AUTOSAR –
An open standardized software architecture for the automotive industry

Simon Fürst, BMW
1st AUTOSAR Open Conference &
8th AUTOSAR Premium Member Conference
October 23rd, 2008, Cobo Center, Detroit, MI, USA
Document Information and Change History

<table>
<thead>
<tr>
<th>Document Owner</th>
<th>Heiko Dörr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Responsibility</td>
<td>SC</td>
</tr>
<tr>
<td>Document Version</td>
<td>1.5</td>
</tr>
<tr>
<td>Document filename</td>
<td>AUTOSAR_Tutorial</td>
</tr>
<tr>
<td>Document Status</td>
<td>&lt;Draft</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30.05.2007</td>
<td>0.1</td>
<td>Heiko Dörr</td>
<td>Initial creation and proposal for content</td>
</tr>
<tr>
<td>10.08.2007</td>
<td>0.2</td>
<td>Heiko Dörr</td>
<td>Ready for approval</td>
</tr>
<tr>
<td>16.08.2007</td>
<td>0.3</td>
<td>Heiko Dörr</td>
<td>Comments by Mr. Bunzel entered</td>
</tr>
<tr>
<td>17.08.2007</td>
<td>0.4</td>
<td>Heiko Dörr</td>
<td>Section on Methodology updated after discussion with member of WPII-1.2; Section on Exploitation removed; Ready for approval by SC</td>
</tr>
<tr>
<td>23.08.07</td>
<td>1.0</td>
<td>Heiko Dörr</td>
<td>Tutorial approved by SC</td>
</tr>
<tr>
<td>31.08.07</td>
<td>1.1</td>
<td>Heiko Dörr</td>
<td>PM memberships updated according to slide provided by admin</td>
</tr>
<tr>
<td>07.04.08</td>
<td>1.2</td>
<td>Heiko Dörr</td>
<td>CP updated, readability improved</td>
</tr>
<tr>
<td>07.05.08</td>
<td>1.3</td>
<td>Heiko Dörr</td>
<td>Bus state managers added to BSW modules slide</td>
</tr>
<tr>
<td>03.07.2008</td>
<td>1.4</td>
<td>Heiko Dörr</td>
<td>Selected slides from ATI added (Animated use case for distributed scenario, Validator 2), Schedule updated</td>
</tr>
<tr>
<td>01.10.2008</td>
<td>1.5</td>
<td>Heiko Dörr</td>
<td>Selection of slides for presentation at 8th PM conference</td>
</tr>
</tbody>
</table>
## Document Information and Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>22.10.2008</td>
<td>1.51</td>
<td>Simon Fürst</td>
<td>Review</td>
</tr>
</tbody>
</table>
Automotive Systems and SW Engineering
Automotive Open System Architecture
Cooperate on standards – compete on implementation

Non functional legal requirements
Vehicle family management
Resource efficiency

Driver assistance
Driving dynamics
Safety functions (active/passive)
Comfort functions
AUTOSAR Managing Complexity
by Exchangeability and Reuse of Software Components

Exchangeability between supplier’s solutions

Exchangeability between manufacturer’s applications

Exchangeability between vehicle platforms

Supplier A
- Chassis
- Safety
- Body/Comfort
- Multimedia

Supplier B
- Chassis
- Safety
- Telematics
- Multimedia

Supplier C
- Body/Comfort
- Powertrain
- Telematics
- Multimedia

OEM a
- Platform a.1
- Platform a.2
- Platform a.n

OEM b
- Platform b.1
- Platform b.2
- Platform b.n

OEM c
- Platform c.1
- Platform c.2
- Platform c.n

OEM d
- Platform d.1
- Platform d.2
- Platform d.n

OEM e
- Platform e.1
- Platform e.2
- Platform e.n

OEM f
- Platform f.1
- Platform f.2
- Platform f.n
AUTOSAR
Project Objectives and Main Working Topics

- Implementation and standardization of basic system functions as an OEM wide “Standard Core” solution
- Scalability to different vehicle and platform variants
- Transferability of functions throughout network
- Integration of functional modules from multiple suppliers
- Maintainability throughout the whole “Product Life Cycle”
- Increased use of “Commercial off the shelf hardware”
- Software updates and upgrades over vehicle lifetime
- Consideration of availability and safety requirements
- Redundancy activation
AUTOSAR
Main Working Topics

- **Architecture:**
  Software architecture including a complete basic or environmental software stack for ECUs – the so called AUTOSAR Basic Software – as an integration platform for hardware independent software applications.

