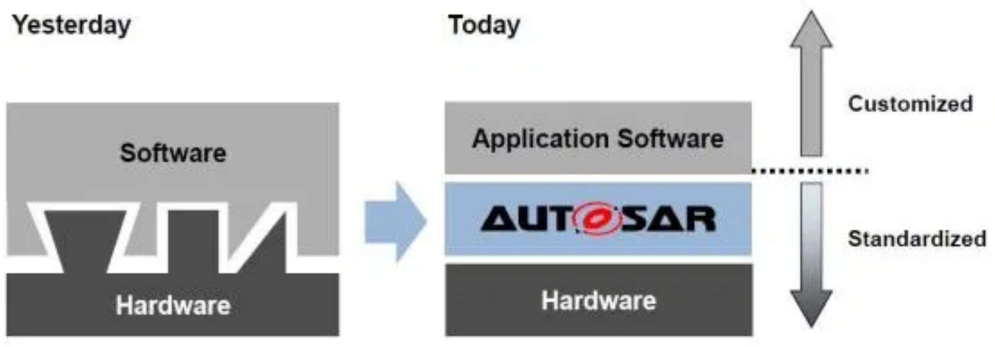
Seamless To Seattle
Fly non-stop from Singapore to Seattle on Singapore Airlines

IOTIVE BASICS

collective information..

S

AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture, jointly developed by automobile manufacturers, suppliers and tool developers. The AUTOSAR-standard enables the use of a component based software design model for the design of a vehicular system. The design model uses application software components which are linked through an abstract component, named the virtual function bus.



The application software components are the smallest pieces of application software that still have a certain functionality. The software of an application can then be composed by using different application software-components. Standardized interfaces for all the application software components necessary to build the different automotive applications are specified in the **AUTOSAR**-standards. By only defining the interfaces, there is still freedom in the way of obtaining the functionality.

The virtual function bus connects the different software components in the design model. This abstract component interconnects the different application software components and handles the information exchange between them. The virtual function bus is the conceptualization of all hardware and system services offered by the vehicular system. This makes it possible for the designers to focus on the application instead of the infrastructure software.

By using the virtual function bus, the application software components do not need to know with which other application software components they communicate. The software components give their output to the virtual function bus, which guides the information to the input ports of the software components that need that information. This is possible due to the standardized interfaces of the software components which specifies the input and output ports as well as the format of data exchange.

This approach makes it possible to validate the interaction of all components and interfaces before software implementation. This is also a fast way to make changes in the system design and check whether the system will still function.

The AUTOSAR-project created a methodology that can be used to create the E/E system architecture starting from the design-model. This approach uses 4 steps:

Step 1: Input Descriptions

The input description step contains three descriptions:

- Software Components: This description is independent of the actual implementation of the software component. Among the necessary data to be specified are the interfaces and the hardware requirements.
- System: The system topology (interconnections between ECUs) need to be specified together with the available data busses, used protocols, function clustering and communication matrix and attributes (e.g. data rates, timing/latency, …).
- Hardware: The available hardware (processors, sensors, actuators, …) needs to be specified together with the signal processing methods and programming capabilities

Step 2: System Configuration

This step distributes the software component descriptions to the different ECU. This is an iterative process where ECU-resources and system-constraints are taken into account. For example, there needs to be checked whether the necessary communication-speeds are met.

Step 3: ECU-configuration

In this step, the Basic Software and the Run Time Environment of each electronic control unit (ECU) is configured. This is based on the dedication of the application software components to each ECU.

Step 4: Generation of Software Executables

Based on the configuration of the previous step, the software executables are generated. For this step, it's necessary to specify the implementation of each software component.

This methodology is automated by using tool-chains. All subsequent methodology steps up to the generation of executables are supported by defining exchange formats (using XML) and work methods for each step.

To support the Autosar-methodology, a metamodel is developed. This is a formal description of all methodology related information, modeled in UML. This leads to the following benefits:

- 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 meta-model and be used as input for the methodology.
- Easy maintenance of the entire vehicular system

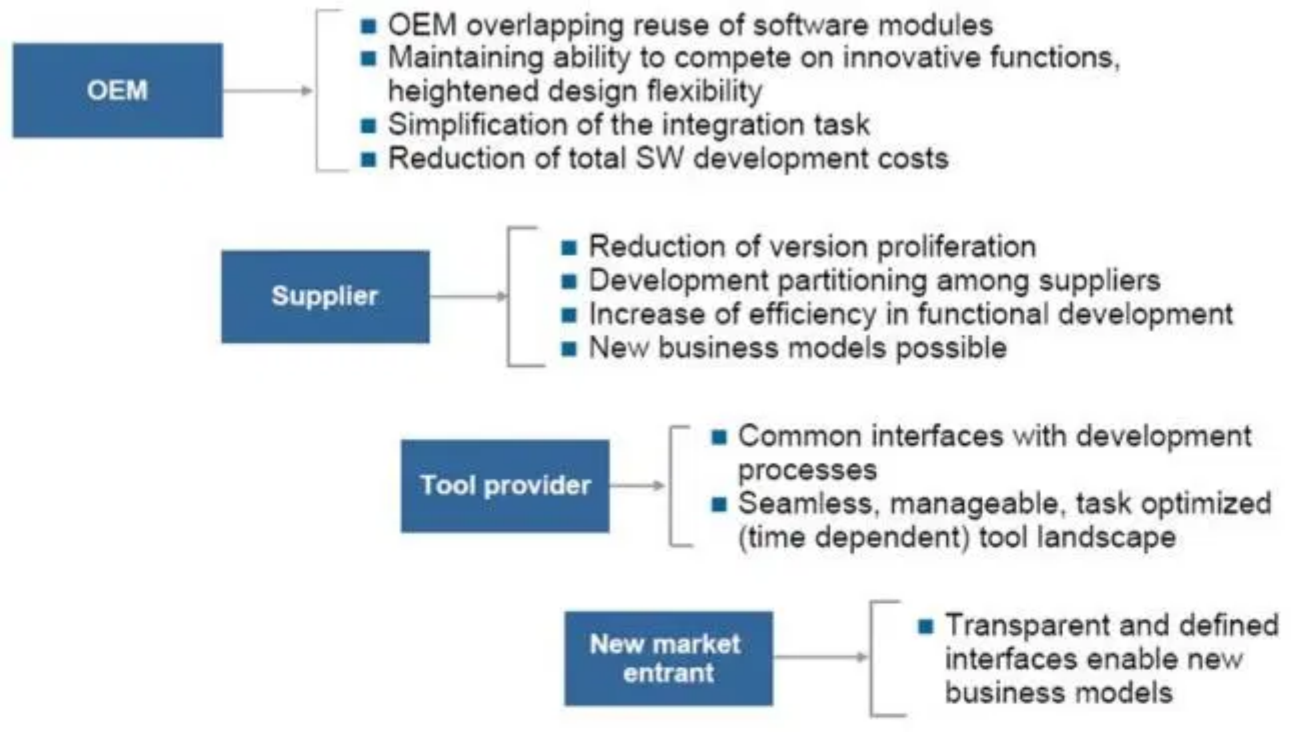
There are four types of membership for AUTOSAR:

- Core (founding) members
- Premium members
- Associate members
- Development members

Core membership only is available for leading car manufacturers and Tier1; the other typesof membership are open to other companies as well.

Core members include the PSA peugueot citroien, Toyata Motor Corporation, Volkswagen , BMW Group, Daimler AG, Ford Motor Company, Opel , and automotive suppliers Bosch, Continental AG and Siemens VDO (now Continental AG).

