Introduction to AUTOSAR
“AUTomotive Open System Architecture”

Tarek Kabbani

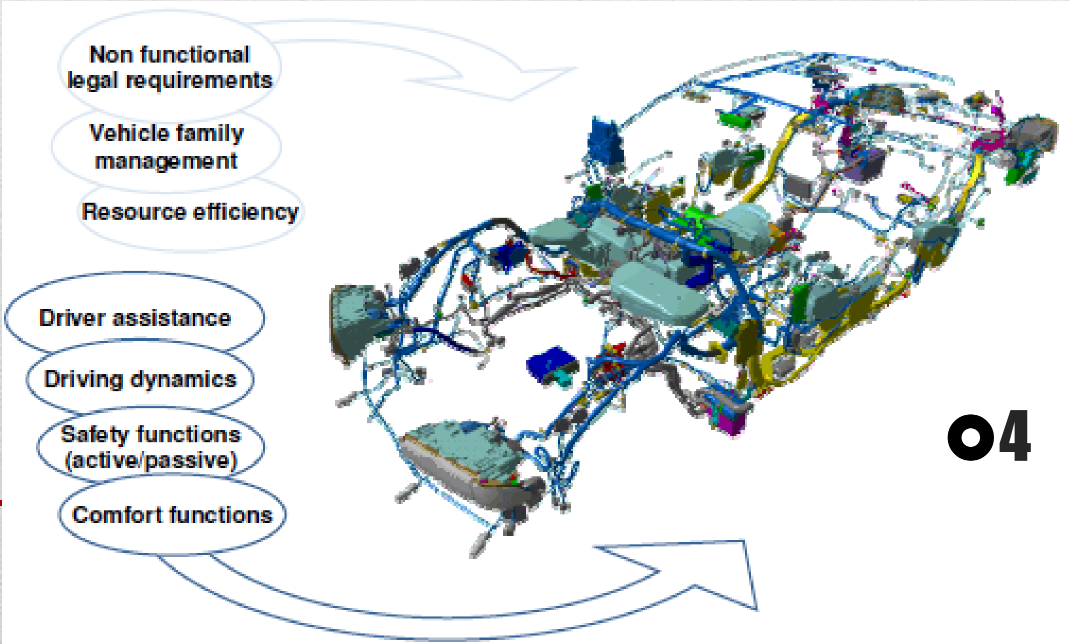
Contents

Contents

What is AUTOSAR

__AUTOSAR (AUTomotive Open System ARchitecture)__

- **Middle-ware and system-level standard,**
- **Jointly-developed** by automobile manufacturers, electronics and software suppliers and tool vendors and developers.
- More than **100 member** companies
- Homepage : **www.autosar.org**
- Motto: **"Cooperate on standards – compete on implementation"**



What is AUTOSAR - History

Automotive domain

<u>OEM request the reduce of ECU Number</u> for standard Functions

- Engine Control
- Breaking Control
- Body Computer
- Automatic Transmission
- Gear selector
- Steering Column
- Steering Wheel
- Steering angle sensor
- Electric power steering
- Parking Sensor
- Dashboard
- Adaptive Cruise Control
- Tyre Pressure control
- Air Bag unit Control
- Driver Door
- Passenger Door
- Passive Entry
- Driver seat ECU Trunk
- Passenger seat
- Left Headlight unit Control
- Right Headlight control
- Air Conditioning
- Alarm Sensor/roof control
- Rain/sun sensor
- External light control
- Alarm unit control
- Infotainment

<u>Leave room for innovation</u>: NEW functions are first implemented as new ECUs

Driving Mechanics
- 2WD/4WD control
- 4-wheel steering
- Power steering
- Brake by wire
- Collision warning
- Suspension control
- Distribution systems (Gateway)

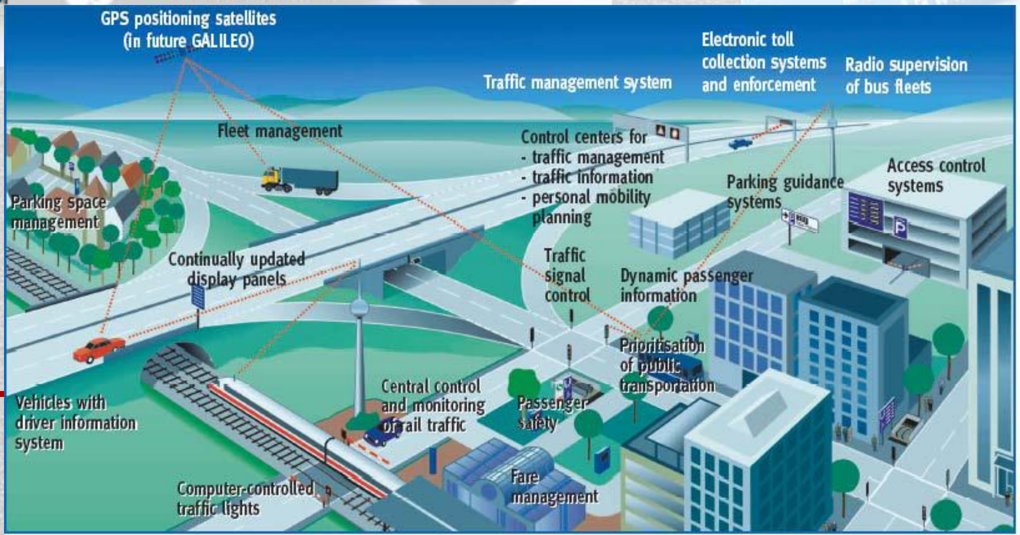
Driving Electrical
- ISAD alternator
- Start and Stop
- Rear lamp clusters - LED
- Side-direction indicators - LED

Driving Comfort
- Memory seat, mirror, steer Positions
- Electronic compass
- Electro chromic mirrors
- Power roof
- Electric sunroof
- Off driving lines warning
- Parking helps

Driving Infotainment
- Integrated cell phone
- Traffic information system
- GPS

Driving Safety
- Vision enhancement
- Exterior camera
- Interior camera

That will later be ported to software



Courtesy Siemens AG

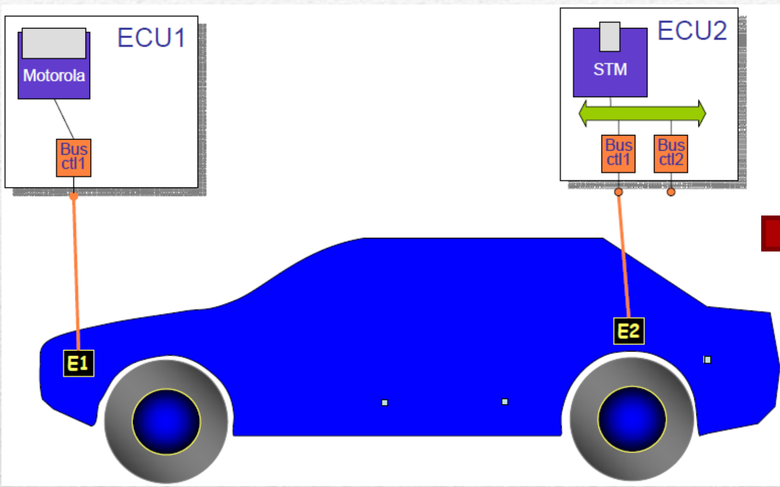
<u>And Prepare the Future:</u> Toward partially open Auto-motive EE Architecture

OEM: Original Equipment Manufacturers
ECUs: Electronic Control Units
EE : Electronic and Embedded

