AUTOSAR Introduction

Part 2 – Current Features in a Nutshell
Agenda

Part 1
► The AUTOSAR Partnership
► The AUTOSAR Standardization

Part 2
► Architecture and Features
  • AUTOSAR in Vehicle Network
  • AUTOSAR Foundation
  • AUTOSAR Classic Platform
  • AUTOSAR Adaptive Platform
► Smart Solutions Based on AUTOSAR
► Processes and Quality
AUTOSAR in a Vehicle Network

Application Layer

RTE

AUTOSAR Classic Platform

Application Layer

RTE

AUTOSAR Classic Platform

Application Layer

AUTOSAR Adaptive Platform

Applications

Non-AUTOSAR Basic Software

Applications

Non-AUTOSAR Basic Software

Common Bus Interface Specification

e.g. SOME / IP
AUTOSAR Foundation

Common Features

The Foundation **assures compatibility** of the different AUTOSAR standards and therefore **contains** all **common artifacts** such as …
... provides means to describe the AUTOSAR **architecture** with all its **interfaces**

... defines **exchange formats** and description **templates** (e.g. manifest) to enable

- a seamless integration of the complete vehicle E/E architecture,
- the automatized configuration of the µC- and µP-software stacks and
- the seamless integration of application software

... supports means to **ensure safety** and **security** of the system

... provides templates to **document the standard**
The layered architecture of the classic platform basically supports

- Hardware abstraction
- Scheduling of runnables and tasks (OS)
- Communication between applications on the same hardware and over the network
- Diagnosis and diagnostic services
- Safety- and
- Security Services
AUTOSAR Classic Platform
Layered Software Architecture (2/2)

- Application Layer
  - System Services
  - Memory Services
  - Crypto Services
  - Off-board Communication Services
  - Communication Services
  - I/O HW Abstraction
  - Complex Drivers

- Runtime Environment
  - Onboard Device Abstraction
  - Memory HW Abstraction
  - Crypto HW Abstraction
  - Wireless Communication HW Abstraction
  - Communication HW Abstraction
  - Communication Services
  - I/O Drivers

- Microcontroller
  - Microcontroller Drivers
  - Memory Drivers
  - Crypto Drivers
  - Wireless Communication Drivers
  - Communication Services
  - I/O Drivers
AUTOSAR Classic Platform
Layered Software Architecture (2/2)
AUTOSAR Adaptive Platform
Architecture - Logical view

User Applications

- Adaptive Application
- Adaptive Application
- Adaptive Application
- ASW::XYZ Non-PF Service
- ASW::ABC Non-PF Service

AUTOSAR Runtime for Adaptive Applications (ARA)

<table>
<thead>
<tr>
<th>Service</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ara::com</td>
<td>Communication Management</td>
</tr>
<tr>
<td>ara::rest</td>
<td>REST</td>
</tr>
<tr>
<td>ara::tsync</td>
<td>Time Synchronization</td>
</tr>
<tr>
<td>ara::diag</td>
<td>Diagnostics</td>
</tr>
<tr>
<td>ara::sm service</td>
<td>State Management</td>
</tr>
<tr>
<td>ara::per</td>
<td>Persistency</td>
</tr>
<tr>
<td>ara::phm</td>
<td>Platform Health Management</td>
</tr>
<tr>
<td>ara::log</td>
<td>Log and Trace</td>
</tr>
<tr>
<td>ara::ucm service</td>
<td>Update and Config Management</td>
</tr>
<tr>
<td>ara::core</td>
<td>Core</td>
</tr>
<tr>
<td>ara::exec</td>
<td>Execution Management</td>
</tr>
<tr>
<td>ara::iam</td>
<td>Identity and Access Management</td>
</tr>
<tr>
<td>ara::crypto</td>
<td>Cryptography</td>
</tr>
<tr>
<td>ara::nm service</td>
<td>Network Management</td>
</tr>
</tbody>
</table>

Key

- SERVICE Non-PF Service
- SERVICE Platform Service FCs
- API Platform Foundation FCs