Benefits of Autosar:


OEM
OEM overlapping reuse of software modules
Maintaining ability to compete on innovative functions, heightened design flexibility
Simplification of the integration task
Reduction of total SW development costs
Supplier
Reduction of version proliferation
Development partitioning among suppliers
Increase of efficiency in functional development
New business models possible
Tool provider
Common interfaces with development processes
Seamless, manageable, task optimized (time dependent) tool landscape
New market entrant
Transparent and defined interfaces enable new business models

What are the pitfalls on path to a future where the automotive industry embrace and incorporate the AUTOSAR vision?

First the industry needs to gain a common understanding on the usage of models and the different levels of abstraction. AUTOSAR are concrete design elements, while OEMs traditionally have a focus on functionality. AUTOSAR puts more responsibility on the OEM, the challenge is to transform functional requirements into AUTOSAR components and putthem together in a system.

The introduction of AUTOSAR in form of predefined design elements can lead to a cultural clash in the industry where the traditional development process is the waterfall. The development of the design elements is traditionally top-down where the elements are a result of a stepwise refining of the desired concepts in the project. The introduction of AUTOSAR is a restriction of development freedom; from the early phases in the development the resulting design must be aimed at AUTOSAR. Unresolved technical issues might impede a migration to AUTOSAR as well. Support from tools and interoperability must be resolved before a successful implementation. Even if all technical issues are resolved, the implementation of AUTOSAR on a system level is a huge leap. The development processes of vehicles E/E architectures are built on previous architectures. The scenario of an entire project starting from scratch embracing AUTOSAR is not likely.

AUTOSAR Meta Model is the backbone of the AUTOSAR architecture definition contains complete specification, how to model AUTOSAR system.

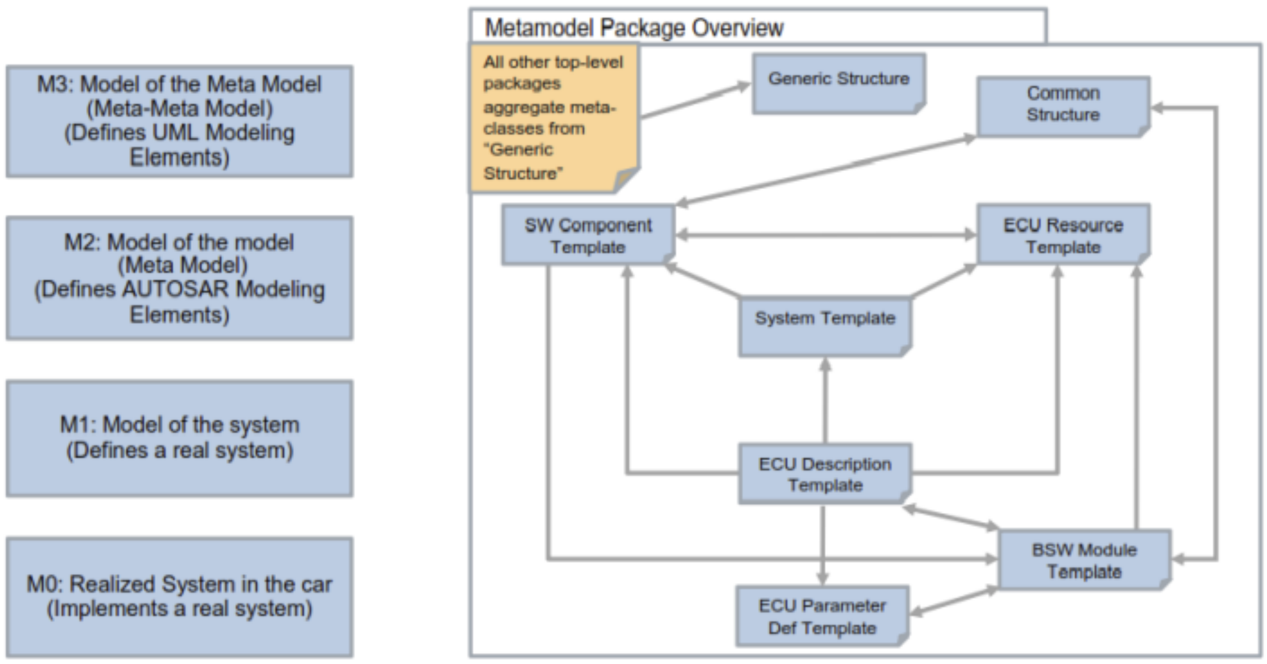

M3: Model of the Meta Model (Meta-Meta Model) (Defines UML Modeling Elements)
M2: Model of the model (Meta Model) (Defines AUTOSAR Modeling Elements)
M1: Model of the system (Defines a real system)
M0: Realized System in the car (Implements a real system)
Metamodel Package Overview
All other top-level packages aggregate meta-classes from "Generic Structure"
Generic Structure
Common Structure
SW Component Template
ECU Resource Template
System Template
ECU Description Template
BSW Module Template
ECU Parameter Def Template

Figure 4 : Autosar MetaModel Overview

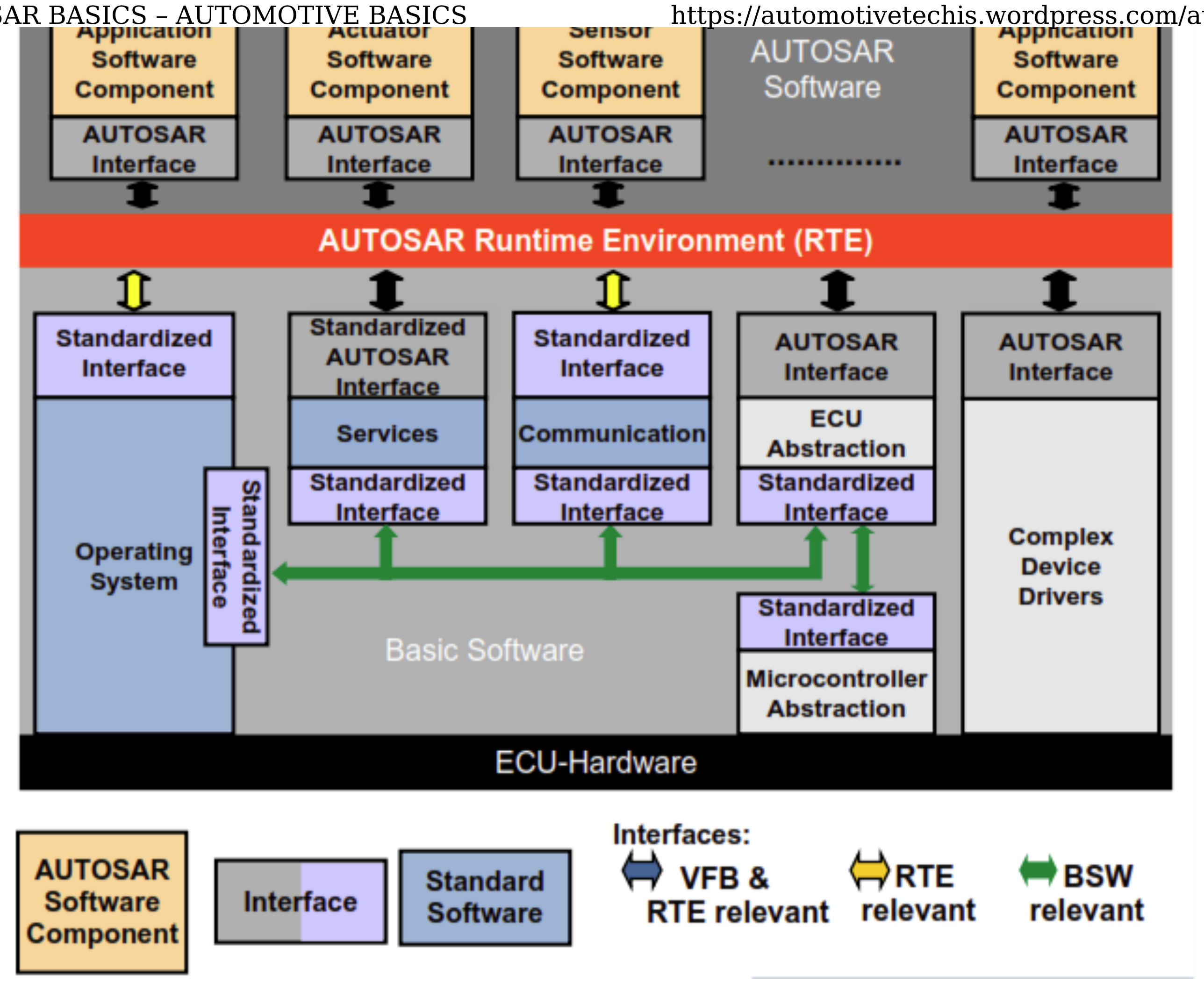


Figure 6: Autosar Layered Architecture

Classification Of Interfaces:

There are three different types of interfaces in Autosar Layered Architecture.