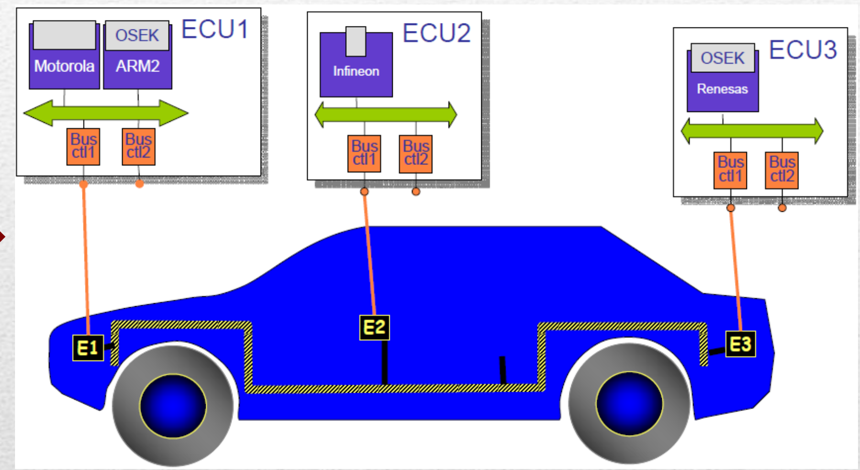
What is AUTOSAR - History

History of ECU in Vehicles

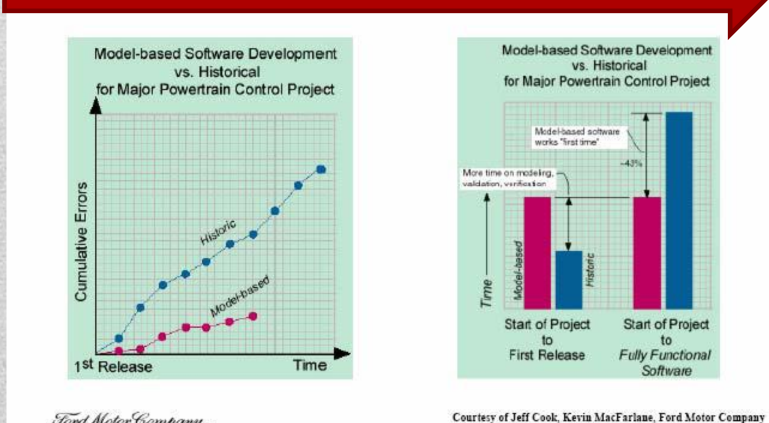
1990 - Designing ECU's

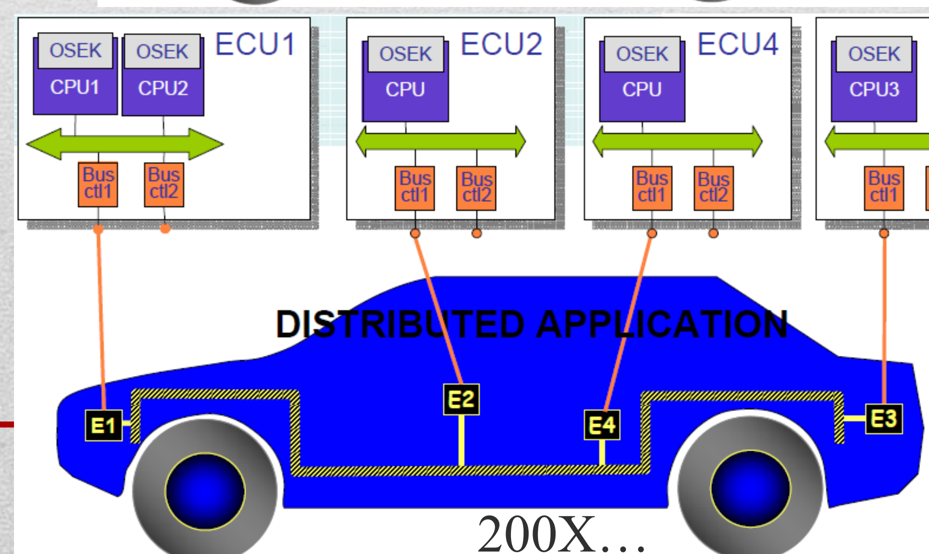


1995 : Defining Network and Messaging Systems of Today



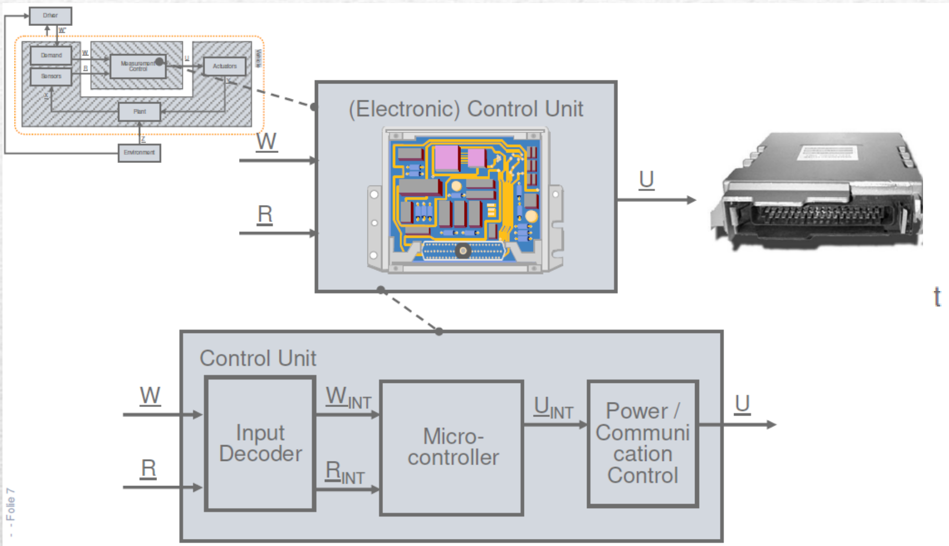
2000 – Model-based design

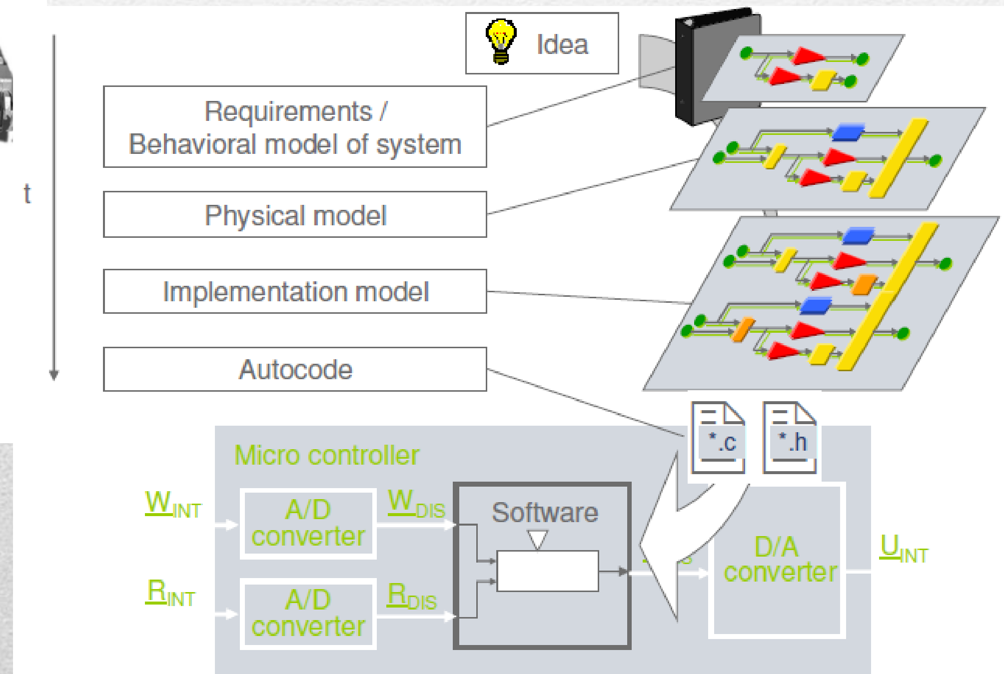




What is AUTOSAR - History

- Model-based Development/General Task/Evolution of Models



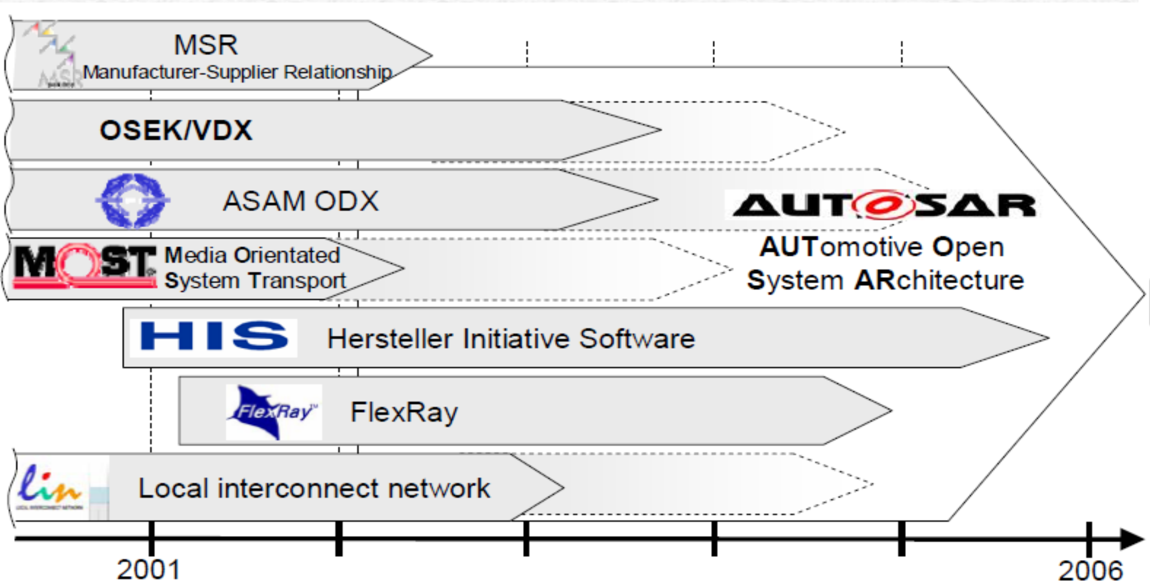


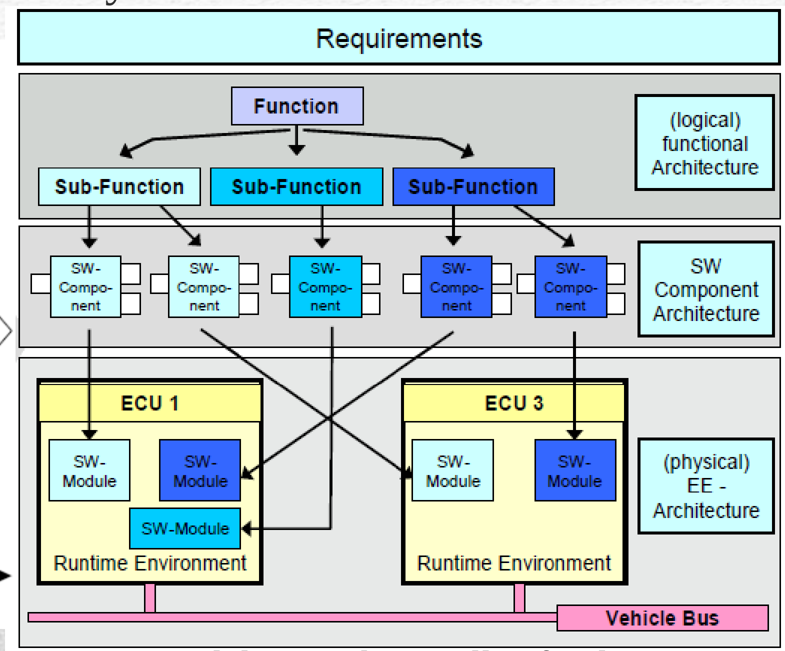
SAE Embedded Software standardization efforts

- ❑ **Just Beginning** – April 13th Open Discussion – Model-based Development and Autocode Generation – Is There a Need for Automotive Industry Standardization?

What is AUTOSAR - History

- AUTOSAR integrates existing and emerging industry electronics standards.





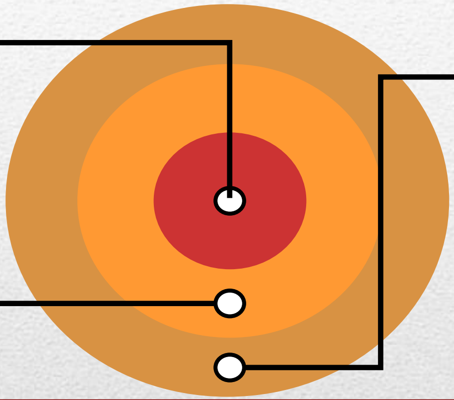
- The AutoSar OS is a derivative of OSEK "Offene Systeme und deren Schnittstellen für die Elektronik in Kraftfahrzeugen" (Open Systems and their Interfaces for the Electronics in Motor Vehicles) **Responsible for** real time functions, priority based scheduling and different protective functions.
- Some **systems will continue to use their own OS** but these must have AutoSar interface (follows AutoSar specifications).

