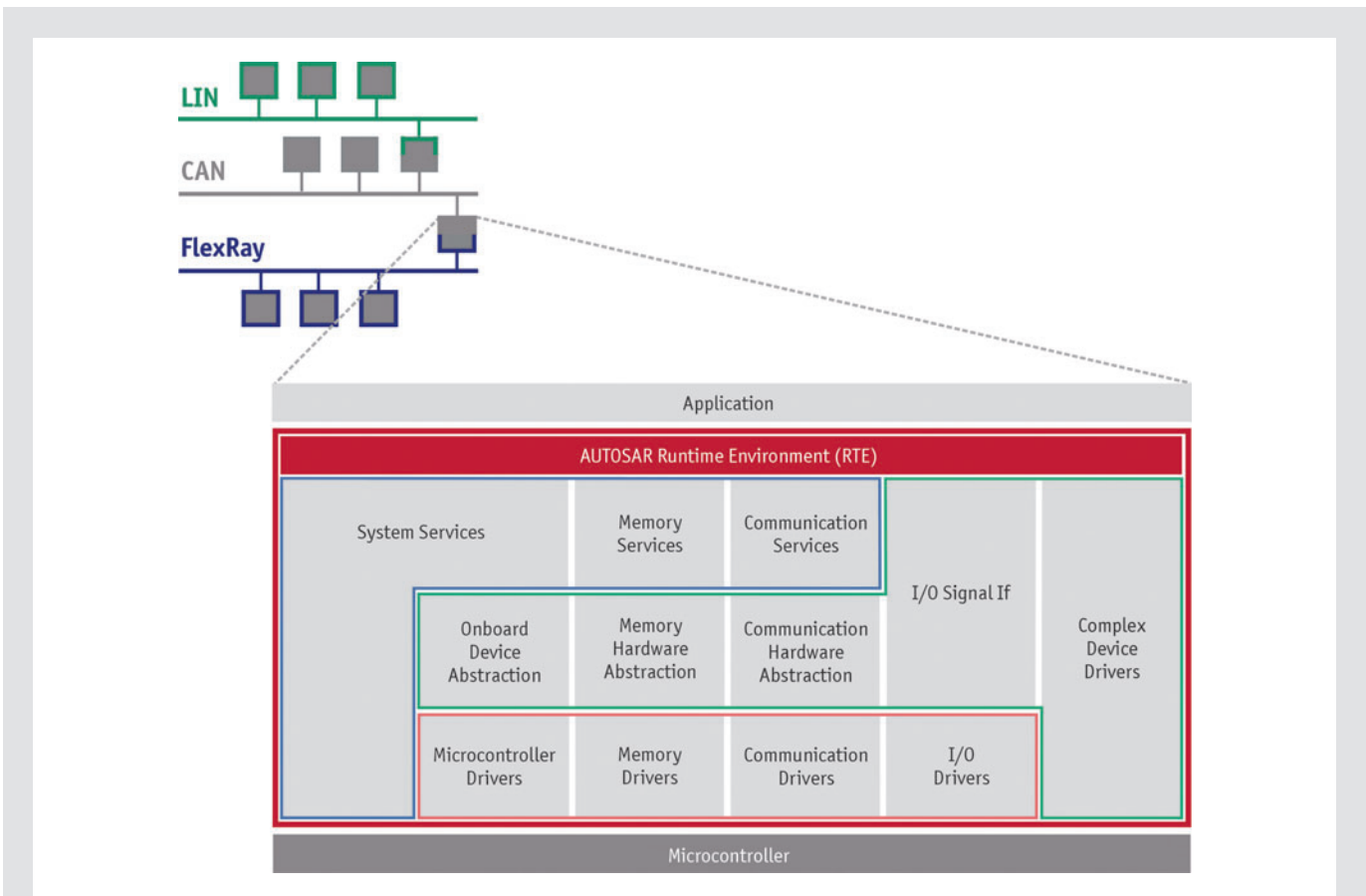


Figure 3: The Vector AUTOSAR solution: From system description to the standardized ECU software.

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, see 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, see Figure 6.