- **Methodology:**
  Exchange formats or description templates to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs and it includes even the methodology how to use this framework.

- **Application Interfaces:**
  Specification of interfaces of typical automotive applications from all domains in terms of syntax and semantics, which should serve as a standard for application software.
Technical scope of AUTOSAR

New concepts

Industry-wide consolidation of 'existing' basic software designs
Benefits from AUTOSAR

- OEM overlapping reuse of software modules
- Maintaining ability to compete on innovative functions, enlarged design flexibility
- Simplification of the integration task
- Reduction of total SW development costs

- Reduction of version proliferation
- Development partitioning among suppliers
- Increase of efficiency in functional development
- New business models possible

- Common interfaces with development processes
- Seamless, manageable, task optimized (time dependent) tool landscape

- Transparent and defined interfaces enable new business models
Project Setup Phase II
AUTOSAR – Partnership Structure

**Core Partners (OEM & Tier 1 Supplier)**
- Organizational control
- Technical contributions
- Administrative control
- Definition of external Information (web-release, clearance, etc.)
- Leadership of Working Groups
- Involvement in Working Groups

**Premium Members**
- Leadership of Working Groups
- Involvement in Working Groups
- Technical contributions
- Access to current information

**Associate Members**
- Access to finalized documents
- Utilization of AUTOSAR standard

**Support Roles**
- Development Members
- Attendees
AUTOSAR – Core Partners and Members

Status: 10th October 2008

9 Core Partner

55 Premium Member

6 Development Member

85 Associate Member

General

OEM

Generic

Tier 1

Standard

Software

Tools and

Services

Semi-

conductors

Up-to-date status see: http://www.autosar.org
AUTOSAR Phase II 2007 – 2009

- AUTOSAR Development Partnership continues
- Identical Core Partners
- Exploitation and maintenance
  - Already in 2008 the first cars on the road with AUTOSAR technology inside
  - All Core Partners have planned the introduction of AUTOSAR products until 2010
  - Establish conformance test specifications and process
- Further development and amendment of the standard, e.g.
  - Functional safety features
  - Support for multi core microcontrollers
  - Vehicle & application mode management
  - Debugging and error handling
  - Variant handling
  - Timing model
  - Standardization of application interfaces
Top Level Schedule for AUTOSAR in phase II

Phase II (2007 – 2009)

Basic Software & RTE

- Specification R3.0
  - Finaliz.
  - Maintain R3.0 Specifications
- Specification R3.1
  - Finaliz.
  - Maintain R3.1 Specifications
- Concepts R4.0
- Specification R4.0
  - Maint. R4.0
- CT Pilot
- Conformance Test Preparation
- CT Spec 1st set
- CT Spec 2nd set
- CT 3rd set
- CT 4th set
- Maintain CTS

Methodology and Templates

- Methodology & Templates R3.0
  - Finaliz.
  - Maintain R4.0
- Methodology & Templates R4.0
  - Maint. R4.0

Application Interfaces

- Specification Appl. Interfaces R3.0
  - Finaliz.
- Specification Application Interfaces R4.0
  - Maint. R4.0

- Specification 3.0 Ready
- Release 3.0
- Specification 3.1 Ready
- Release 3.1
- Spec R4.0 MS3 Ready
- Validation & CT 4.0
- Release 4.0

- 28.09.07
- 21.12.07
- 30.05.08
- 27.06.08
- 12.12.08
- 24.07.09
- 27.11.09


2007 2008 2009
AUTOSAR Phase II
Work Package Breakdown Structure

**II-1**
System Architecture
- II-1.1 Software Architecture
  - WPII-1.1.1 Software Architecture and OS
  - WPII-1.1.2 Vehicle and Application Mode Mgmt.
  - WPII-1.1.3 Debugging
  - WPII-1.1.4 Error Handling
  - WPII-1.1.5 VFB and RTE
- II-1.2 Methodology and Configuration
- II-1.3 Functional Safety

**II-2**
Software and Test Specification
- II-2.1 Basic Software
  - WPII-2.1.1 COM Stack
  - WPII-2.1.2 FlexRay
  - WPII-2.1.3 MCAL
  - WPII-2.1.4 Diagnostics
  - WPII-2.1.5 Libraries
- II-2.2 Conformance Test Specification