What is AUTOSAR - Story

- AutoSar **partnership**

9 Core Partners

BMW
DaimlerChrysler
BOSCH
VOLKSWAGEN AG
OPEL
A General Motors Company
Continental
PSA PEUGEOT CITROËN
SIEMENS VDO Automotive
TOYOTA
Ford



112 Associate Members

Johnson Controls
WIND RIVER
ITRI
auto-com
MANDO
preh
Dräxlmaier

43 Premium Members

Hyundai · Kia Motors
HONDA
mazda
NISSAN
Renault
PORSCHE
VOLVO
FIAT

DELPHI
Autoliv
DENSO
Magneti Marelli
LEAR CORPORATION
HELLA Ideas today for the cars of tomorrow
Valeo
Visteon
ZF

ALPINE Car Audio and Navigation Systems
3SOFT
micron Electronic Devices AG
DECOMSYS Your FlexRay Development Partner
IBM
LiveDevices ETAS Group
tni SOFTWARE
T-Systems
Volcano Automotive Group
vector
FEV
dSPACE
ETAS
softing
The MathWorks
Infosys

freescale semiconductor Launched by Motorola
Infineon technologies
ST
NEC
HITACHI
RENESAS Everywhere you imagine

OEM	Tier 1	Standard Software	Tools	Semi-conductors

Source: AUTOSAR

28 Development Members
18 Attendees

What is AUTOSAR - Story

- AutoSar **Timeline**

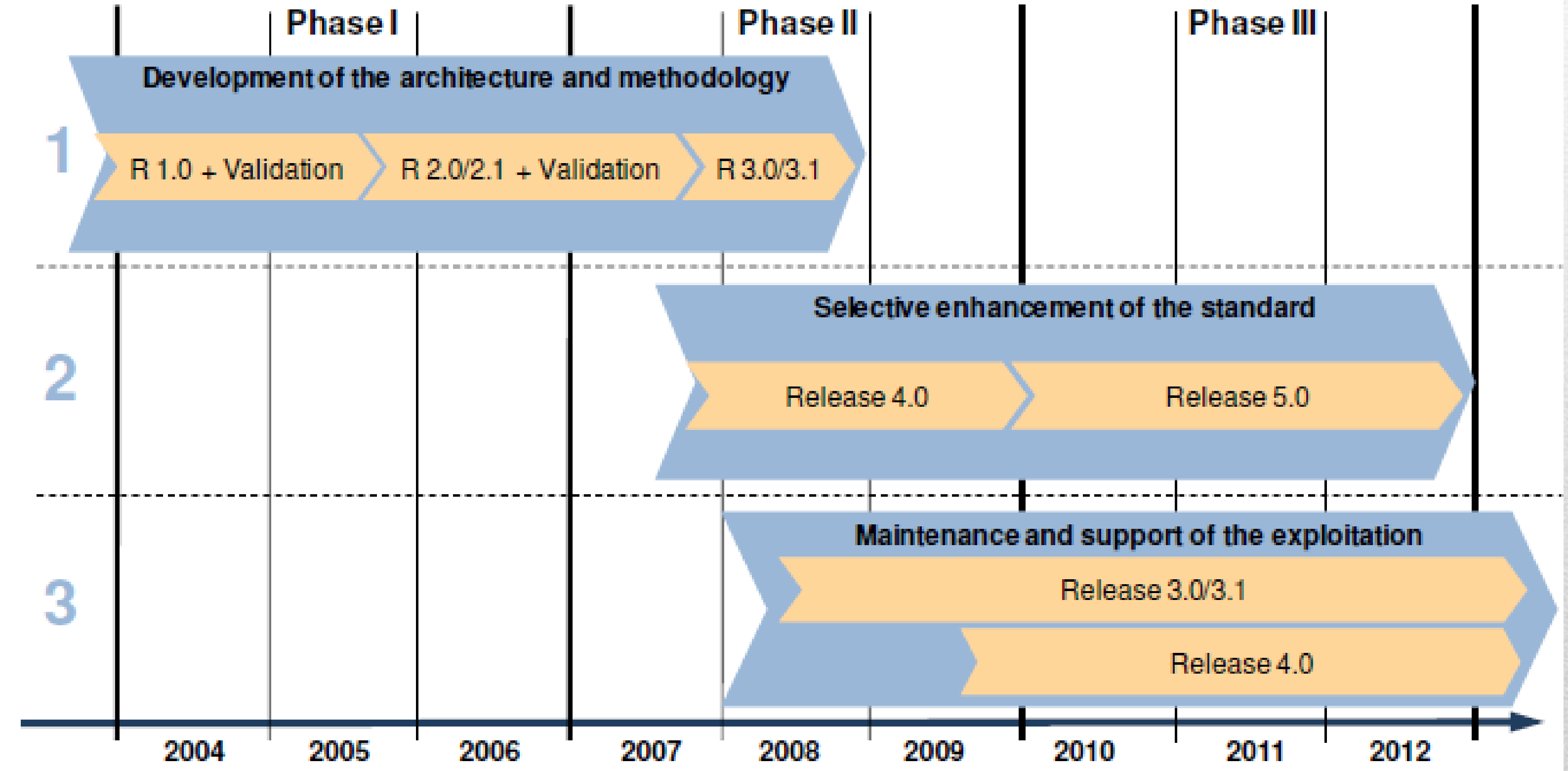


Figure 1: The AUTOSAR Timeline

Main Concepts: Architecture

Application scope of AUTOSAR

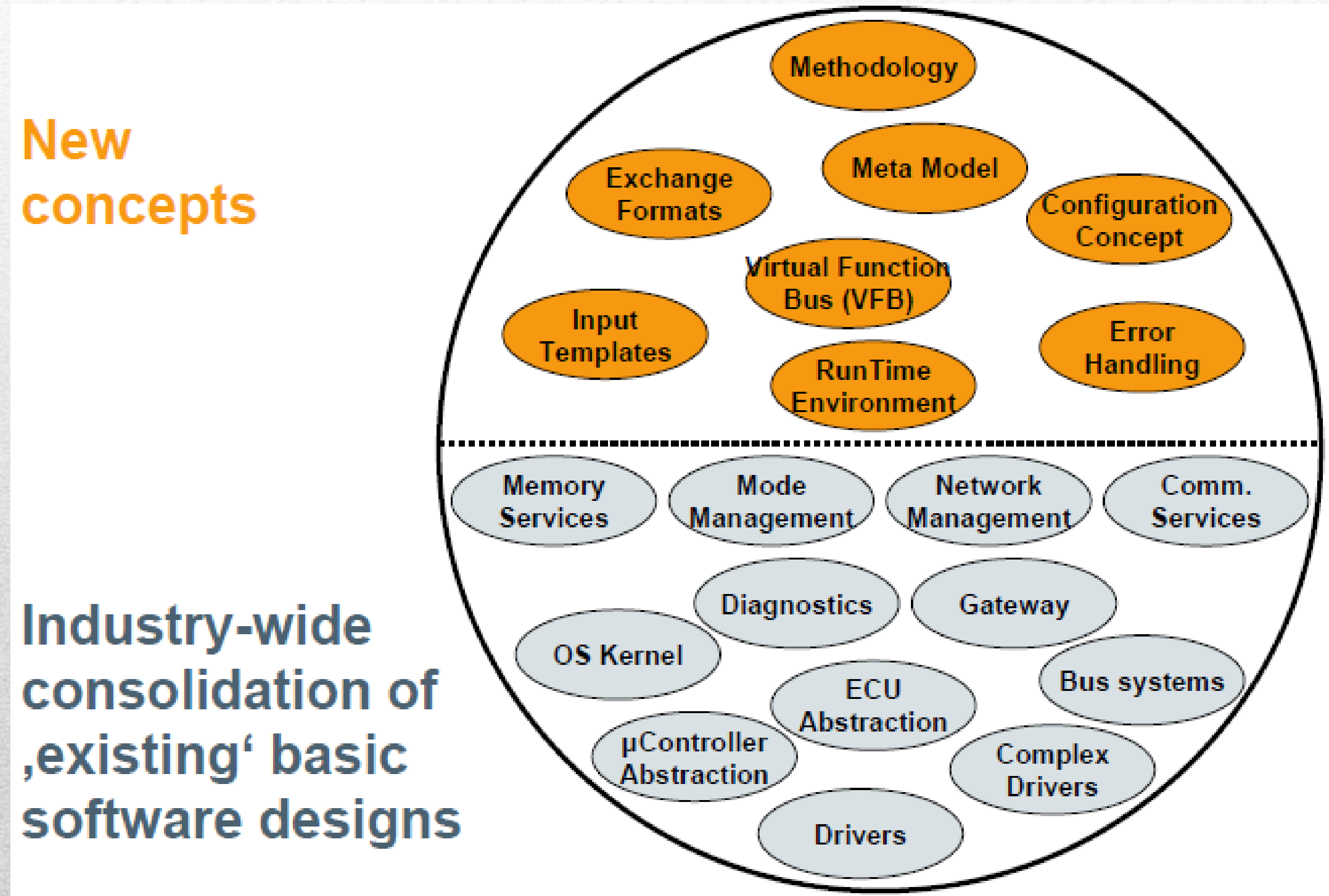
AUTOSAR is dedicated for **Automotive ECUs**. Such ECUs have the following **properties**:

- Strong interaction with hardware (sensors and actuators),
- Connection to vehicle networks like CAN, LIN, FlexRay or Ethernet,
- Microcontrollers (typically 16 or 32 bit) with limited resources of computing power and memory (compared with enterprise solutions),
- Real Time System and
- Program execution from internal or external flash memory.

NOTE: In the AUTOSAR sense an **ECU means** one microcontroller plus peripherals and the according software/configuration. The **mechanical design is not in the scope of AUTOSAR**. This means that if more than one microcontroller in arranged in a housing, then each microcontroller requires its own description of an AUTOSAR-ECU instance.