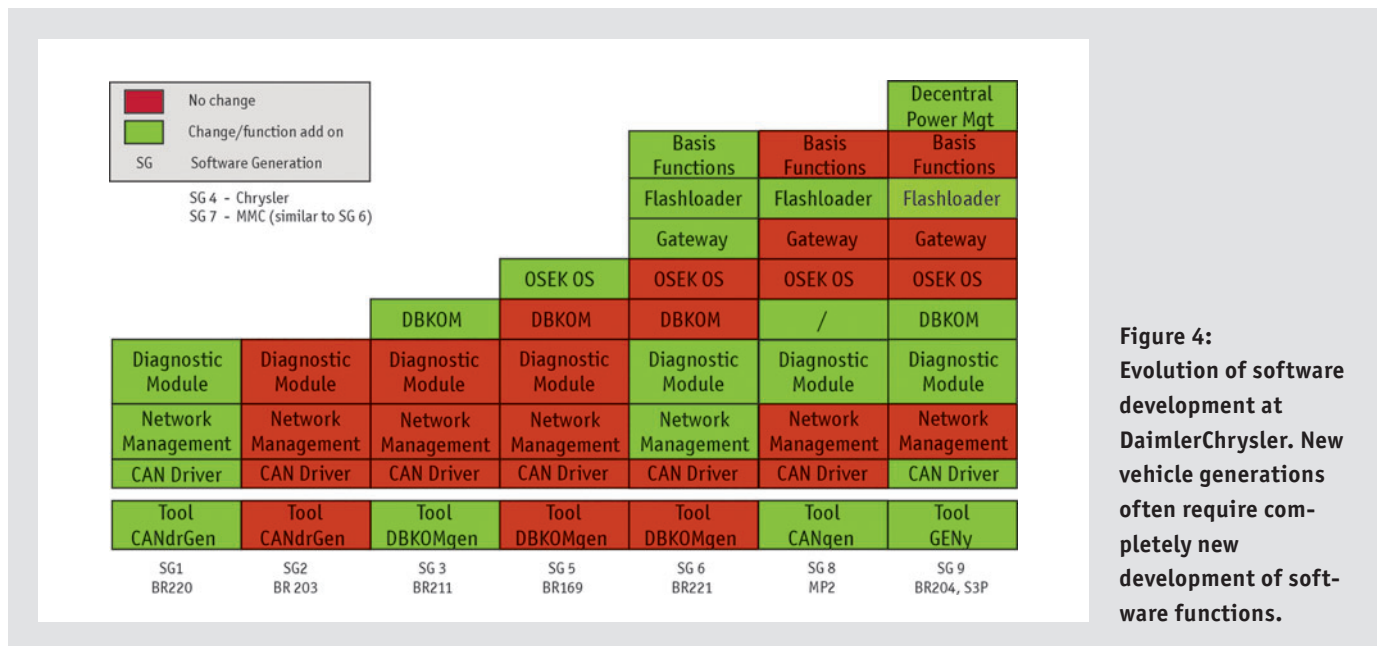


Figure 4: Evolution of software development at DaimlerChrysler. New vehicle generations often require completely new development of software functions.