**II-3**
Validation
- WPII-3.1 Basic Software Validation
- WPII-3.2 Methodology Validation

**II-4**
Enabling Exploitation
- WPII-4.1 Basic Software Validation
- WPII-4.2 Communication and Marketing
- WPII-4.3 Follow-up Organization

**II-5**
Maintenance of Releases
- WPII-5.1 Problem Management
- WPII-5.2 Change and Release Mgmt.
- WPII-5.3 Maintenance of Specifications

**II-10**
Application Interfaces
- WPII-10.0 Coordination of Appl. Interfaces
- WPII-10.1 Body and Comfort
- WPII-10.2 Powertrain
- WPII-10.3 Chassis Control
- WPII-10.4 Occupants and Pedest. Safety
- WPII-10.5 MM / T / HMI

**II-10.0**
Coordination of Application Interfaces

**II-10.1**
Body and Comfort

**II-10.2**
Powertrain

**II-10.3**
Chassis Control

**II-10.4**
Occupants and Pedestrian Safety

**II-10.5**
MM / T / HMI
Use Cases of AUTOSAR Results

- Exchange of SW-Components
- Re-use of SW components for different platforms

... shown by uses cases
- pedal management
- front light management
Use Case ‘Pedal Management’ view for one ECU

- Implementation of functions independent on distribution on different ECU as communication will be done via ECU-individual AUTOSAR-RTE exclusively

```c
void distribute_v(void)
{
    ...
    Rte_Write_p_v(rte_i, v)
    ...
}

void v_warn(void)
{
    ...
    Rte_Read_p_v(rte_i, v)
    ...
}
```
Use Case ‘Pedal Management’ view for two ECUs

- Reuse of Intellectual Property
- Increase in design flexibility
- Simplification of the integration task
- Reduction of SW development costs
Use case ‘Front-Light Management’ in AUTOSAR

- **SwitchEvent**
  - check_switch()
  - switch_event(event)
  - AUTOSAR Int.

- **LightRequest**
  - switch_event(event)
  - request_light(type, mode)
  - AUTOSAR Interface

- **Front-Light Manager**
  - request_light(type, mode)
  - get_keyposition()
  - set_light(type, mode)
  - AUTOSAR Interface

- **Headlight**
  - set_light(type, mode)
  - AUTOSAR Interface

**Standardized Interface**

**Operating System**

**Services**

**CAN Driver**

**PWM**

**DIO**

**Microcontroller Abstraction**

**ECU-Hardware**

**Complex Device Drivers**
Exchange of type of front-light

SwitchEvent
- check_switch
  switch_event(event)
AUTOSAR Interface

LightRequest
- switch_event(event)
- request_light(type, mode)
AUTOSAR Interface

Front-Light Manager
- request_light(type, mode)
- get_keyposition()
- set_light(type, mode)
AUTOSAR Interface

Xenonlight
- set_light(type, mode)
- set_current(…)
AUTOSAR Interface

Operating System
- Standardized Interface

Services
- Std. AUTOSAR Interface

Communication
- Std. Interface

ECU Abstraction
- Std. Interface

Complex Device Drivers
- Std. Interface

DIO
- Std. Interface

CAN Driver
- Std. Interface

Microcontroller Abstraction

ECU-Hardware
Distribution on ECUs

SwitchEvent
switch_event(event)

LightRequest
switch_event(event)
request_light(type, mode)

Front-Light Manager
request_light(type, mode)
set_light(type, mode)

Xenonlight
set_light(type, mode)
set_current(…)

AUTOSAR Interface

AUTOSAR RTE

Std. AUTOSAR Interface

Services

ECU-Hardware

STD. AUTOSAR Interface

Communication

Operating System

DIO

PWM

Xenonlight

set_current(…)

set_light(type, mode)

Front-Light Manager
get_keyposition()
set_light(type, mode)

request_light(type, mode)

request_light(type, mode)

request_light(type, mode)
Use case ‘Front-Light Management’ in AUTOSAR
AUTOSAR Key Topics

AUTOSAR provides three main areas of results:

- **Architecture:**
  Software architecture including a complete basic (environmental) software stack for an ECU as an integration platform for hardware independent SW applications

- **Methodology:**
  Exchange formats (templates) to enable a seamless configuration process of the basic software stack and the integration of application software in ECUs

- **Application Interfaces:**
  Specification of application interfaces as a standard for application software modules
Main Concepts: Architecture

- Basic Software modules
- Run time environment and communication
- Results of sample implementation in „Validator 2“
Standardized AUTOSAR interfaces will support HW independence and enable the standardization of SW components.