POSIX PSE51 / C++ STL
Operating System Interface

(Virtual) Machine / Container / Hardware
Agenda

Part 1
► The AUTOSAR Partnership
► The AUTOSAR Standardization

Part 2
► Architecture and Features
► Smart Solutions Based on AUTOSAR
  • Software Architecture – AUTOSAR Defined Interfaces
  • Distribution ECUs
  • AUTOSAR Platform Application
► Processes and Quality
Use Case ‘Front Light Management’: Exchange Type of Front Light

- SwitchEvent:
  - check_switch()

- LightRequest:
  - switch_event(event)
  - request_light(type, mode)

- Front-Light Manager:
  - request_light(type, mode)
  - get_keyposition()
  - set_light(type, mode)
  - set_dboard(type, mode)

- SwitchEvent:
  - switch_event(event)

- LightRequest:
  - request_light(type, mode)

- ECU Abstraction:
  - Communication
  - Std. Interface

- Std. AUTOSAR Interface:
  - Services
  - Std. Interface

- AUTOSAR Interface:
  - Request_light
  - Switch_event
  - Check_switch
  - Get_keyposition
  - Set_light
  - Set_dboard
  - Set_current

- CAN Driver
- PWM
- DIO

- Microcontroller Abstraction
- Std. Interface
Software Architecture – AUTOSAR Defined Interfaces

Use Case ‘Front Light Management’: Exchange Type of Front Light

- **Integrator**
  - SwitchEvent
    - check_switch()
  - Switch_event(event)

- **Supplier B**
  - LightRequest
    - switch event (event)
  - request_light (type, mode)

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

- **Supplier A**
  - Xenonlight
    - set_light (type, mode)
    - set_current(…)

- **Silicon Vendor A**
  - AUTOSAR Interface

- **ECU-Hardware**
  - Microcontroller Abstraction
  - Standardized Interface
    - CAN Driver
    - DIO

- **Integrator Supplier B**
  - AUTOSAR Interface
  - Std. AUTOSAR Interface
    - Services
    - Std. Interface

- **OEM**
  - AUTOSAR Interface
  - Std. Interface
    - Communication
    - Std. Interface

- **Supplier A**
  - AUTOSAR Interface
  - Std. Interface
    - ECU Abstraction
    - Std. Interface

- **AUTOSAR RTE**

**Notes:**
- AUTOSAR introduced standardized interfaces between different components in the automotive domain, allowing for better integration and exchange of functionalities. This diagram illustrates the interactions and interfaces defined by AUTOSAR, focusing on the front light management scenario.
Distribution on ECUs – ‘Front-Light Management’

- **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(…)

- **CAN Bus**

- AUTOSAR Interface

- AUTOSAR Int.

- AUTOSAR Interface

- AUTOSAR Interface

- AUTOSAR Interface
Distribution on ECUs – ‘Front-Light Management’
AUTOSAR Platform Application
Continuous improvement cycle for ADAS systems

On Board
- Scene Understanding
- Sensor Fusion
- Perception
- ara:adi
- ara:com
- Plan Execution, Motion Control
- Mission and Trajectory Planning

Secure data exchange on top of
- DDS
- SOME/IP
- or
- REST

Off Board
- Labeling of trained data
- Evolution by training
- Simulation and sign-off

21-Jul-2021
AUTOSAR Introduction - Part 2
User Applications

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ara::com</td>
<td>ara::rest</td>
<td>ara::time</td>
<td>ara::phm</td>
<td>ara::per</td>
<td>ara::am</td>
<td>ara::log</td>
<td>ara::crypto</td>
<td>ara::com</td>
<td></td>
</tr>
<tr>
<td>Vehicle Manager</td>
<td>RESTful</td>
<td>Time Synchronization</td>
<td>Platform Health Mgmt.</td>
<td>Persistence</td>
<td>Identity Access Mgmt.</td>
<td>Logging &amp; Tracing</td>
<td>Cryptography</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HMI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distance Radar</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The operation completed successfully