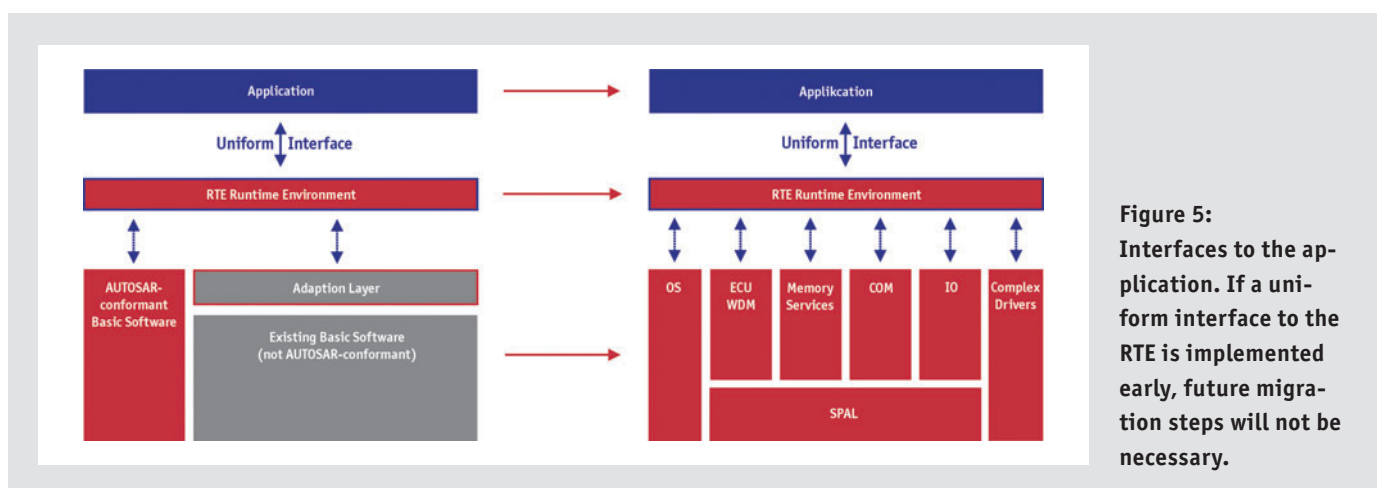


Figure 5: Interfaces to the application. If a uniform interface to the RTE is implemented early, future migration steps will not be necessary.

Simple test scenarios can be created quickly. 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.

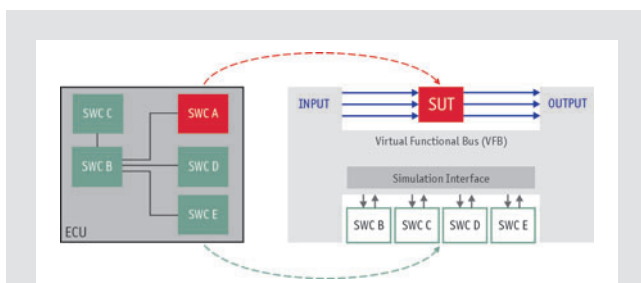
Possible supplements to such an IDE include test case generators and static code or coverage analyses. The better the linkage of such add-ons to the development platform, the simpler and more economical are the tests, and the product will operate more reliably later.

## 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 re-used. 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 soft-



**Figure 6:**  
Test of a software component in the execution and simulation platform. While a real executable software function is tested, additional functions can simply be added as models via the simulation interface.

ware. 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.

### Literature:

- [1] Beck, T.: Before considering tools it is essential to have a handle on the processes first, ATZ Elektronik 02/2006, pp. 32f.
- [2] Vector AUTOSAR Conference July 2006, Weber, T. "The value added by AUTOSAR from the OEM perspective"



#### Matthias Wernicke

(Graduate engineer) is responsible for product management of the DaVinci Tool Suite and is actively involved in AUTOSAR standardization work.



#### Jochen Rein

(Graduate engineer) is team leader in the "Embedded Software Components" department and is responsible for the AUTOSAR BSW and CANbedded Product Management area.

# Tool-Supported Integration of Microcontroller-Specific Modules

The development of ECU-specific software is taking a new path. The AUTOSAR development partnership has defined a software architecture resulting in standardized base software to be used as a foundation for development of reusable applications. This places special requirements on semiconductor producers.

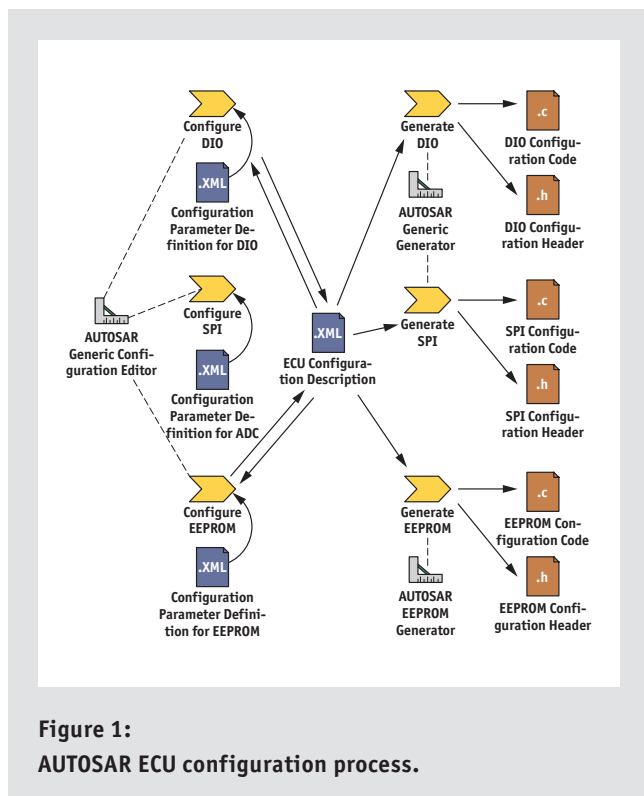


Figure 1:  
AUTOSAR ECU configuration process.