What is AUTOSAR - Story

- **Technical scope** of AUTOSAR

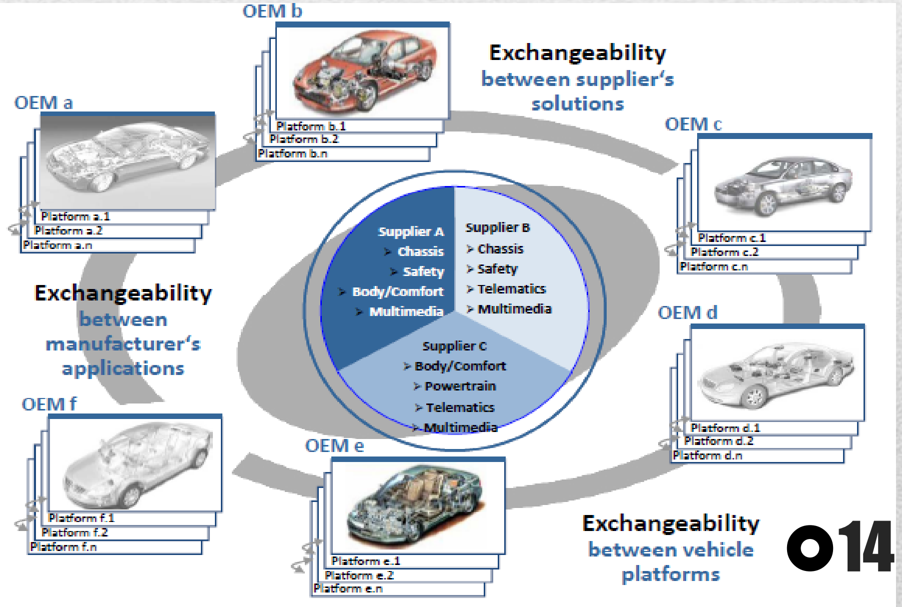


Contents

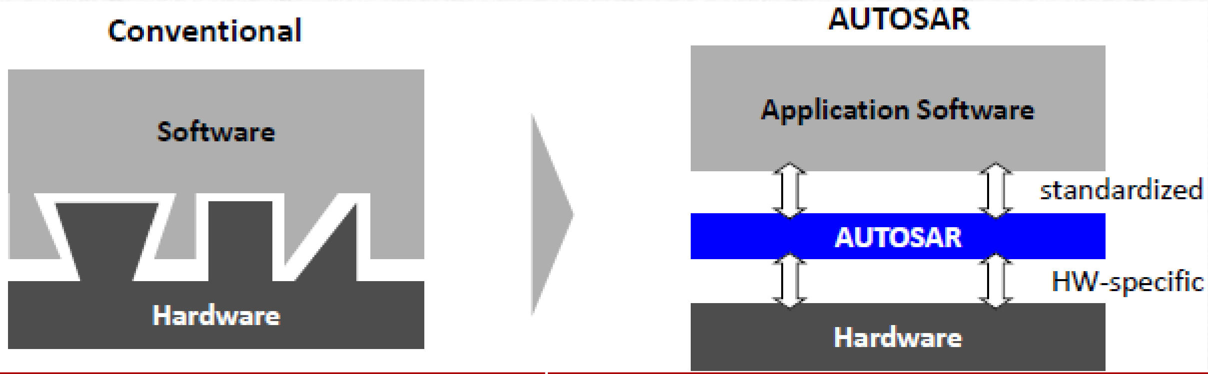
- What is AUTOSAR
- Project Objectives & Benefits
- Use case “ Front-Light Management”
- AUTOSAR Main Concepts
  - Architecture
  - Methodology
  - Application Interfaces
- Example of AUTOSAR System
- Conclusion

Project Objectives & Benefits

- AUTOSAR **objective** is to establish an open reference industry standard for the automotive ECU software architecture.
- The standard **comprises a set of specifications** describing a software layered architecture (implemented in every ECU and frees the applications software from the hardware components) and defining their interfaces.
- The **principal aim** of AUTOSAR is to master the growing complexity and improve its management of highly integrated automotive E/E architectures between suppliers and manufacturers.



Project Objectives & Benefits



How were vehicle functions implemented usually?

- Each function had it´s own ***system and microcontroller*** although they may communicate through ***a bus***.
- The ***number of ECU´s*** (Electronic Control Unit) were ***growing fast***.

- Hardware and software were ***tightly connected***.
- The same ***vendor supplies both*** hardware and 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.
- The ***same Standardized uniquely-specified interface*** for all basic software.
- Functionality is supplied as ***software components.***

- These components are hardware ***independent***.
- The software modules are ***exchangeable and reusable*** between OEM and suppliers.
- More than one ***supplier can compete*** with their software.

Project Objectives & Benefits

Hardware and software will be **widely independent** of each other

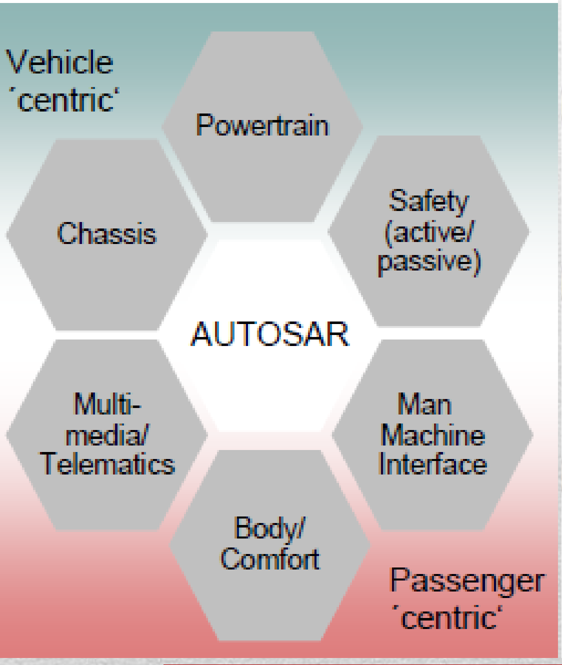
- **Development processes will be simplified**, as designer just picks software components without knowing in what ECU they will be implemented (not think in terms of ECUs when designing the system).
- **Allow software Reuse of software increases** at OEM and at suppliers and **smooth evolutions** (limiting re-development and validation), this reduces development time and costs.
- This **enhances** also **software quality and efficiency** and **easily handling of electronic** in the automobile.
- **Facilitate portability, composability, integration** of SW components over the life time of the vehicle.

Automotive Software will become a product.

Project Objectives & Benefits

The <u>specification of functional interfaces</u> is divided into **6 domains**:

Functional Domains

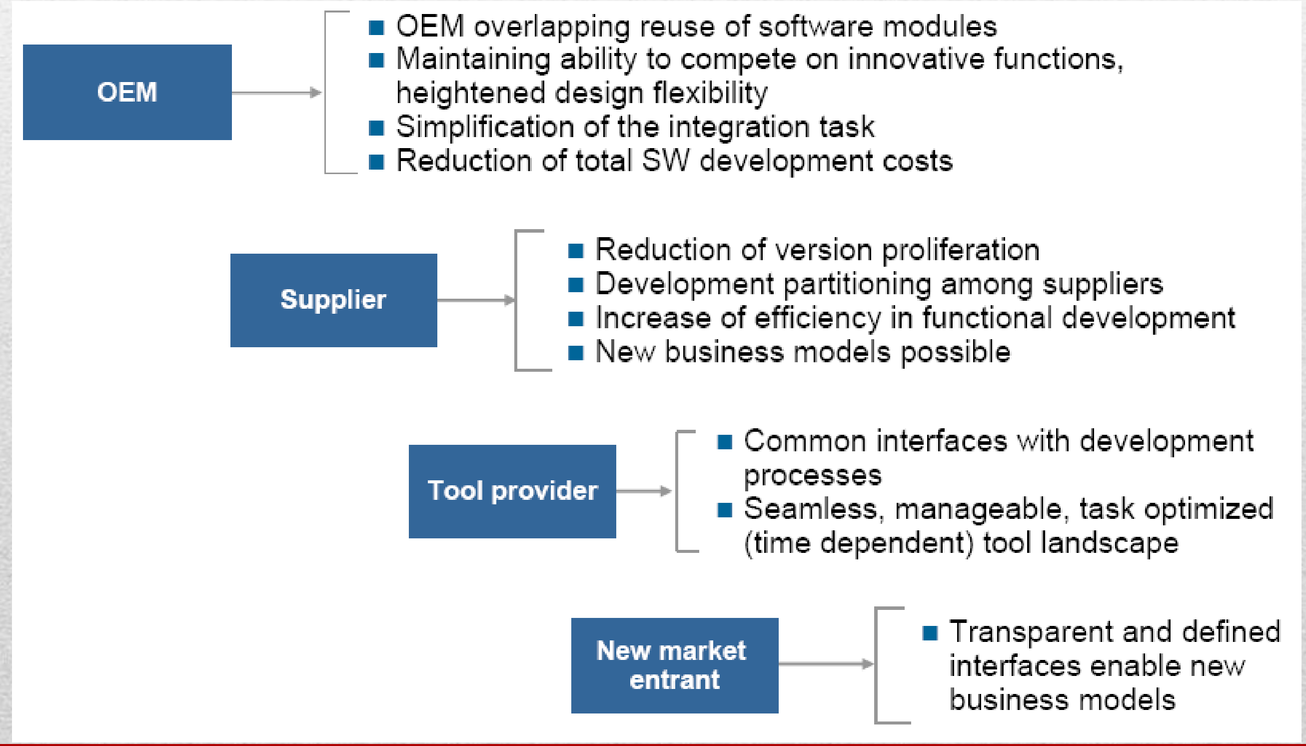


The project **goals will be met** by <u>standardizing the central architectural elements across functional domains</u>, allowing industry competition to focus on implementation

- 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

Project Objectives & Benefits

- The **ongoing development** of AUTOSAR products by the member companies provides a <u>unique feedback loop</u> into the development of the standard itself. This allows **fast and pragmatic improvements.**
- The previous benefits **are for all type of stockholders.**



Project Objectives & Benefits

AUTOSAR increases the value of non-competitive software

- ❑ Automotive embedded software must handle both the application side (feature content) and the non-application infrastructure side (networking, operating system, etc.)
- ❑ Software complexity can be reduced by standardization – especially if concentrated on the non-application infrastructure side

Standardizing a software architecture – one that encompasses:

- ❑ common vehicle network solutions
- ❑ common use of operating systems
- ❑ common use of generic I/O

can decrease development time and increase the value of model-based development

AUTOSAR – Increases Software Standardization

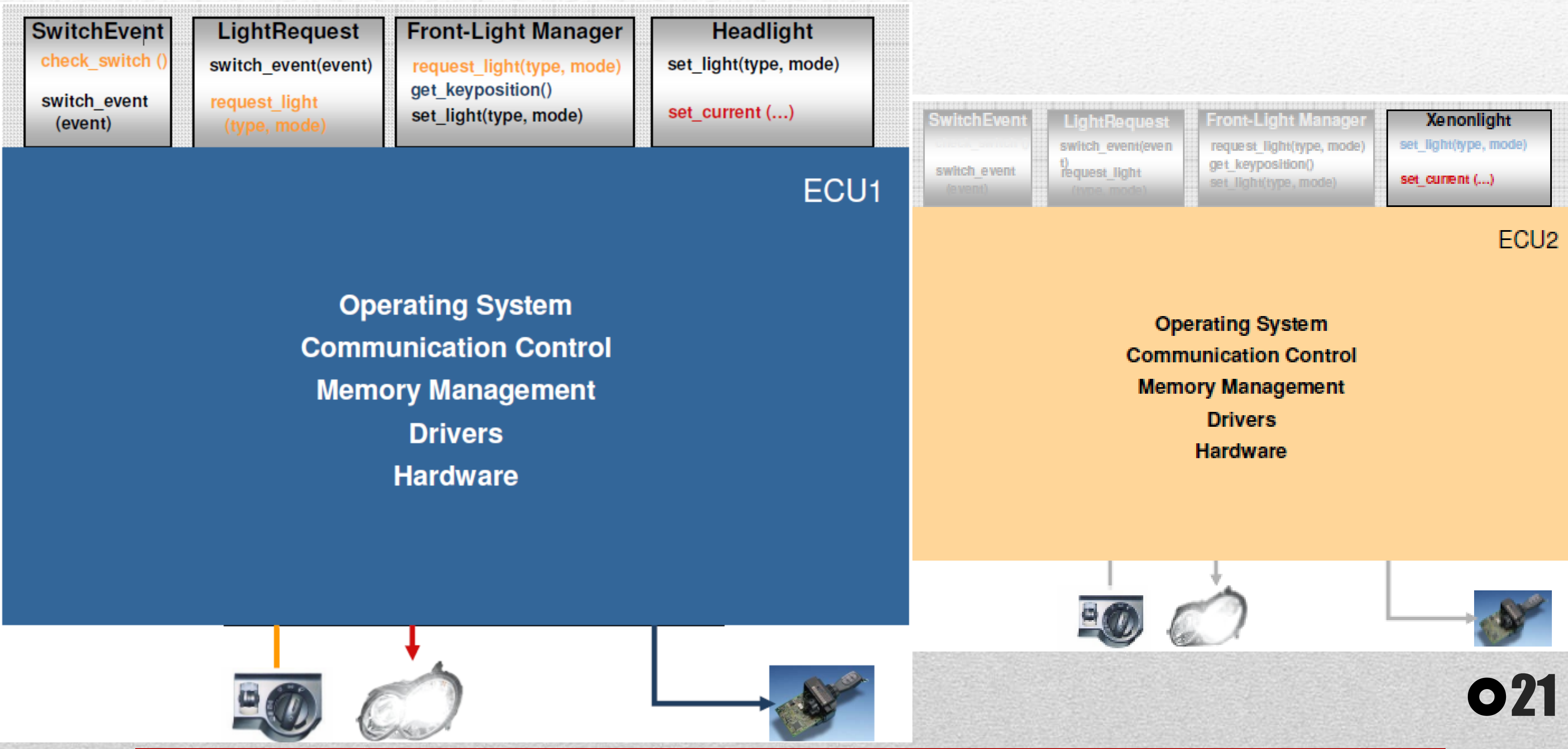
- ❑ AUTOSAR standardizes only the non-application infrastructure side of automotive embedded software

Contents

- What is AUTOSAR
- Project Objectives & Benefits
- Use case “ Front-Light Management”
- AUTOSAR Main Concepts
  - Architecture
  - Methodology
  - Application Interfaces
- Example of AUTOSAR System
- Conclusion

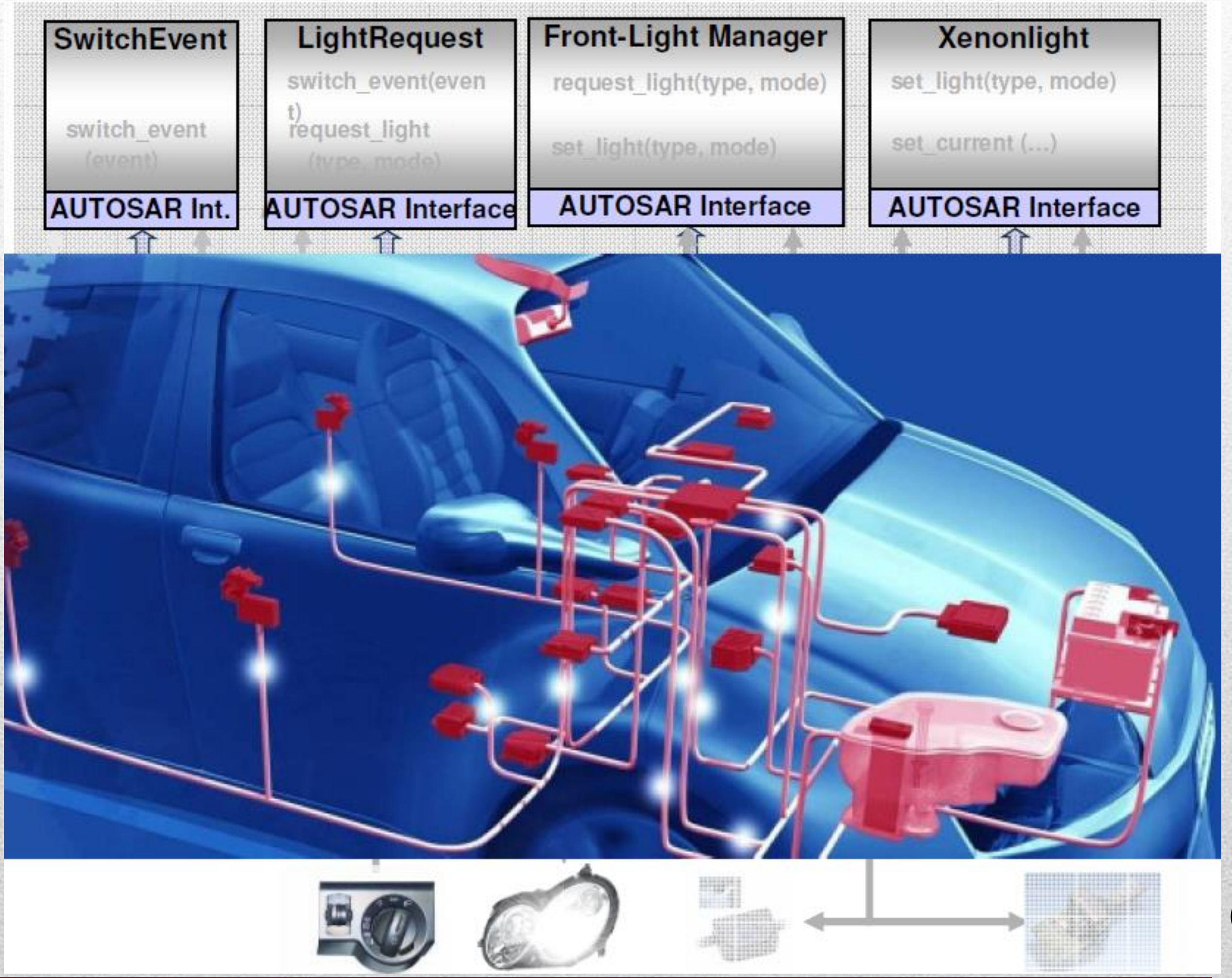
Use case “ Front-Light Management”

- Exchange of type of front-light



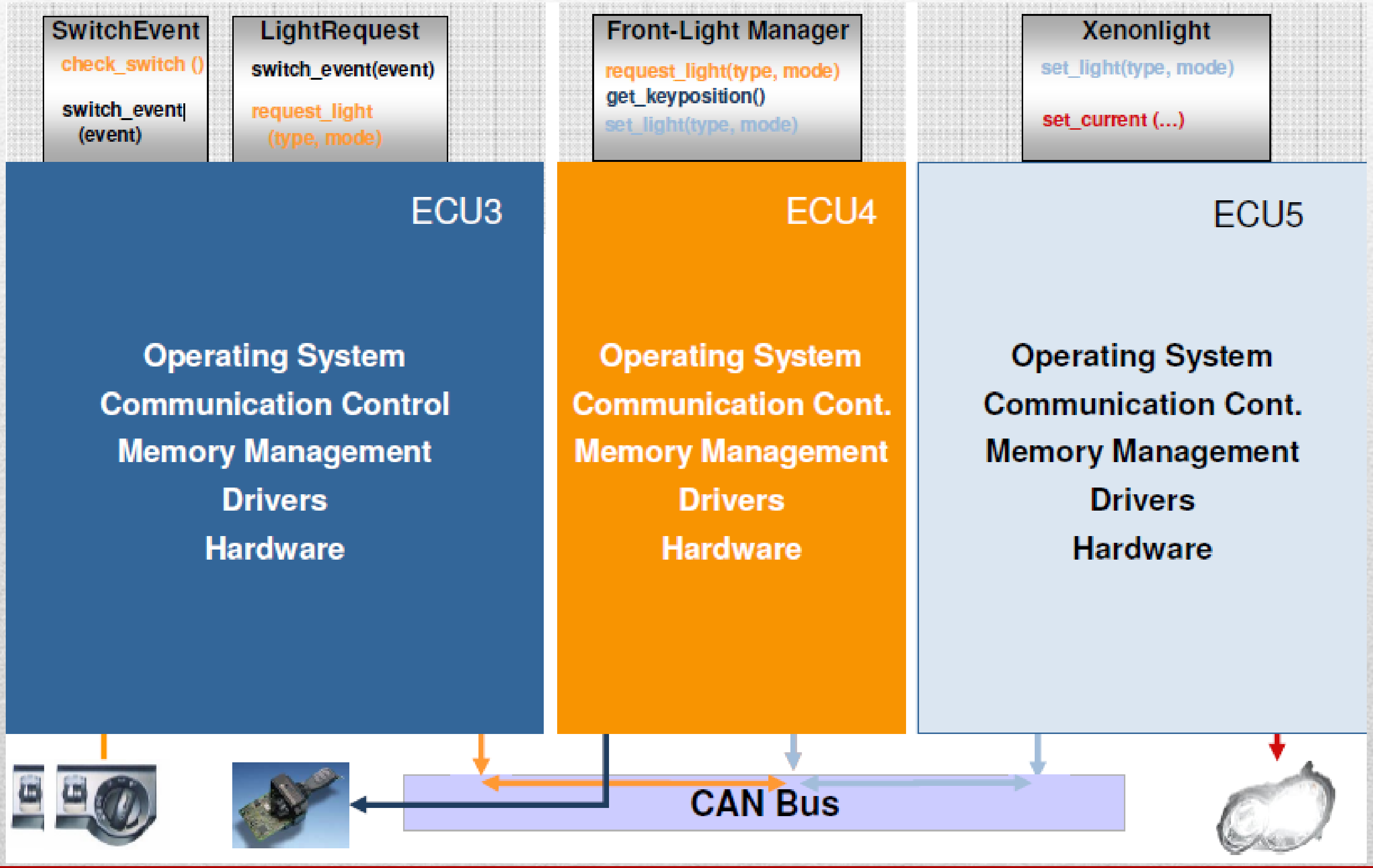
Use case " Front-Light Management"