>> Run Execution Manager
...
...
...
ara::exec > start (Vehicle Manager)
The operation completed successfully
ara::exec > start (HMI)
The operation completed successfully
ara::exec > start (Distance Radar)
The operation completed successfully
Adaptive Cruise Control

User Applications

- ara::com Communication Mgmt.
- ara::rest RESTful
- ara::time Time Synchronization
- ara::per Persistence
- ara::ucm service (User Applications Management)
- ara::evm service (Error and Event Management)
- ara::log Logging & Tracing
- ara::ucm service (Update and Configuration Management)
- ara::state service State Management
- ara::diag service Diagnostics
- ara::adi service Automated Driving Interfaces
- ara::core Core Types
- ara::exec Execution Mgmt.
- ara::iam Identity Access Mgmt.
- ara::xss service Signal to Service Mapping
- ara::uci service Update Interface Management
- ara::xsm service Signalling Management

Run Persistency

- Install new application software from ara::ucm service (ACC)
- Deploy persistent data to Key-Value Database:
  - Vehicle Manager
  - HMI
  - Distance Radar
  - ACC

AUTOBOOK Runtime for Adaptive Applications (ARA)
<table>
<thead>
<tr>
<th>Vehicle Manager</th>
<th>HMI</th>
<th>Distance Radar</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>ara::com</td>
<td>ara::time</td>
<td>ara::rest</td>
<td>ara::state service</td>
</tr>
<tr>
<td>Communication Mgmt.</td>
<td>Time Synchronization</td>
<td>RESTful</td>
<td>Management</td>
</tr>
<tr>
<td>ara::core</td>
<td>ara::per</td>
<td>ara::exec</td>
<td>ara::diagn service</td>
</tr>
<tr>
<td>Core Types</td>
<td>Persistency</td>
<td>Execution Mgmt.</td>
<td>Diagnostics</td>
</tr>
<tr>
<td>ara::idm</td>
<td>ara::phm</td>
<td>ara::log</td>
<td>ara::adi service</td>
</tr>
<tr>
<td>Identity Access Mgmt.</td>
<td>Platform Health Mgmt.</td>
<td>Logging &amp; Tracing</td>
<td>Automated Driving Interfaces</td>
</tr>
<tr>
<td>POSIX PSE51 / C++ STL</td>
<td>ara::crypto</td>
<td>ara::ss service</td>
<td></td>
</tr>
<tr>
<td>Operating System</td>
<td>Cryptography</td>
<td>Signal to Service Mapping</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ara::nm service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Network Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ara::com service</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Update and Configuration Management</td>
<td></td>
</tr>
</tbody>
</table>

AUTOSAR Runtime for Adaptive Applications (ARA)

(Virtual) Machine / Container / Hardware
AUTOSAR Introduction - Part 2
Agenda

Part 1
► The AUTOSAR Partnership
► The AUTOSAR Standardization

Part 2
► Architecture and Features
► Smart Solutions Based on AUTOSAR
► Processes and Quality
  • AUTOSAR Adaptive Platform Development Approach
AUTOSAR Adaptive Platform Development Approach

**Specification**

Identify needs & use-cases:
1) Concepts
2) Features
3) Requirements

Quality:
• TF-ARC approval
• Cross team review
• Lifecycle: preliminary → draft → valid

**Implementation**

Gain speed:
1) Spec validation
2) Reduce room for spec interpretation
3) Training / dissemination of AP

Attracting environment for coders:
• Appealing technology (C++, Yocto, Git, …)
• Modern use case (ADAS EBA)
• Handy documentation (Wiki)
• Peer programming sessions

**Demonstration**

Gain trust:
1) Advertises the progress
2) Highlights some specific features

Show AUTOSAR interoperability
• of classic and adaptive platforms
• but also with others

---

Best tradeoff between commercial cooperation & compatibility between different vendors
Thank you for your attention

If you’d like to become a partner, contact us at:

+49 89 23 88 57 410  
admin@autosar.org  
http://autosar.org  

Bremer Str. 11  
80807 Munich  
Germany