Standardized Autosar Interfaces:

- A Standardized AUTOSAR Interface is an AUTOSAR Interface standardized within the AUTOSAR project.

Standardized Interfaces:

- A software interface is called Standardized Interface if a concrete standardized API exists (e.g. OSEK COM Interface Com_ReceiveSignal & Com_TransmitSignal which are called by RTE module)

Autosar Interfaces:

- An AUTOSAR Interface describes the data and services required or provided by a component and is specified and implemented according to the AUTOSAR Interface Definition Language. An AUTOSAR Interface is partly standardized within AUTOSAR, e.g. it may include OEM specific aspects. The use of AUTOSAR Interfaces allows software components to be distributed among several ECUs. The RTEs on the ECUs will take care of making the distribution transparent to the software components.

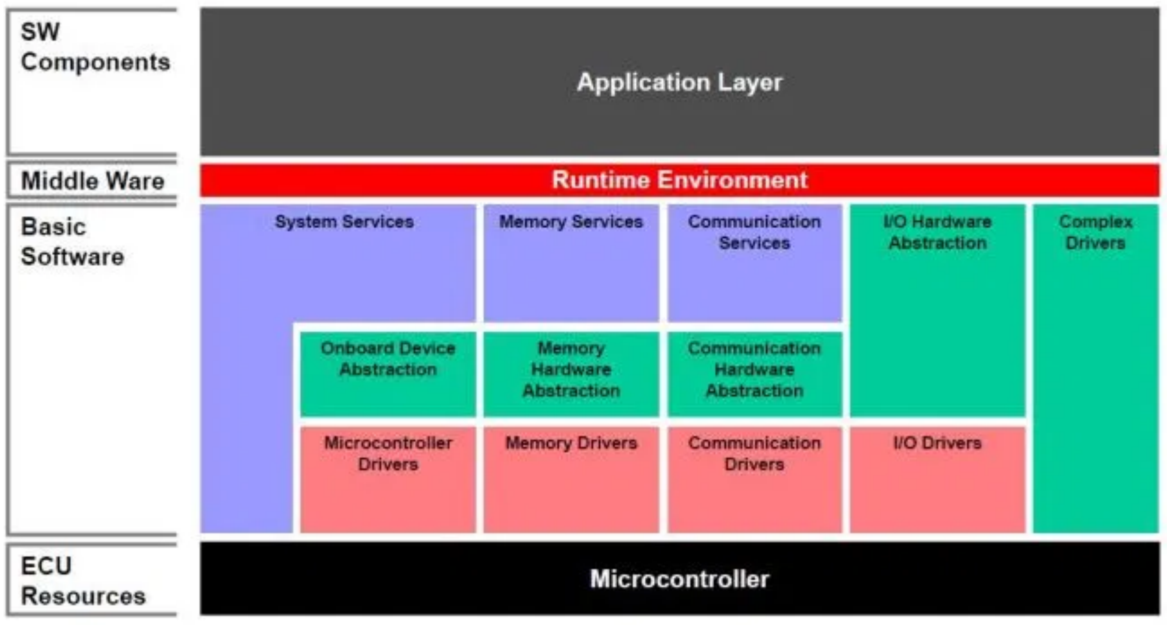


Figure 5: Autosar Layered Architecture

How are vehicle functions implemented today?

• Each function has it´s own system although they may communicate through a bus

• Hardware and software are tightly connected

• Each function has it´s own microcontroller

• The number of ECU´s (Electronic Control Unit) are growning fast

• The same vendor supplies both the hardware and the software

• There are no alternative software suppliers

What will Autosar give?

• A standard platform for vehicle software

• An OS with basic functions and interface to software

• Functionality is supplied as software components

• An with basic functions and interface to software

• These components are hardware independent

• No applications of its own

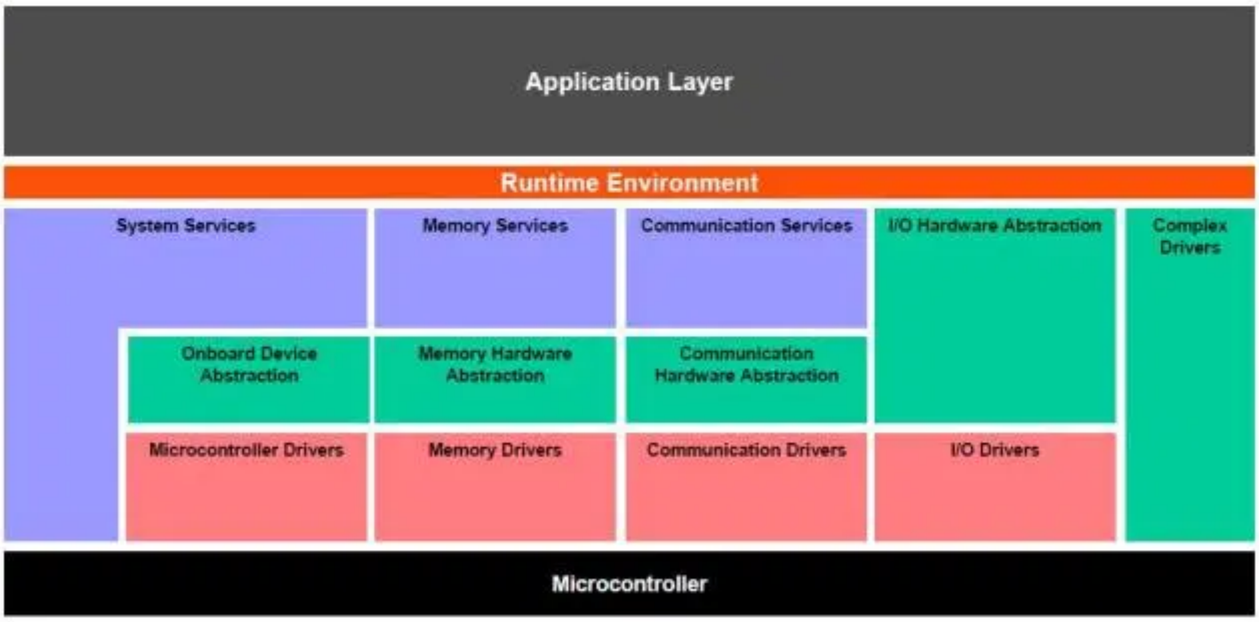
• The same interface for all basic software

• The software is exchangable

• The software can be reused

• More than one supplier can compete with their software

The Basic Software Layers are further divided into functional groups. Examples of Services are System, Memory and Communication Services.



The **Microcontroller Abstraction Layer** is the lowest software layer of the Basic Software.

It contains internal drivers, which are software modules with direct access to the µC and internal peripherals.