- Distribution on ECUs



Use case " Front-Light Management"

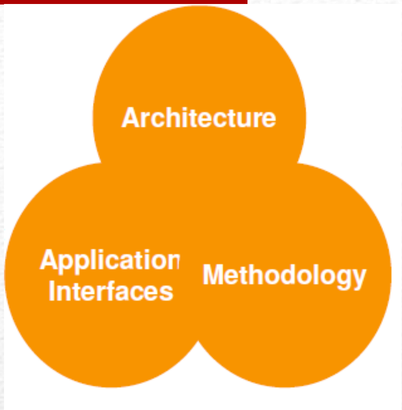
- Multiple ECUs



Contents

- What is AUTOSAR
- Project Objectives & Benefits
- Use case “ Front-Light Management”
- AUTOSAR Main Concepts
  - Architecture
  - Methodology
  - Application Interfaces
- Example of AUTOSAR System
- Conclusion

AUTOSAR Main Concepts



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 integration of application software in ECUs and it includes even the methodology how to **use this framework.**

❖ ***Application Interfaces:***

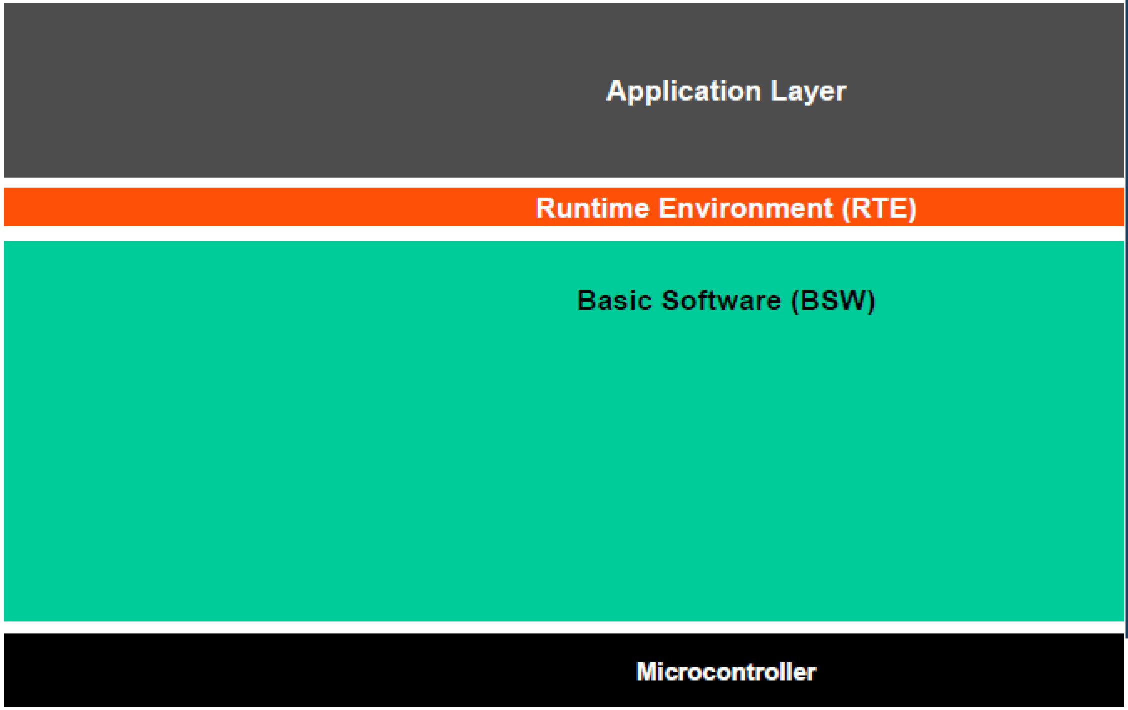
Specification of interfaces of typical automotive applications from all domains in terms of syntax and semantics, which should serve as a standard for application software modules.

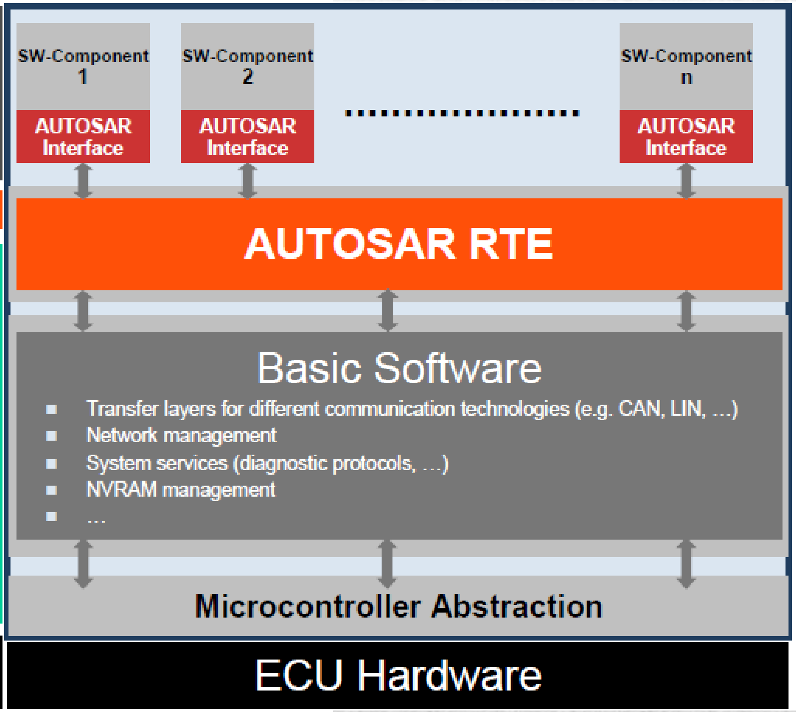
Main Concepts: Architecture

Main Concepts: Architecture

- The AUTOSAR Layered Architecture distinguishes on the **highest abstraction level** between three software layers:

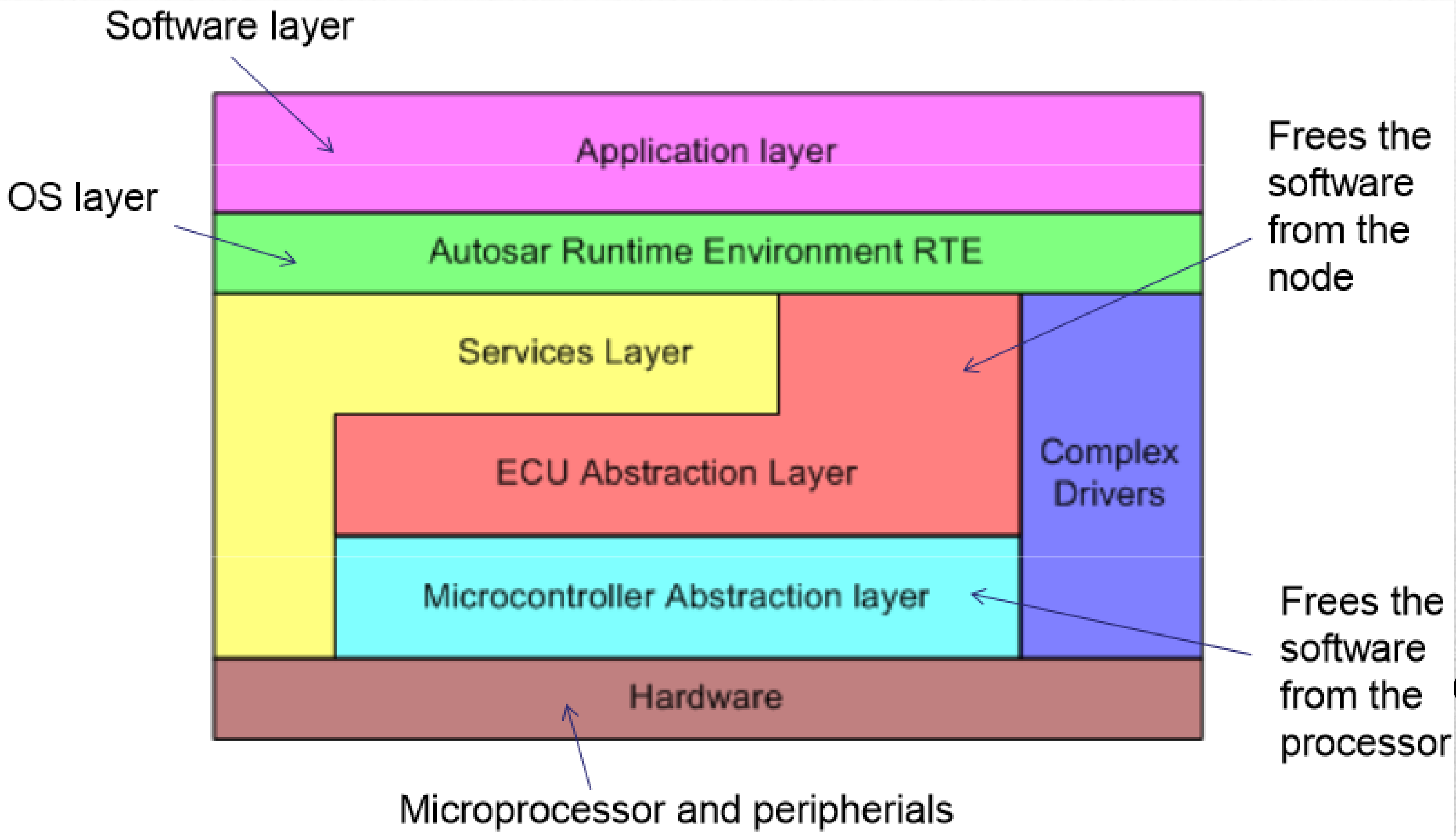
  Application, Runtime Environment and Basic Software which run on a Microcontroller.