Now that AUTOSAR specification Version 2.1 has been released just recently, the first semiconductor producers are offering implementations of AUTOSAR SPAL (Standard Peripheral Abstraction Layer) modules for their microcontrollers. The AUTOSAR specification defines the behavior of any given module in the AUTOSAR architecture, regardless of whether it is a hardware-dependent module from the SPAL layer or a hardware-independent module from the Service Layer. Reusability and interchangeability of individual modules must be assured. Implementations of SPAL modules and hardware-independent modules only differ in the cost and effort involved. Hardware-independent modules are usually only implemented once, while SPAL modules must be re-designed for each microcontroller.

This means that semiconductor producers are treading in unfamiliar terrain, because – in addition to implementation of the SPAL modules – the AUTOSAR specification requires that a suitable configuration tool have to be provided. The tool should support these AUTOSAR-specified interfaces: The “AUTOSAR Parameter Definition” and the “AUTOSAR ECU Configuration Description” (Figure 1).

## Configuration tool for AUTOSAR architecture

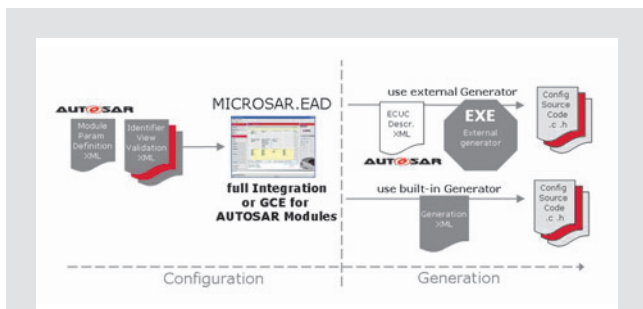
The AUTOSAR specification leaves the tool question open: All components can be configured either with a generic tool or a number of different tools. Neither alternative offers a comprehensive solution for guaranteeing a correct, consistent and yet easy-to-manage configuration process.

A tool chain with products from various producers does not guarantee error-free configuration, since the tools are not perfectly tuned to one another. The concept of utilizing a generic configuration tool that can read in the AUTOSAR Parameter Definition File for any AUTOSAR module, and can therefore master the entire ECU configuration, is in principle desirable. But the whole system will only work if parameters are queried throughout the process and are found to be consistent, so that optimal and valid results are obtained.

## Valid and open approach to configuration

A total solution for configuration, which goes beyond a purely generic approach and contains the necessary validity check, is offered by the tool MICROSAR EAD (Embedded Architecture Designer) from Vector Informatik. First, the tool performs the role of a Generic Configuration Editor and represents a module in an AUTOSAR Standard GUI based on the AUTOSAR Parameter Definition File. This functionality might indeed be sufficient for some less complex modules. For more difficult configuration concepts the XML Interface Description can be accessed, which enables a very detailed description of the configuration parameters and contains an innovative validation concept (Figure 2).

MICROSAR EAD offers an optimal foundation for integrating external components by implementing XML technology. Each integrated module is described by a set of XML files. External components may include: AUTOSAR SPAL modules from semiconductor producers,



**Figure 2:**  
**MICROSAR EAD – Integration concept.**

complex customized drivers or existing solutions that are not yet AUTOSAR-conformant. Vector offers integration of external components as a service. Vector acquired the necessary expertise from development experience with all types of AUTOSAR components and associated tools for configuring the base software and the overall system.

## SPAL modules require hardware expertise

Since each SPAL module contains the specific properties of the microcontroller, it is especially important here to provide a configuration tool with hardware know-how. To optimally assist the user in the startup of a new microcontroller, as early as configuration time Vector gives the user a lot of decision-making assistance. Defining parameter values in MICROSAR.EAD prevents invalid data input and invalid data import. Creation of one intelligent GUI per module enables a simplified, smooth configuration process for the user. Optimal data consistency is assured by a three-stage validation concept. This customizes the validation process, and qualification of the data is guaranteed before the files are generated.