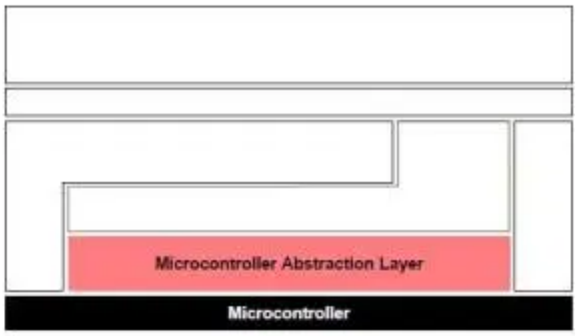
Task

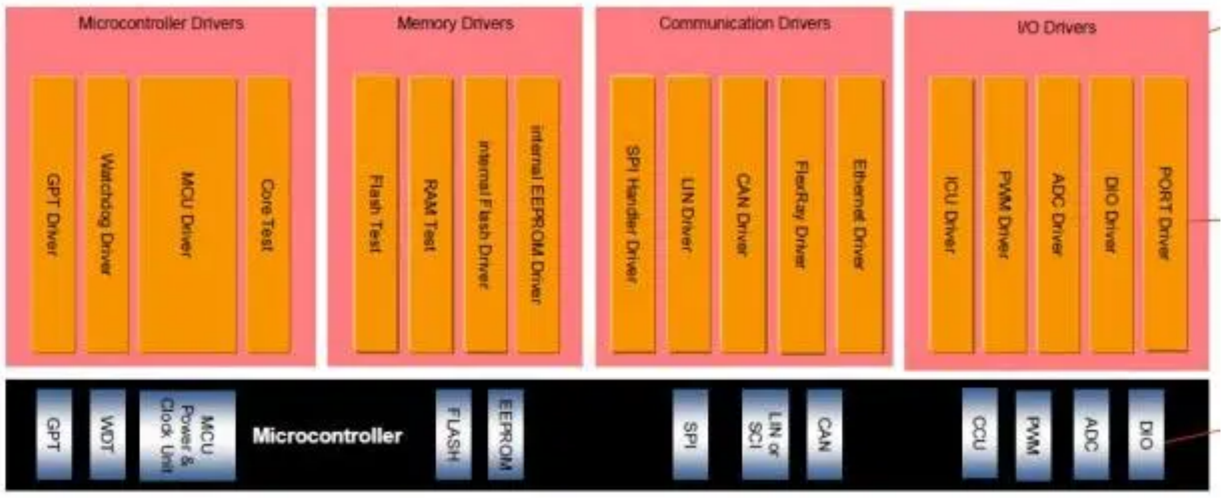
Make higher software layers independent of µC

Properties

Implementation: µC dependent

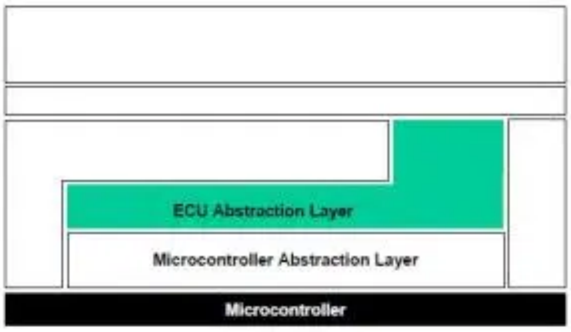
Upper Interface: standardized and µC independent





The **ECU Abstraction Layer** interfaces the drivers of the Microcontroller Abstraction Layer. It also contains drivers for external devices.

It offers an API for access to peripherals and devices regardless of their location (µC internal/external) and their connection to the µC (port pins, type of interface)



Task

Make higher software layers independent of ECU hardware layout

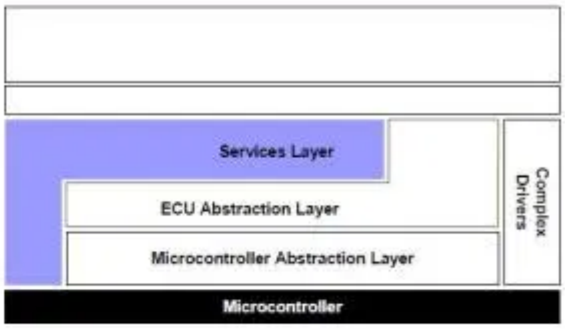
Properties

Implementation: µC independent, ECU hardware dependent

Upper Interface: µC and ECU hardware independent

The **Services Layer** is the highest layer of the Basic Software which also applies for its relevance for the application software: while access to I/O signals is covered by the ECU Abstraction Layer, the Services Layer offers:

- Operating system functionality
- Vehicle network communication and management services
- Memory services (NVRAM management)
- Diagnostic Services (including UDS communication, error memory and fault treatment)
- ECU state management, mode management
- Logical and temporal program flow monitoring (Wdg manager)



Task

Provide basic services for applications and basic software modules.

Properties

Implementation: mostly µC and ECU hardware independent

Upper Interface: µC and ECU hardware independent

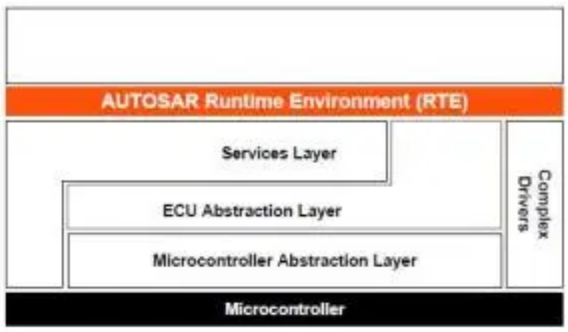
The **RTE** is a layer providing communication services to the application software (AUTOSAR Software Components and/or AUTOSAR Sensor/Actuator components).

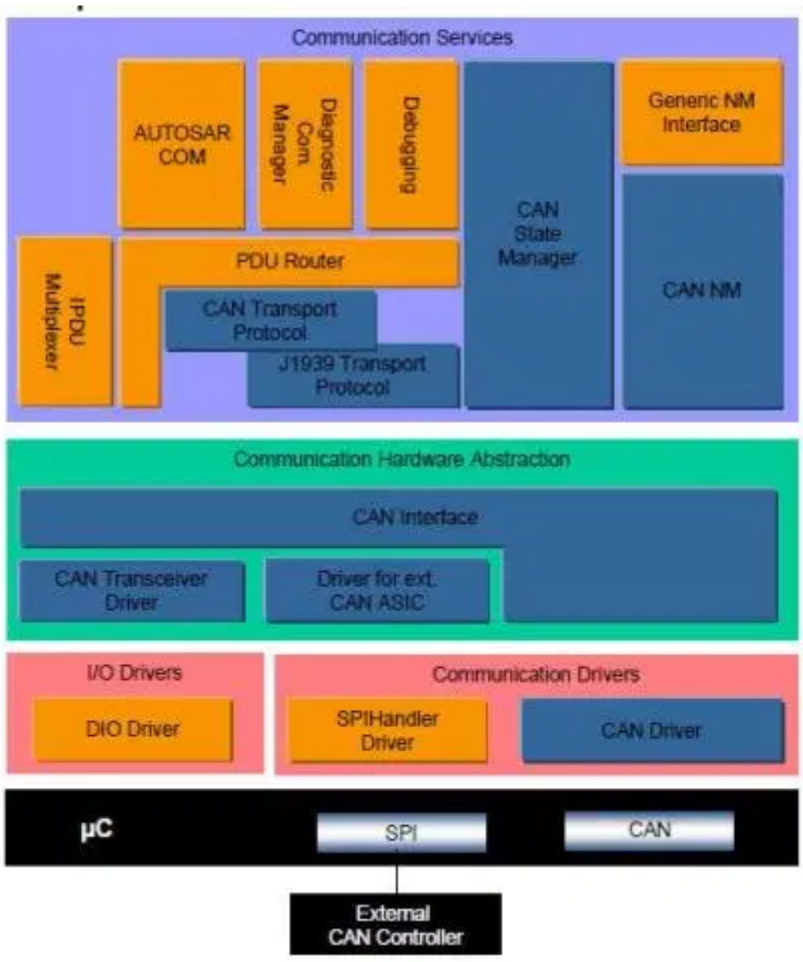
Above the RTE the software architecture style changes from "layered" to "component style".