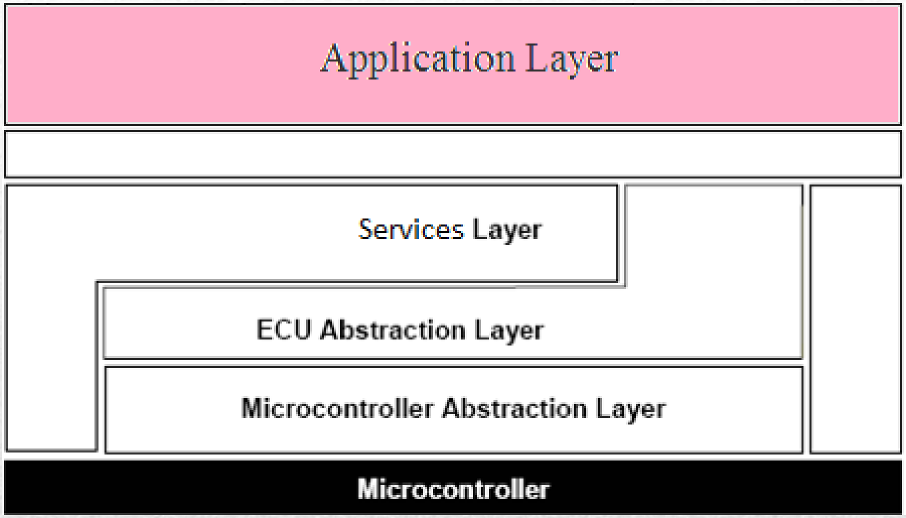


Main Concepts: Architecture

- The AUTOSAR Basic Software is divided in the layers: Services, ECU Abstraction, Microcontroller Abstraction and Complex Drivers.

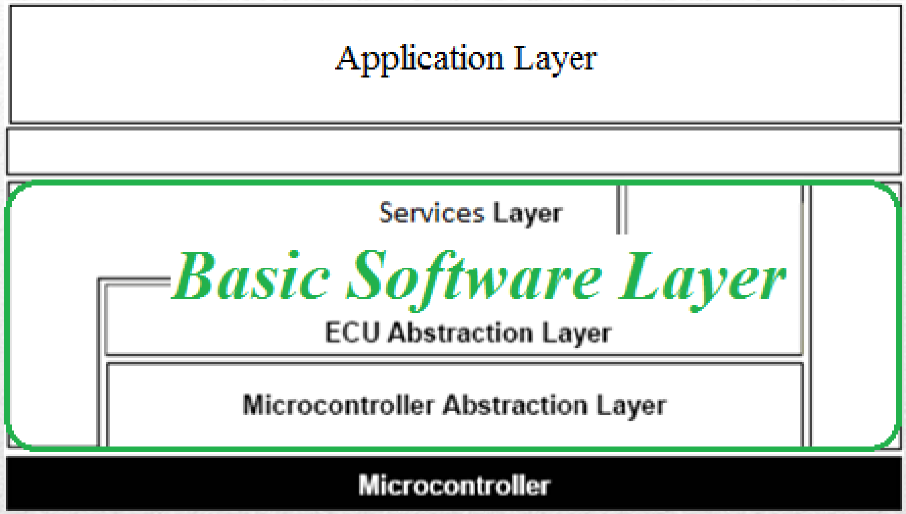


Main Concepts: Architecture



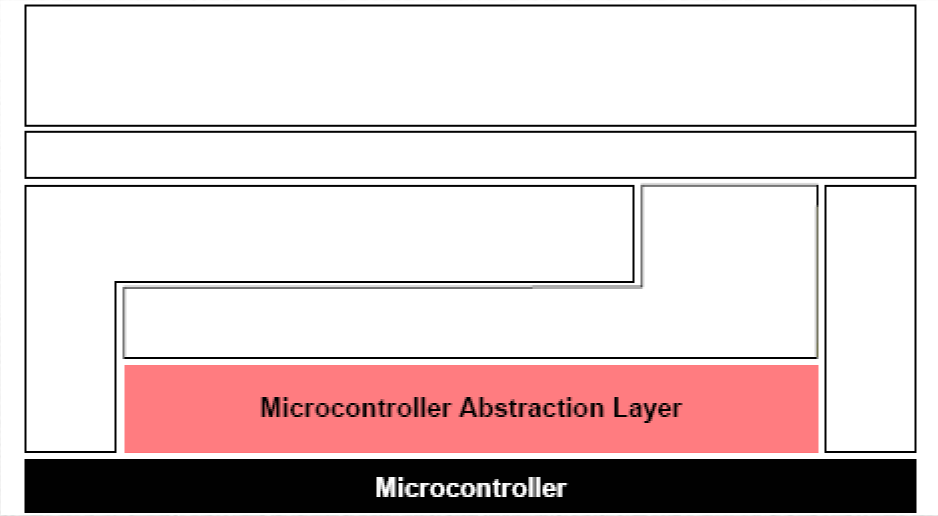
- **The applications functionality** reside in the application layer.
- The only part of an AutoSar system that **doesn’t consist of standardized software.**
- **Consists of SWCs** (Software Components), the smallest part of a software application that has a specific functionality.
- Within AutoSar there are standard interfaces so that the components can be used to build out a software applications.

Main Concepts: Architecture



- It is a standardized software without any own functionality that **offers both hardware dependent and independent services** to higher layers. This is through API (Application Programming Interfaces).
- It **makes** the higher layers hardware independent.
- **It has**, for e.g., memory interfaces and interfaces to communication busses (LIN, CAN and FlexRay)
- Can be **subdivided into** the following **types of services**:
  - **Input/Output (I/O) Standardized access** to sensors, actuators and ECU onboard peripherals
  - **Memory Standardized access** to internal/external memory (non volatile memory)
  - **Communication Standardized access** to: vehicle network systems, ECU onboard communication systems and ECU internal SW
  - **System** Provision of standardizeable (operating system, timers, error memory) and ECU specific (ECU state management, watchdog manager) services and library functions

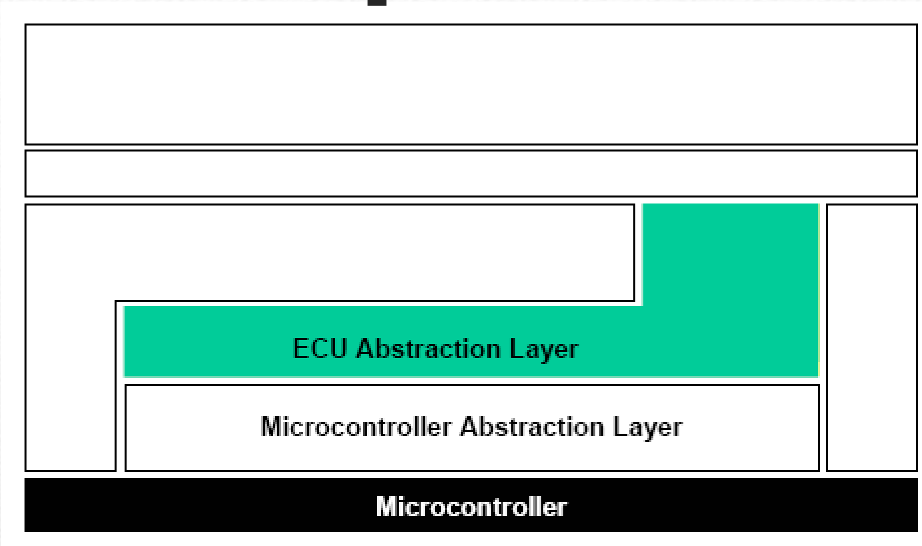
Main Concepts: Architecture



MCAL (Microcontroller Abstraction Layer)

- MCAL is the **lowest software layer of the Basic Software**
- **It contains internal drivers**, which are software modules with direct access to the µC internal peripherals and memory mapped **µC external devices**.
- **Consists of standardized functions** that frees the hardware from the software and gives a standardized interface to Basic Software, making higher software layers processor independent of abstracted µC and prevented from directly accessing the registers in the µC.
- MCAL **handles** the µC **peripheral units** and **supplies processor-independent values** to the Basic Software.

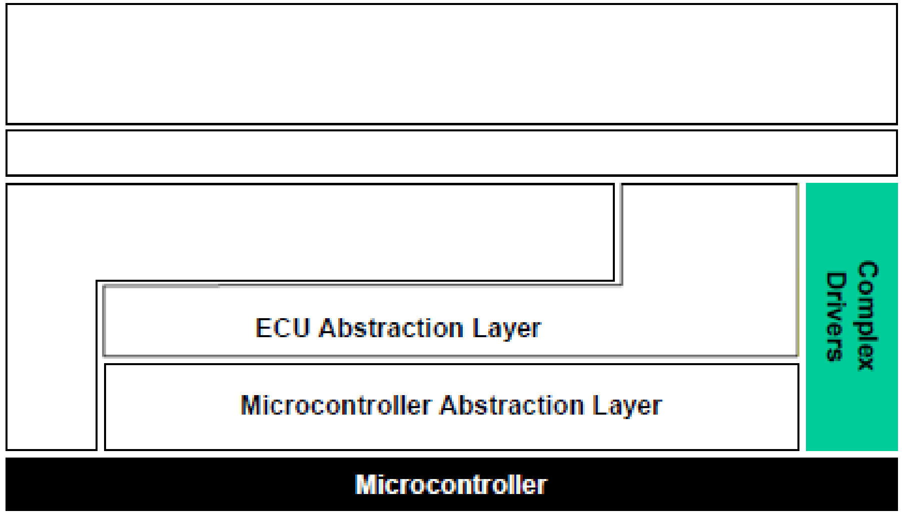
Main Concepts: Architecture



EAL (ECU Abstraction Layer)