After successful configuration and validation the generation process is started, in which files are generated according to the settings. It does not matter whether the generator used is the built-in generator of MICROSAR.EAD, which is also controlled by XML file, or an external generator is called. In this connection it is really important that a generation process may only occur after successful validation.

## Integration solutions proven in practice

Technical implementation and integration of the configuration have been established and undergone trials at Vector. With regard to the business model, certain questions arise: What form might integration of the SPAL modules of a semiconductor producer take in the MICROSAR EAD? Who assumes responsibility for distribution, maintenance and support? Many different scenarios are conceivable here. Vector Informatik is currently conducting discussions on individual solutions with various OEMs and semiconductor producers.

The advantages of an integration solution are obvious: Microcontrollers and SPAL modules from a single source shorten development times and increase consistency. The MICROSAR EAD configuration tool with integrated SPAL modules supports the user optimally with easy handling and openness in integration. The extensive validation process detects configuration errors early on and improves quality. The capability of integrating additional modules and even external components in MICROSAR.EAD makes it

unnecessary to seek out a coherent tool chain. That is because configuration of an entire architecture can be guaranteed with just one tool. In porting existing projects to new controllers or derivatives the data transfer runs smoothly.

There remains the question of the practical feasibility of the modularity prescribed by the AUTOSAR architecture. The well-coordinated Vector product line, and its interconnection to external products, shows that this modularity can operate properly. Integration of the SPAL modules from semiconductor producers targets another AUTOSAR commandment: Interchangeability. Vector Informatik is up to the challenge here. With a full product lineup of AUTOSAR software architecture, Vector offers support in all project development phases: From design to implementation to integration. In this process, it does not matter whether what is being used are Vector products, in-house developments or third-party components.



**Justus Maier**

(Graduate engineer) is team leader in the "Embedded Software" department and is responsible for the further development of Vector's AUTOSAR tooling solution.



# SOLUTIONS FOR AUTOSAR

## Start your Series Development with AUTOSAR

Enjoy the benefits of field-tested modules that can be used right away:

- > Efficiency through reusability and time savings
- > Quality through tried-and tested use in serial projects
- > Openness through the option of optimally integrating third-party components
- > Flexibility to convert to AUTOSAR step-by-step
- > Focus on application development

**We create your AUTOSAR solution using expertise and commitment. Base software and high-performance tools — integrated in their own software architecture.**

▶ For more information, please visit: [www.autosar-solutions.com](http://www.autosar-solutions.com)

**AUTOSAR**

# Your Contact

## Germany and all countries not named below

Vector Informatik GmbH  
Ingersheimer Str. 24  
70499 Stuttgart  
GERMANY  
Tel.: +49 711 80670 0  
Fax: +49 711 80670 111

## France, Belgium, Luxemburg

Vector France SAS  
168, Boulevard Camélinat  
92240 Malakoff  
FRANCE  
Tel.: +33 1 4231 4000  
Fax: +33 1 4231 4009

## Sweden, Denmark, Norway, Finland, Iceland

VecScan AB  
Lindholmspiren 5  
41756 Göteborg  
SWEDEN  
Tel.: +46 31 764 76 00  
Fax: +46 31 764 76 19

## USA, Canada, Mexico

Vector CANtech, Inc.  
Suite 550  
39500 Orchard Hill Place  
Novi, Mi 48375  
USA  
Tel.: +1 248 449 9290  
Fax: +1 248 449 9704

## Japan

Vector Japan Co., Ltd.  
Seafort Square Center Bld. 18F  
2-3-12 Higashi-shinagawa,  
Shinagawa-ku  
Tokyo 140-0002  
JAPAN  
Tel.: +81 3 5769 6970  
Fax: +81 3 5769 6975

## Korea

Vector Korea IT Inc.  
Daerung Post Tower III, 508  
Guro-gu, Guro-dong 182-4  
Seoul 152-790  
REPUBLIC OF KOREA  
Tel.: +82 2 2028 0600  
Fax: +82 2 2028 0604

[www.vector-worldwide.com](http://www.vector-worldwide.com)