The AUTOSAR Software Components communicate with other components (inter and/or intra ECU) and/or services via the RTE.

Task
Make AUTOSAR Software Components independent from the mapping to a specific ECU.

Properties
Implementation: ECU and application specific (generated individually for each ECU)
Upper Interface: completely ECU independent


AUTOSAR Runtime Environment (RTE)
Services Layer
ECU Abstraction Layer
Microcontroller Abstraction Layer
Complex Drivers
Microcontroller



CAN Communication:

Application Layer and RTE

Applications written in the context of AUTOSAR consist of components. These components communicate with each other via ports (component view). The communication between two components can consist of a single (AUTOSAR) signal or a whole signal group. From the view of the AUTOSAR SWC it is not known at implementation time, which communication media is used. All bus specific replications of send requests by a SWC to underlying layers and bus specific timing behavior must be done by COM or by the appropriate bus interfaces and drivers. The RTE uses the capability to send and receive signals of AUTOSAR COM. VFB's send modes corresponding to the transfer property of a signal and the transmission mode of an I-PDU.

Transmission Modes and Transmission Model Selection

COM shall provide different transmission modes for each I-PDU.

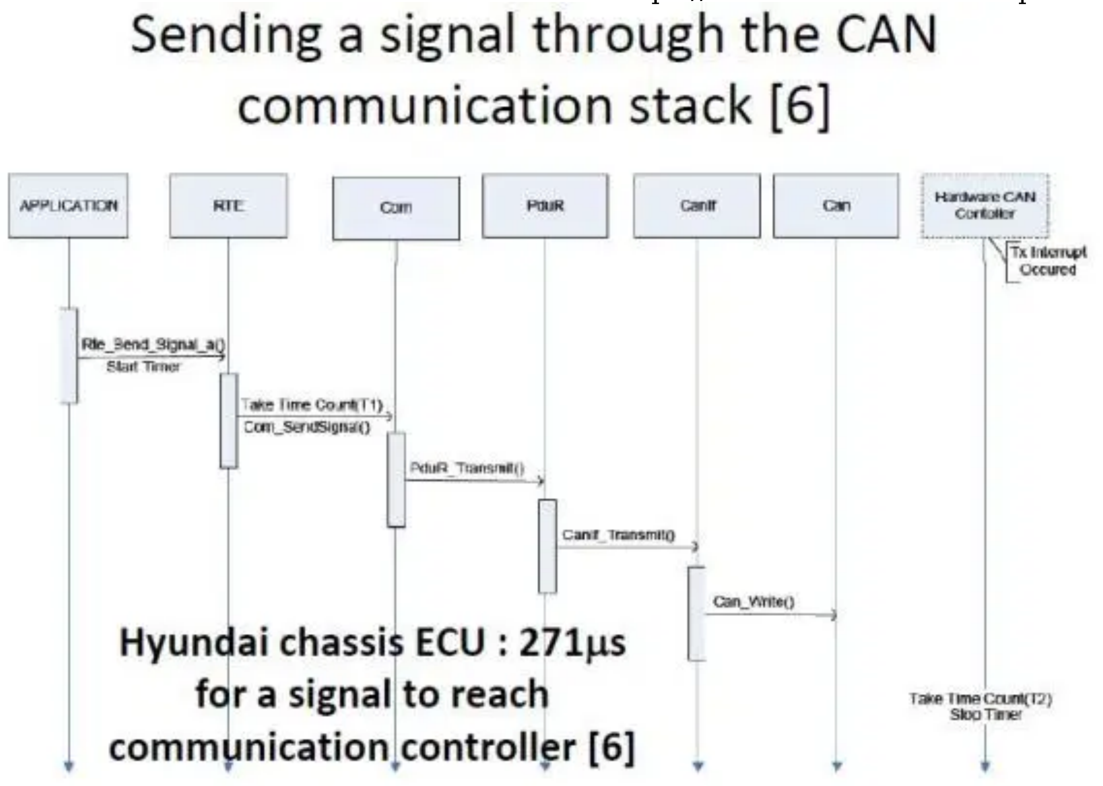
Periodic: transmissions occur indefinitely with a fixed period between them

Direct / n-times: event driven transmission with n-1 repetitions

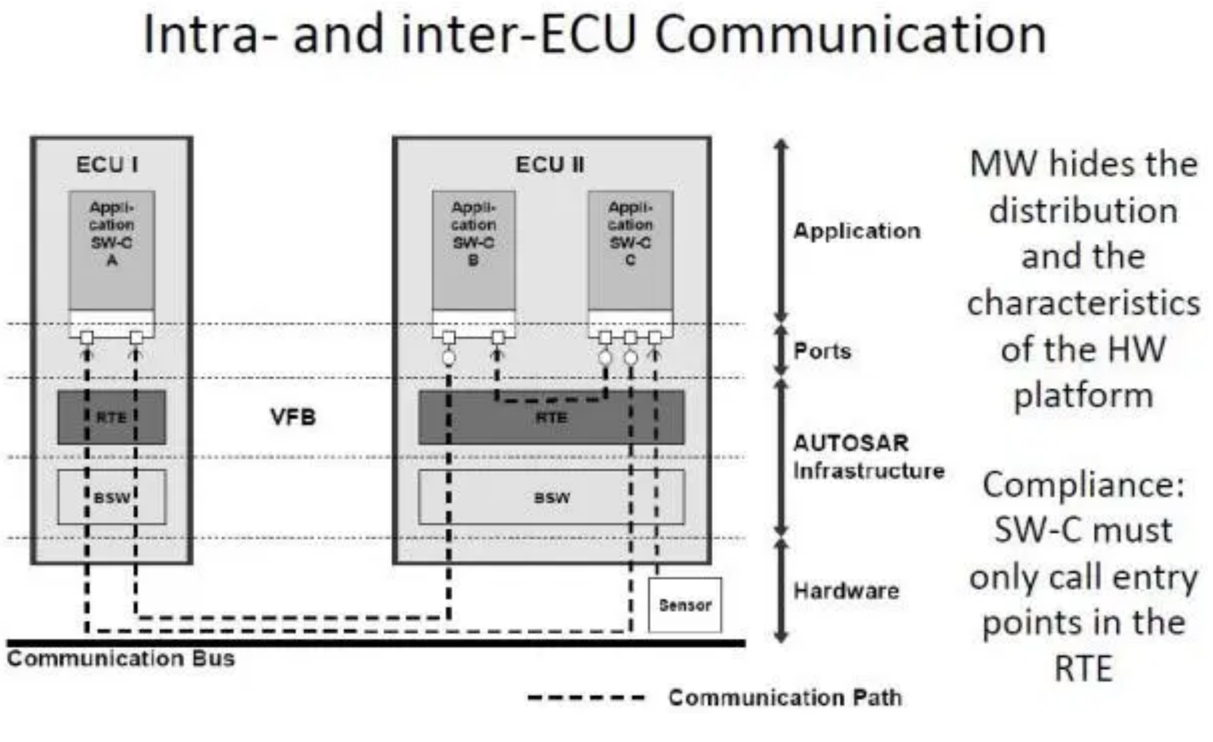
Mixed: periodic transmission with direct/n-times transmissions in between