Automotive Open System Architecture (AUTOSAR):

- Standardized, openly disclosed interfaces
- HW independent SW layer
- Transferability of functions
- Redundancy activation

**AUTOSAR RTE:**
by specifying interfaces and their communication mechanisms, the applications are decoupled from the underlying HW and Basic SW, enabling the realization of Standard Library Functions.
AUTOSAR Tutorial

AUTOSAR Basic Software

AUTOSAR Runtime Environment (RTE)

- System Services
- Memory Services
- Communication Services
- I/O Hardware Abstraction
- Complex Drivers

- Onboard Device Abstraction
- Memory Hardware Abstraction
- Communication Hardware Abstraction

- Microcontroller Drivers
- Memory Drivers
- Communication Drivers
- I/O Drivers

Microcontroller

Application Layer
Example: “NVRAM Manager” ensures the storage and maintenance of non-volatile data and is independent of the design of the ECU.
Intra- and Inter-ECU Communication

- Ports implement the interface according to the communication paradigm (here client-server based).
- Ports are the interaction points of a component.
- The communication is channeled via the RTE.
- The communication layer in the basic software is encapsulated and not visible at the application layer.
Validation of AUTOSAR Release 2.0

AUTOSAR Specifications

AUTOSAR Concepts & Methodology

Validator 2 dealt with ...

Integration Resource & Consumption Measurement

Module Implementations & ECU Configuration Tools

Software Platforms

TriCore 1766 (32 Bit)
HCS12X (16 Bit)
Used Release 2.0 AUTOSAR specifications

AUTOSAR Tutorial Oct. 23rd 2008

All generic and module specific SW Specifications of all Software Layers of AUTOSAR BSW & RTE

AUTOSAR Methodology Specifications regarding ECU Configuration

Configure System

Extract ECU-Specific Information

Generate Executable
The specified application provides ‘realistic’ functionality:

- Calculating the vehicle speed based on several inputs
- Displaying the calculated speed
Scalability is divided into 3 aspects:
- Distribution of the given application on several nodes.
- Using the appropriate communication bus technology.
- Using the appropriate platform for each node.
Experience with AUTOSAR concepts and methodology: RTE

RTE concept was validated in the system test where a “dummy” real world application was created with a couple of AUTOSAR SW-C’s and IO Hardware Abstraction.

RTE overhead = low!

Lessons learned:

- Configuration of RTE might be very complex as long the requirements of the RTE and the OS are not optimized.
- Close linkage of RTE & OS requires close cooperation between implementers
Experience with AUTOSAR concepts and methodology:
IO Hardware Abstraction

The IO HW Abstraction is a special kind of AUTOSAR SW-C.
It enables the integration of SW-C which use IOs from the ECU (e.g. SW-C for sensors and actors).

- The IO HW abstraction SWS implemented in validator 2 was handled as a SW-C (AUTOSAR interface) and as a BSW module (interface to BSW Scheduler and IO driver).
- It was needed to specify the AUTOSAR interface of the IO HW Abstraction at the beginning of the project, since it is not defined in the SWS of the IO HW Abstraction.
- The definition of the AUTOSAR interface was done by defining ports for IO HW Abstraction and SW-Cs.
- Port interfaces instead of ports should be defined first.
AUTOSAR Architecture – Conclusion

1. AUTOSAR harmonizes already existing basic software solutions and closes gaps for a seamless basic software architecture.

2. AUTOSAR aims at finding the best solution for each requirement and not finding the highest common multiple.

3. The decomposition of the AUTOSAR layered architecture into some 50 modules has proven to be functional and complete.

4. The AUTOSAR 2.0 specifications for the modules of the layered architecture have been successfully implemented and integrated.

5. Conformance tests and processes are being prepared to ensure and to maintain a stable standard.
Main Concepts: Methodology

- Overall methodology
- Structure of configuration information
- System Design – Implementation Process
- Meta-model structure
Following the AUTOSAR Methodology, the E/E architecture is derived from the formal description of software and hardware components.