None: no transmission

Two of these transmission modes shall be supported for each PDU so that it will be possible to switch between both modes during runtime. To decide which of the two transmission modes is selected, COM shall provide the possibility to attach a condition to each signal within an I-PDU separately. If all conditions that are defined for signals within one specific I-PDU evaluate to true then one transmission mode shall be used for this I-PDU. If at least a single condition defined for a signal within this I-PDU evaluates not to true, then the other mode shall be used. These conditions shall be checked directly if a related signal or signal group is sent by the RTE. The attached condition on a signal for evaluating the transmission mode for an I-PDU is called transfer property. A transfer property of a signal can either be triggered or pending. A transfer property of a signal with the triggeredvalue causes an immediate transmission of the I-PDU except if transmission mode periodic or none is defined for the I-PDU. If the transfer property of a signal is pending, no transmission of an I-PDU is caused. Because of I-PDUs can contain more than one single signal there is needed a method to derive the I-PDU's transmission mode from the state of signals that are contained in one specific I-PDU. For this method signals within a signal group are treated like normal signals. For each I-PDU there is defined a Transmission Mode Selector (TMS). The TMS is calculated by evaluating the Transmission Mode Conditions (TMC) of a con gurable subset of signals belonging to the specific I-PDU. The TMS is de ned to be true if at least one TMC of the con gurable subset of signals evaluates to true. If all TMCs evaluate to false the TMS is de ned to be false. If Com SendSignal or Com SendSignalGroup are called, the TMS of the IPDU shall be re-calculated. Figure 4.2 shows the mapping of signals into an I-PDU and the evaluation of the TMS. A detailed description of the selection of transmission modes is situated in.



In Autosar, two types of SWC communication **Intra & Inter**,



Client-Server Communication:

A widely used communication pattern in distributed systems is the client-server pattern, in which the server is a provider of a service and the client is a user of a service.
The client initiates the communication, requesting that the server performs a service, transferring a parameter set if necessary. The server waits for incoming communication requests from a client, performs the requested service, and dispatches a response to the client's request. The direction of initiation is used to categorize whether an AUTOSAR Software Component is a client or a server. A single component can be both a client and a server, depending on the software realization. The client can be blocked (synchronous

communication) or non-blocked (asynchronous communication), respectively, after the service request is initiated until the response of the server is received. The image gives an example how client-server communication for a composition of three software components and two connections is modeled in the VFB view.

Sender-Receiver Communication:

The sender-receiver pattern gives solution to the asynchronous distribution of information, where a sender distributes information to one or several receivers. The sender is not blocked (asynchronous communication) and neither expects nor gets a response from the receivers (data or control flow), i.e. the sender just provides the information and the receivers decides autonomously when and how to use this information. It is the responsibility of the communication infrastructure to distribute the information.
The sender component does not know the identity or the number of receivers to support transferability and exchange of AUTOSAR Software Components. The image illustrates an example how sender-receiver communication is modeled in the AUTOSAR VFB view.

The central structural element in AUTOSAR is the COMPONENT. A component has well-defined ports, through which it interacts with other components. A port always belongs to exactly one component. The AUTOSAR Interface concept defines the services or data that are provided on or required by a port of a component. The AUTOSAR Interface can either be a Client-Server Interface (defining a set of operations that can be invoked) or a Sender-Receiver Interface, which allows the usage of data-oriented communication mechanisms over the VFB. A port is either a PPort or an RPort. A PPort provides an AUTOSAR Interface while an RPort requires one.
When a PPort of a component provides an interface, the component to which the port belongs provides an implementation of the operations defined in the Client-Server Interface respectively generates the data described in a data-oriented Sender-Receiver Interface.
When an RPort of a component requires an AUTOSAR Interface, the component can either invoke the operations when the interface is a Client-Server Interface or alternatively read the data elements described in the Sender-Receiver Interface.

CAN Driver: The CAN Driver is part of the lower layer and offers the CAN Interfaceuniform interfaces to use. It hides hardware specific properties of the CAN Controller as far as possible. The CAN Driver performs the hardware access and provides a hardware independent API to the upper layer, the CAN interface (CanIf). Services for initiating transmission are offered by the CAN Driver and it calls the callback funtions of the CanIf module for notifying events hardware independently. In addition there are services provided by the CAN Driver module to control the state of all CAN controller belonging to the same CAN hardware unit. A CAN controller serves exactly one physical channel. A detailed description of the CAN bus is given in [30]. A CAN hardware unit is represented by one CAN Driver and either on chip or an external device. It may consist of one or multiple CAN controllers of the same type and one or multiple CAN RAM areas [29]. A single CAN Driver module can handle multiple CAN controllers if they belong to the same hardware unit. If an L-PDU shall be transmitted, the CAN Driver writes the L-PDU in a buffer inside the CAN controller hardware and if an L-PDU is received, the CAN Driver module calls the RX indication callback funtion with the L-PDUs ID, the DLC (see: ch. 2.3) and with a pointer to the L-SDU. The CAN Driver can access hardware resources and converts the given information for transmission into a hardware speci c format and triggers the transmission. The CAN Driver module offers the CanIf services to control the state of the CAN. Controllers by callback functions for bus-off and wake-up events. It implements the interrupt service routines for all CAN hardware unit interrupts that are needed. While startup the CAN Driver initializes static variables including flags, sets common settings for the complete CAN hardware unit and sets CAN controller specific settings for each CAN controller.

Communication Manager (ComM): The ComM is a resource manager which encapsulates the control of the underlying communication services. It controls the basic software modules related to communication and coordinates the bus communication access requests. The ComM shall simplify the usage of the bus communication stack for the user. It shall offer an API for disabling the sending of signals, shall be able to control more than one communication bus channel of an ECU and shall simplify the resource management by allocating all resources necessary for the requested communication mode. The COM Manager (ComM) controls the starting and stopping of sending and receiving I-PDUs via AUTOSAR COM. The NM is used by the ComM to synchronize the control of communication capabilities across the network.

CAN/FlexRay/LIN Bus State Manager: The actual bus states are controlled by the corresponding Bus State Manager. The actural states of the bus corresponds to a communication mode of the ComM. The ComM requests a specific communication mode from the state manager and the state manager shall map the communication mode to a bus state.

E.g. the comM uses the API of the CanSM to request communication modes of CANneworks. The CanSM uses the API of COM to controll CAN related PDU groups and it communicates with the CanIf to conrol the operating modes of the CAN controllers and to get noti ed by the CanIf about peripheral events.

Network Management Modules (NM) The Generic Nework Management Interface adapts the ComM to the bus specific network management modules. It provides an inteface to the ComM and uses services of the network management modules. The bus specific network management modules are CAN NM, FlexRay NM and LIN NM. The AUTOSAR NM Interface can only be applied to communication systems that support broadcast communication and bus-sleep mode. For network management data exchange the PDU Router module is bypassed.

DCM(**Diagnostic Communication Manager**): The main purpose of the DCM is providing a common API for diagnostic services. It is used while development, manufactoring or service by external diagnostic tools [25]. In gure 3.5 there is an overview of the communication over the DCM. The DCM performs the scheduling of diagnostic PDUs. It acts as a user by requesting full communication from the ComM if diagnostic shall be performed.

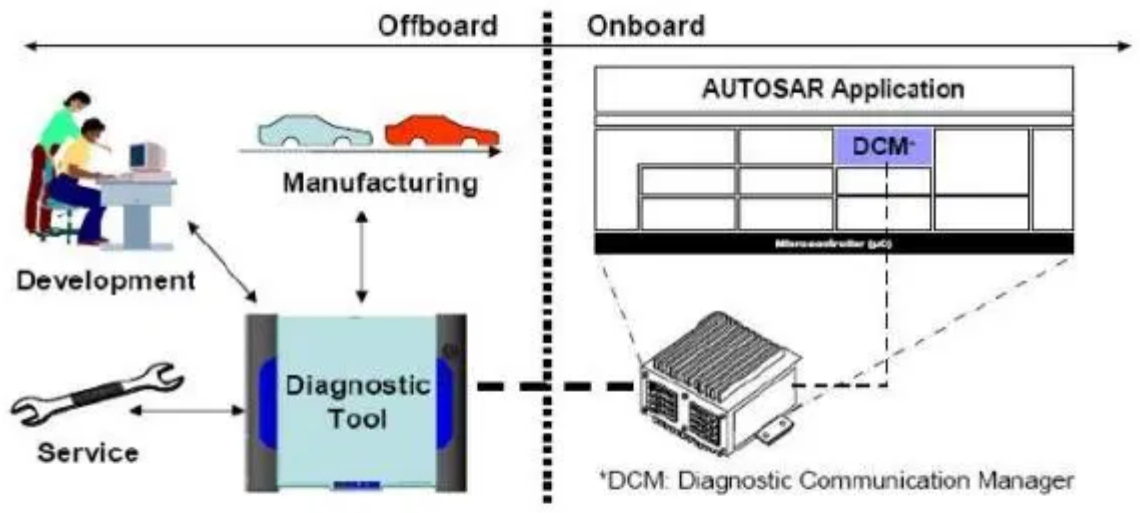


Fig. 3.5. Overview of the communication between the external diagnostic tools and the onboard AUTOSAR Application [25]

A more likely scenario is the injection of isolated AUTOSAR-"islands" in a project. This scenario gives few of AUTOSAR´s intended benefits in the short perspective but gives valuable AUTOSAR experience.

Drawbacks of AUTOSAR: The methodology used for mapping AUTOSAR concepts on existing has serious drawbacks.

•If both the existing concept and the newly introduced AUTOSAR concept carry the same information, there is a risk of inconsistency. There is also a risk of confusion over which concepts to use, but this is easily resolved by making the old concept abstract thereby forcing the use of the new concept

• Conflicting inheritance. Even if individual concepts map perfectly on each other, they canbe parts of inheritance structures violating AUTOSAR. On behalf of multiplied information and the risk of inconsistency the problem can theoretically be resolved with multiple inheritances. In practise the problems cannot be resolved; SystemWeaver does not support multiple inheritances. The final part of the implementation of the prototype meta model in an existing meta model is more of a demonstration piece; the implementation is what usually is called "ugly hack" In order to implement a fully working AUTOSAR meta model the usual customisation of the SystemWeaver is necessary, not only with respect of the customers needs but also with respect to AUTOSAR´s requirements.

Please find the documents to know in brief about AUTOSAR.

1. AUTOSAR_EXP_LayeredSoftwareArchitecture
2. Autosar_ppt
3. 27239727-Automotive-Embedded-System-Development-in-AUTOSAR
4. lesson19_autosar
5. CommunicationStack_gosda
6.

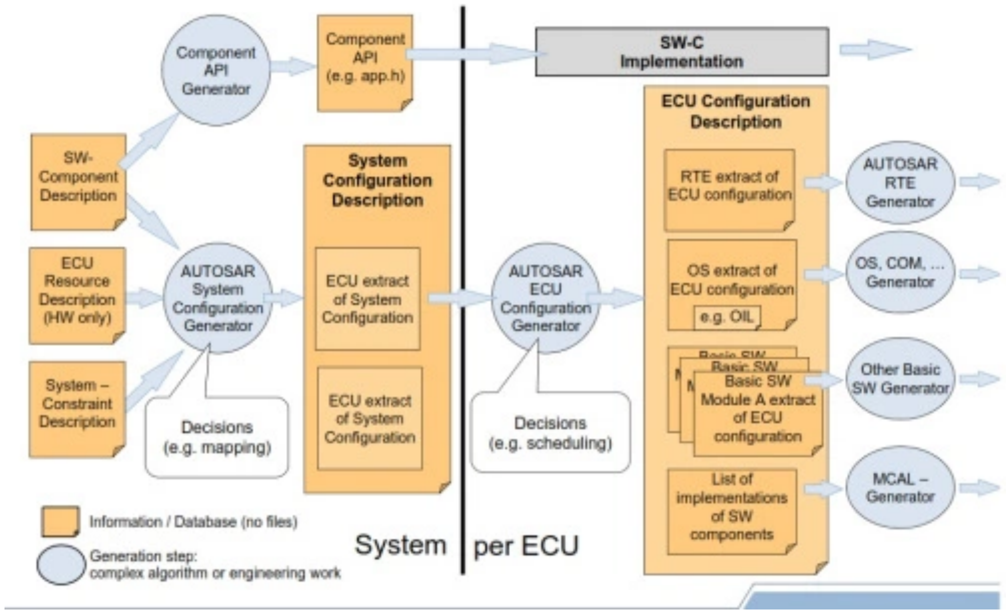
How they have standardised the Autosar Basic Software modules(BSW)?

In Every module, there are standardised functionality for example ADC, The main functionof ADC is to convert Analog to Digital conversion. So, we can develop standard functions but the channel, Bit resolution, Interrupts might be changing based upon the microcontroller and hardware pins. In Each module, we can divide into two types configuration files for hardware related information(like Channels, Groups, Pins, Pin Direction, Resolution) and Standard functions for modules main functionality. In each BSW (Basic Software module) Configuration files which are configured using tools(Eg: EBTresos, EcuSpectrum, DaVinci Configurator) we can generated configuration files and Standardized function which are defined as per autosar SWS specification.

How to generate the System Description files and ECU Description files?

Load the Dbc(Can Database) or LDF(Lin Description File) or Fibex (Field Bus Exchange) in Configuration tool and Configure the missing parameter as per System Description Template and extract the System Description arxml file using Export option. We can generate the ECU Description file using ECU extract option after loading the System Description file. After the extraction of ECU Description arxml load it configuration tool and configure the BSW modules as per autosar SWS Specification document.

- **System Configuration Description:**
  includes all system information and the information that must be agreed between different ECUs
- **System Configuration Extractor:**
  extracts the information from the System Configuration Description needed for a specific ECU
- **ECU extract:**
  is the information from the System Configuration Description needed for a specific ECU
- **ECU Configuration Description:**
  all information that is local to a specific ECU the runnable software can be built from this information and the code of the software component


Component API Generator
Component API (e.g. app.h)
SW-C Implementation
SW-Component Description
ECU Resource Description (HW only)
System – Constraint Description
AUTOSAR System Configuration Generator
Decisions (e.g. mapping)
System Configuration Description
ECU extract of System Configuration
ECU extract of System Configuration
AUTOSAR ECU Configuration Generator
Decisions (e.g. scheduling)
ECU Configuration Description
RTE extract of ECU configuration
OS extract of ECU configuration
e.g. OIL
Basic SW Module A extract of ECU configuration
List of implementations of SW components
AUTOSAR RTE Generator
OS, COM, ... Generator
Other Basic SW Generator
MCAL – Generator
Information / Database (no files)
Generation step: complex algorithm or engineering work
System
per ECU

AUTOSAR TOOLS

Implementer	BSW Implementation	BSW Configurator	RTE Generator	System To
ArcCore	Arctic Core –	BSW Builder	RTE Builder	SWCBuilde Builder
	CUBAS, iSolar [6]	CUBAS, iSolar [7]	CUBAS,iSolar [7]	Unknown
Continental	Yes	Yes	Yes	Yes
dSPACE	No	No	SystemDesk RTE Generator	SystemDesk
Elektrobit	EB tresos AutoCore	EB tresos Studio	EB tresos Studio	No
ETAS	Yes	Yes	RTA	ISOLAR-A
Freescale	Yes [8]	No	Yes [8]	Unknown
Dassault Systèmes	No	GCE	RTEG	AAT
KPIT Cummins	Yes	ECU Spectrum Toolchain	ECU Spectrum Toolchain	ECU Spectr Toolchain
Mecel	Yes	Yes	Yes	Unknown
Mentor Graphics	Volcano VSTAR	Volcano VSTAR	VolcanoVSTAR	Volcano Sys Architect
OpenSynergy	COQOS (OS & SchM)	COQOS	COQOS	No
Renesas Electronics	Yes	No	No	No
see4sys	Yes	Yes	Yes	ECU-Desig
Vector Informatik GmbH	MICROSAR	DaVinci Configurator Pro	MICROSARRte Generator	DaVinci Sys Architect