- Functional software is described formally in terms of “Software Components” (SW-C).
- Using “Software Component Descriptions” as input, the “Virtual Functional Bus” validates the interaction of all components and interfaces before software implementation.
- Mapping of “Software Components” to ECUs and configuration of basic software.
- The AUTOSAR Methodology supports the generation of an E/E architecture.
AUTOSAR Methodology

Derive E/E architecture from formal descriptions of soft- and hardware components

VFB view

- Standardized description templates for application software components (interfaces and BSW requirements)
- Standardized exchange formats and methodology for component, ECU, and system level
- Tools for
  - support of component mapping
  - generation of RTE, i.e. inter- and intra ECU communication
- Standardized Basic Software (BSW) architecture, detailed specifications for implementation and configuration of BSW
To configure the system, input descriptions of all software components, ECU resources and system constraints are necessary.
The system configuration maps software components to ECUs and links interface connections to bus signals.
AUTOSAR – System Design – Implementation Process

Input: Requirements & Vehicle Info

1a SW Component Description

1c System Description

1b ECU Resource Description

2 Configure System & generate extracts of ECU descriptions

3 Configure each ECU

4 Generate SW executables for each ECU

Iterative corrections and/or optimizations (if required)
AUTOSAR – The Virtual Functional Bus
Input to the System Design on an abstract level

- SW-Component-Description „get_v()“ describes a function to acquire the current vehicle speed and defines the necessary resources (such as memory, run-time and computing power requirements, etc.)
- Function „v_warn()“ makes use of „get_v()“
- „Virtual Integration“ by check of
  - completeness of SW-Component-Descriptions (entirety of interconnections)
  - integrity/correctness of interfaces
- The Virtual Functional Bus is implemented by the AUTOSAR/Runtime-Environment (RTE) and underlying Basic-SW
AUTOSAR – Input Descriptions (1 of 3)
Step 1a): Description of SW-Components independently of hardware

Information for each SWC e.g. “get_v()”
- interfaces, behavior (repetition rate, …)
- direct hardware interfaces (I/O)
- requirements on run-time performance (memory, computing power, throughput, timing/latency, …)
- …

SW Component Description

- General characteristics (name, manufacturer, etc.)
- Communication properties:
  - p_ports
  - r_ports
  - interfaces
- inner structure (composition)
  - sub-components
  - connections
- required HW resources:
  - processing time
  - scheduling
  - memory (size, type, etc.)

= tool based
AUTOSAR – Input Descriptions (2 of 3)
Step 1b): Description of hardware independently of application software

Information for each ECU
e.g. ECU1
- sensors and actuators
- hardware interfaces
- HW attributes (memory, processor, computing power, ...)
- connections and bandwidths, etc.
- ...

ECU Resource Description

- General characteristics (name, manufacturer, etc.)
- Temperature (own, environment, cooling/heating)
- Available signal processing methods
- Available programming capabilities
- Available HW: - μC, architecture (e.g. multiprocessor)
  - memory
  - interfaces (CAN, LIN, MOST, FlexRay)
  - periphery (sensor / actuator)
  - connectors (i.e. number of pins)
- SW below RTE for micro controller
- Signal path from Pin to ECU-abstraction

= tool based
AUTOSAR – Input Descriptions (3 of 3)
Step 1c): Description of system

System Information
overall system
- bus systems, protocols, communication matrix and attributes (e.g. data rates, timing, …)
- function clustering
- function deployment (distribution to ECU)
- …

System Description

- Network topology
  - bus systems: CAN, LIN, FlexRay
  - connected ECUs, Gateways
  - power supply, system activation

- Communication (for each channel)
  - K-matrix
  - gateway table

- Mapping / Clustering of SW components

AUTOSAR-Description Editor

= tool based
AUTOSAR – System Configuration

Step 2: Distribution of SW-Component-Descriptions to ECU

- Configuration on the basis of descriptions (not on the basis of implementations!) of SW-Components, ECU-Resources and System-Description
- Consideration of ECU-Resources available and constraints given in the System-Description
AUTOSAR – ECU-Configuration

Step 3: Generation of required configuration for AUTOSAR-Infrastructure per ECU

**AUTOSAR Configuration Generator**

- Configuration-Descript. ECU1
  - Description 1,
  - Description 2,
  - ...
  - Resources

- System Description
  - e.g. mapping of signals to CAN matrices
  - ...

**AUTOSAR-RTM-Config-Info**

- communication mechanisms
- transport protocols
- ...

**AUTOSAR-Configuration ECU1**

- configuration of the AUTOSAR-RTE
- configuration of AUTOSAR OS
- configuration of MCAL
- Configuration of COM stack
- etc

= tool based
AUTOSAR – Generation of Software Executables

Step 4: Based on the configuration information for each ECU (example ECU1)

- Application SW
  - Body of the SW components

- AUTOSAR-Library
  - communication
  - transport protocols, ...
  - (code, macros, Objects, ...)