FAQs in AUTOSAR:

1. **What is AUTOSAR?**
2. **What is SWC?**
3. **Difference between Intra ECU and Inter ECU Communication?**
4. **What is meant by Client-Server Communication and Sender-Receiver Communication**
5. **What is meant by Communication Stack?**
6. **What is Pack and Unpacking IPdu?**
7. **What is MDT(Minimum Delay Timer)?**
8. **What is TMS (Transmission Mode Selection)?**
9. **Explain about AUTOSAR COM module?**

10. **What is RTE ? What are its function?**
11. **How the SWC interact with CAN module?**
12. **What is NM?**
13. **What are functions of CANSM, CANIF & CAN module?**
14. **Example of DET errors?**
15. **Example of DEM errors?**
16. **What is the functionality of DCM module**
17. **Explain the AUTOSAR architecture?**
18. **What are the pros & cons of AUTOSAR?**
19. **What is meant by Pre-Compile, Post-Build & Link Tme**

INCA V7.2 Service Pack 14 – What's New: Opening MDA V8.3.4 a...

//rcm-na.amazon-adsystem.com/e/cm?o=1&p=48&l=ur1&category=amzn_smp_ne_oscars&banner=06SBH2N77Y30S2PZEKG2&f=ifr&linkID=ee71df45b50c979b3bb14da3b292ff23&t=1103053-20&tracking_id=1103053-20

Advertisements

Advertisements
REPORT THIS AD

REPORT THIS AD

22 thoughts on “AUTOSAR BASICS”

1. **Eugene**

   says:
   January 19, 2015 at 10:39 pm
   Nice posting but I would recommend to update ETAS AUTOSAR Tools information in your “AUTOSAR TOOLS” table above to the latest one from this link: “http://www.etas.com/en/products/applications_autosar.php”

2. **sudhakar maradana**

   says:
   January 20, 2015 at 12:29 pm
   Thank you for the correction, updated the ETAS link in Autosar tools.

3. **egorbonosov@hotmail.com**

   says:
   January 20, 2015 at 5:22 pm
   Hello Sudhakar, Thanks for the updating the link but I also noticed that the whole line for ETAS in the AUTOSAR TOOLS table does not match a Wikipedia table. E.g., there is no mentioning of ISOLAR-A, etc. from the link I shared. Thanks, Eugene

   Date: Tue, 20 Jan 2015 12:29:05 +0000 To: egorbonosov@hotmail.com

4. **Sai Prasad**

   says:
   February 26, 2015 at 7:22 pm
   Nice collection of AUTOSAR concepts!

5. **natraj**

   says:
   July 29, 2015 at 5:30 am
   Hi

   Thank you for this tutorial ..i want to know workflow of AUTOSAR ..i have got some ideas from this tutorials..i need from the beginning to end which means oem to hexfile file..could you please do this ..

6. **sudhakar maradana**

   says:
   August 27, 2015 at 4:44 pm
   thats a nice idea.. thank you.. 🙂

7. **mona dash**

   says:
   November 22, 2015 at 7:11 pm
   thanks sudhakar for the tutorial…

8. **Maya Chaugule**

   says:
   February 2, 2016 at 6:17 pm
   good representation of information.

9. **whilda chaq**

   says:
   February 25, 2016 at 2:32 am
   This is a good starting point to begin undestanding the AUTOSAR. Did you already write about step by step for developing Software Component? and what is the important things to do it?

10. **Santosh**

    says:
    June 22, 2016 at 2:34 am
    Hi,
    Thank you very much for nice tutorial.
    Could you please explain more about AUTOSAR NM, by considering some project example (simple CAN network (topology)).
    1. How NM (CANNm) message loks like.
    2. Which functions are called by ComM.
    3. How to get NM mode and state, for application development.
    4. etc.. 🙂

11. **Kisshorkumar**

    says:
    August 6, 2016 at 8:45 pm
    Link me to brief description of comM module.

12. **Nimesh Salunkhe**

    says:
    August 30, 2016 at 2:21 pm
    Thanks Sudhakar for the complete overview of all the major steps in AUTOSAR.

13. **hanmnath reddy**

    says:
    October 18, 2016 at 6:48 am
    Thank you Maradhana for deatils.

14. **Dr.Nirmal**

    says:
    December 6, 2016 at 3:38 am
    Dear Sudhakar,
    Excellent Good Job.
    Thanks
    Dr.Nirmal
    Brazil/USA.

15. **Rashmi R**

    says:

December 8, 2016 at 11:25 am
nice info

16. **dhananjayshetty**

    says:
    December 21, 2016 at 8:34 am
    nice post …..Sudakarsimiler Automotive concept you can get in
    http://www.andthinker.com

17. **Sebastian**

    says:
    February 25, 2017 at 8:26 pm
    Thanks for this introduction *thumbsup*

18. **Pradeep**

    says:
    July 20, 2017 at 5:07 pm
    Hi. I have a query. Say if the CAN signal [raw signal] has interface CompuMethod with factor 0.01. In the Application we need to use the physical value. Does the Application SwC handle the CAN to physical value conversion, or will RTE handle the conversion.

19. **Umair**

    says:
    November 17, 2017 at 9:15 am
    Hi Pradeep,
    SWC will handle the CAN to physical value conversion. We need to handle this conversion by writing some function in CAPL for example in CANoe to handle. The converted value will then be transferred by RTE to other SWC.

20. **Sahil**

    says:
    March 6, 2018 at 4:57 pm
    Thanks for sharing the knowledge well framed and informative 🙂

21. **Irene Hynes**

    says:
    May 7, 2018 at 7:07 am
    Hi Buddie,

    Your writing shines! There is no room for gibberish here clearly you have explained about AUTOSAR BASICS. Keep writing!

    Write a LEX program (i know this is in C) that identifies words from the previous set of phrases, such that an input of the form "triangle BCD" returns:
    —Triangle: a geometric entity (lol at that)
    —BCD: name of a geometric entity
    b) Define the tokens, lexemes and patterns to be found in formal expressions concerning these three sentences.
    c) Construct the finite automaton that corresponds to the formal expressios to describe the names of geometric entities in that language. Use this to see if the name AYZ is recognizable.

But nice Article Mate! Great Information! Keep up the good work!

Cheers,
Irene Hynes

22. **Kevin Lee**

    says:
    May 19, 2018 at 8:36 am
    Hallo,

    Such vivid info on the Autosar Basics! Flabbergasted! Thank you for making the read a smooth sail!

    I want to learn Linux.
    I have an idea for an application that I want to be open source.
    I fairly technical (Cisco CCNP) and I'm good with tshooting but I don't want to waste hours of my time narrowing down the root cause of some insignificant thing. I would rather pay someone else to give me the answer (I understand Red Hat has technical support I can pay for).
    These virtual files have unique qualities. Most of them are listed as zero bytes in size. Virtual files such as /proc/interrupts, /proc/meminfo, /proc/mounts, and /proc/partitions provide an up-to-the-moment glimpse of the system's hardware.
    Please keep providing such valuable information.

    Shukran,
    Kevin

BLOG AT WORDPRESS.COM.

UP ↑