- AUTOSAR-Configuration ECU1
  - configuration of the AUTOSAR-RTE
  - configuration of AUTOSAR OS
  - configuration of MCAL
  - Configuration of COM stack

- Tooling
  - AUTOSAR-RTE Generator
  - OS Generator
  - MCAL-Generator
  - COM Generator

- SW-Components ECU1
  - (derived partially from the Virtual Function Bus)
    - AUTOSAR-RTE
    - OS
    - MCAL
    - COM
    - Basic system functions core functions, drivers
    - Hardware
AUTOSAR Metamodel
Formal description of all methodology related information

- The metamodel is modeled in UML
- The structure of the information can be clearly visualized
- The consistency of the information is guaranteed
- Using XML, a data exchange format can be generated automatically out of the metamodel
The AUTOSAR Metamodel

- M0: Realized System in the car (Implements a real system)
- M1: Model of the system (Defines a real system)
- M2: Model of the model (Metamodel) (Defines AUTOSAR Modeling Elements)
- M3: Model of the Metamodel (Meta-Metamodel) (Defines UML Modeling Elements)

The AUTOSAR Metamodel is the backbone of the AUTOSAR architecture definition and contains complete specification, how to model AUTOSAR systems.
AUTOSAR Metamodel and Methodology

- **Methodology**
  - defines activities and work-products
  - is integrated in the metamodel

- **Metamodel defines content of work-products**
  - Formal description of all the information that is produced or consumed in the AUTOSAR methodology
  - Benefit of using the metamodel:
    - No inconsistencies
    - Easy maintenance
    - Consistent terminology
AUTOSAR Methodology – Conclusion

1. The E/E system architecture can be described by means of AUTOSAR.

2. The meta model approach and the tool support for specifying the AUTOSAR information model allow working at the right level of abstraction.

3. A methodology to integrate AUTOSAR software modules has been designed.

4. AUTOSAR pushes the paradigm shift from an ECU based approach to a function based approach in automotive software development.
Main Concepts: Application Interfaces

- Standardization approach
- Current stage of standardization
**AUTOSAR Application Interfaces**

**Syntax** of Interfaces:
- Meta-model, Software Component Template
- Supporting transferability within the network

**Semantics** of Interfaces:
- Physical properties, units, etc.
- Supporting re-use across product lines
- In scope of AUTOSAR workpackages specifying application interfaces
OEM Use case

➢ SHORT TERM: OEM is applying AUTOSAR Naming Convention more than 10,000 interfaces and calibrations data for industrial purposes after two years of intensive work on the specification of the naming convention

➢ Middle Term: Results are foreseen as an “AUTOSAR Application Interfaces Handbook” to support internal design & development of vehicle functions as much as support for exchange in project where suppliers are tied.

Use of standardized application interfaces **increase quality on exchange** with suppliers and **improve software integration** from system standpoint.
Supplier Use case

- Specification of application interfaces will support integration of SW-components.

Use of 10.x application interfaces increase **quality on integration**, i.e. they prevent from inconsistencies.
To ease the re-use of software components across several OEMs, AUTOSAR proceeds on the standardization of the application interfaces agreed among the partners.

---

**Example**

<table>
<thead>
<tr>
<th>Data Type Name</th>
<th>LongAccBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Yaw rate measured along vehicle z-axis (i.e. compensated for orientation). Coordinate system according to ISO 8855</td>
</tr>
<tr>
<td>Data Type</td>
<td>S16</td>
</tr>
<tr>
<td>Integer Range</td>
<td>-32768..+32767</td>
</tr>
<tr>
<td>Physical Range</td>
<td>-2,8595..+2,8594</td>
</tr>
<tr>
<td>Physical Offset</td>
<td>0</td>
</tr>
<tr>
<td>Unit</td>
<td>rad/sec</td>
</tr>
<tr>
<td>Remarks</td>
<td>This data element can also be used to instantiate a redundant sensor interface. Range might have to be extended for future applications (passive safety).</td>
</tr>
</tbody>
</table>

---

**Standardized application interfaces on system level (ESP-system, chassis domain)**
CmdWashing is the interface defined by following information:

- It is provided by the WiperWasherManager component through the [Washer]Activation port
- CmdWashing contains one data element command
- Command is of type t_onoff
- t_onoff is a RecordType, which describes a generic on/off information
### AUTOSAR Application Interfaces

**Compositions under Consideration**

#### Body Domain
- Central Locking
- Interior Light
- Mirror Adjustment
- Mirror Tinting
- Seat Adjustment
- Wiper/Washer
- Anti Theft Warning System
- Horn Control

#### Chassis Control Domain
- Vehicle Longitudinal Control
- Electronic Stability Program
- Electronic Parking Brake
- Adaptive Cruise Control
- Roll Stability Control
- Steering System
- Suspension System
- Stand Still Manager
- High Level Steering
  - Vehicle Stability Steering
  - Driver Assistance Steering
  - All Wheel Drive/ Differential Lock

#### Powertrain Domain
- Powertrain Coordinator
- Transmission System
- Combustion Engine
  - Engine torque and mode management
  - Engine Speed And Position
  - Combustion Engine Misc.
- Electric Machine
- Vehicle Motion Powertrain
  - Driver Request
  - Accelerator Pedal Position
  - Safety Vehicle Speed Limitation
Major task: Conflict Resolution - Example Vehicle Speed

Body Domain
- CentralLockingMaster
- InteriorLight

Powertrain Domain
- DriverReq

Chassis Domain
- CentralLockingMaster
- ESP
- ACC
- SSM
- EPB
- VLC

DataElement
- Name: VehicleSpeed
- value: t_VehicleSpeed
- Min Bit size: 12
- Res: 0.1
- phys low: 0.0
- phys up: 403.4
- Unit: km/h

DataElement
- Name: ActualVehicleSpeed
- DataType: VehicleSpeed

DataElement
- Name: VehicleLongSpeed
- DataType: VehicleLongitudinalSpeed

Error indication areas:
AUTOSAR Application Interfaces – Conclusion

1. For several domains a subset of application interfaces has been standardized to agreed levels.

2. It is a challenge to align standardization with the pace of application development.
Wrap-up
Distribution on ECUs

SwitchEvent

LightRequest

Front-Light Manager

Xenonlight

AUTOSAR Int.

switch_event(event)

request_light(type, mode)

set_light(type, mode)

set_current(…)

AUTOSAR Interface

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface

get_keyposition()

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface

request_light(type, mode)

set_light(type, mode)

AUTOSAR Interface
Use case ‘Front-Light Management’ in AUTOSAR

**SwitchEvent**
- `check_switch()`
- `switch_event(event)`

**LightRequest**
- `switch_event(event)`
- `request_light(type, mode)`

**Front-Light Manager**
- `request_light(type, mode)`
- `get_keyposition()`
- `set_light(type, mode)`

**Xenonlight**
- `set_light(type, mode)`
- `set_current(...)`

**Microcontroller Abstraction**
- `get_keyposition()`
- `request_light(type, mode)`
- `set_light(type, mode)`

**ECU-Hardware**
- `check_switch()`
- `switch_event(event)`

**AUTOSAR Int.**
- `switch_event(event)`
- `request_light(type, mode)`

**AUTOSAR Interface**
- `AUTOSAR Int.`
- `AUTOSAR Interface`

**AUTOSAR RTE**
- `AUTOSAR RTE`
- `AUTOSAR RTE`

**Std. AUTOSAR Interface**
- `Std. AUTOSAR Interface`
- `Std. Interface`

**Services**
- `Services`
- `Services`

**Std. Interface**
- `Std. Interface`
- `Std. Interface`

**Std. Interface**
- `Std. Interface`
- `Std. Interface`

**DIO**
- `DIO`

**CAN Driver**
- `CAN Driver`

**ECU Abstraction**
- `ECU Abstraction`
- `ECU Abstraction`

**Microcontroller Abstraction**
- `Microcontroller Abstraction`

**ECU-Hardware**
- `ECU-Hardware`

**CAN Bus**
- `CAN Bus`
Automotive Software Development will change.

- Hardware- and software will be widely independent of each other.
- Development processes will be simplified. This reduces development time and costs.
- Reuse of software increases at OEM as well as at suppliers. This enhances also quality and efficiency.

Automotive Software will become a product.
AUTOSAR is ready to be used automotive product development
Exploitation has already started
AUTOSAR welcomes new members

“Cooperate on standards, compete on implementation.”
Thank you for your attention!

http://www.autosar.org

request@autosar.org