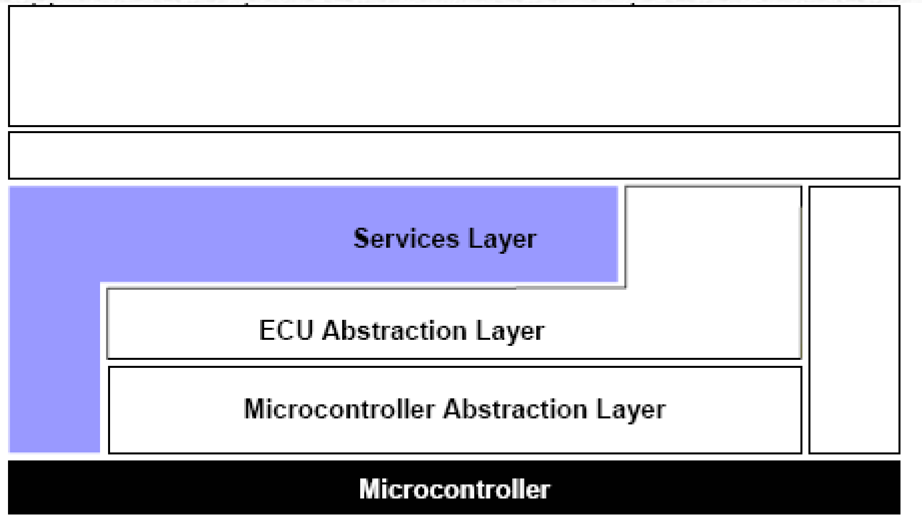
- The EAL **interfaces the drivers of the MACL** and also **contains drivers for external devices**.
- It **offers an API** for access to peripherals and devices regardless of their location and their connection to the microcontroller (port pins, type of interface)
- **Make** higher layers independent of ECU hardware layout
- Its implementation **properties**, is µC independent and ECU hardware dependent. For **Upper Interface**, it's µC and ECU hardware independent

Main Concepts: Architecture



- The **Complex Drivers Layer** spans from the hardware to the RTE.
- **Provide the possibility to integrate special purpose functionality**, e.g. drivers for devices:
  - which are not specified within AUTOSAR,
  - with very high timing constrains or
  - for migration purposes etc.
- Its implementation and Upper Interface **properties are:** it might be an application, it's µC and ECU hardware dependent.

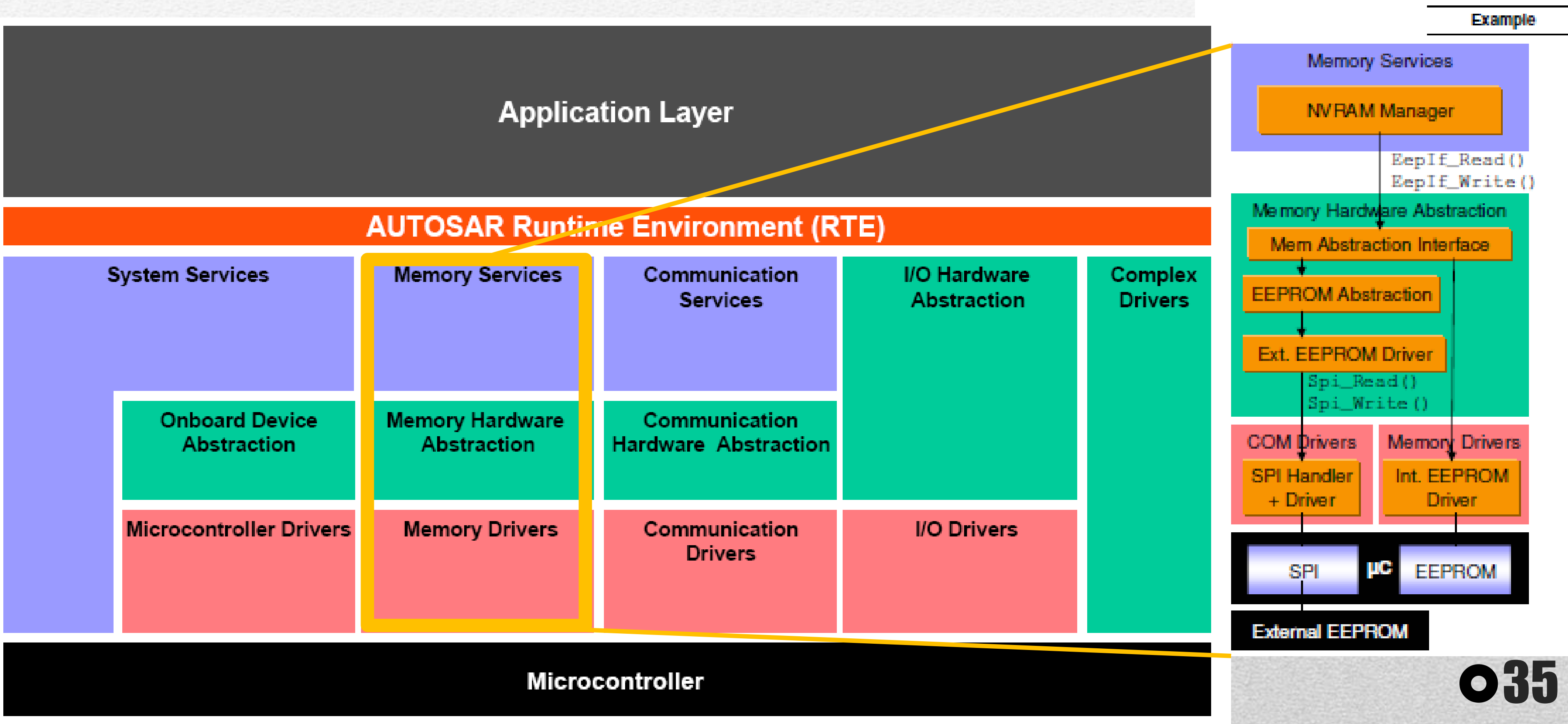
Main Concepts: Architecture



- 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, **it offers basic services** for application and Basic Software modules as:
  - Operating system functionality
  - Vehicle network communication and management services
  - Memory services (NVRAM management), ECU state management
  - Diagnostic services (including UDS communication, error memory and fault treatment)
  - Logical and temporal program flow monitoring (Wdg manager)
- Implementation Properties: mostly µC and ECU hardware independent. Upper Interface: µC and ECU hardware independent

Main Concepts: Architecture

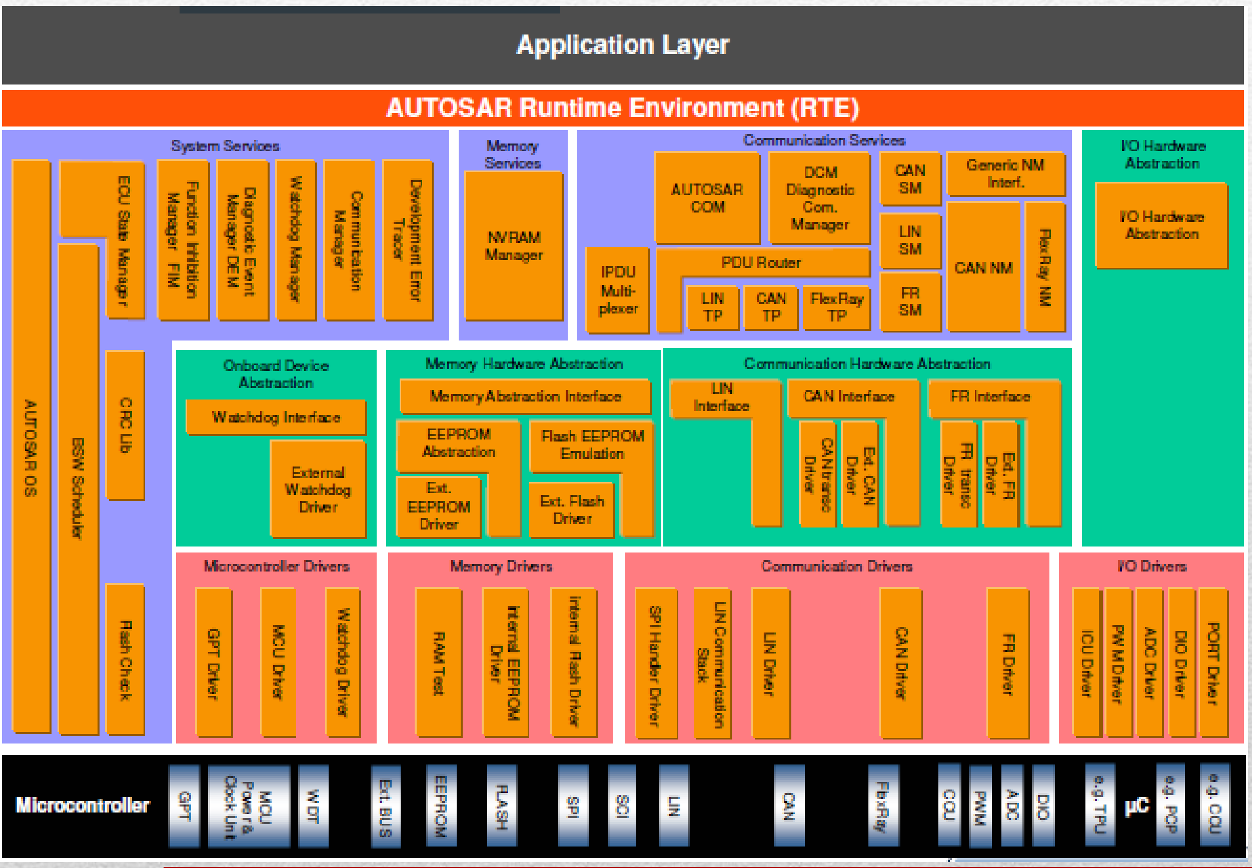
- The Basic Software Layers are further divided into functional groups (11 main blocks plus Complex Drivers). Examples of Services are System, Memory and Communication Services.



Example of Flow through layers: “NVRAM Manager” **ensures the storage and maintenance of non-volatile data** and is independent of the design of the ECU.

Main Concepts: Architecture

- The layered architecture has been further **refined in the area of Basic Software** and around **80 modules** have been defined.

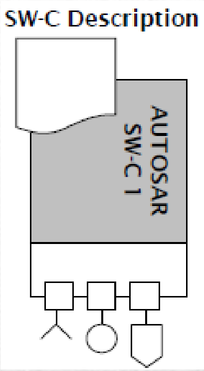


AUTOSAR extensibility

The AUTOSAR Software Architecture is a generic approach:

- **Standard modules** can be extended in functionality.
- **Non-standard modules** can be integrated into AUTOSAR-based systems as Complex Drivers
- **Further layers** cannot be added.

Main Concepts: Architecture



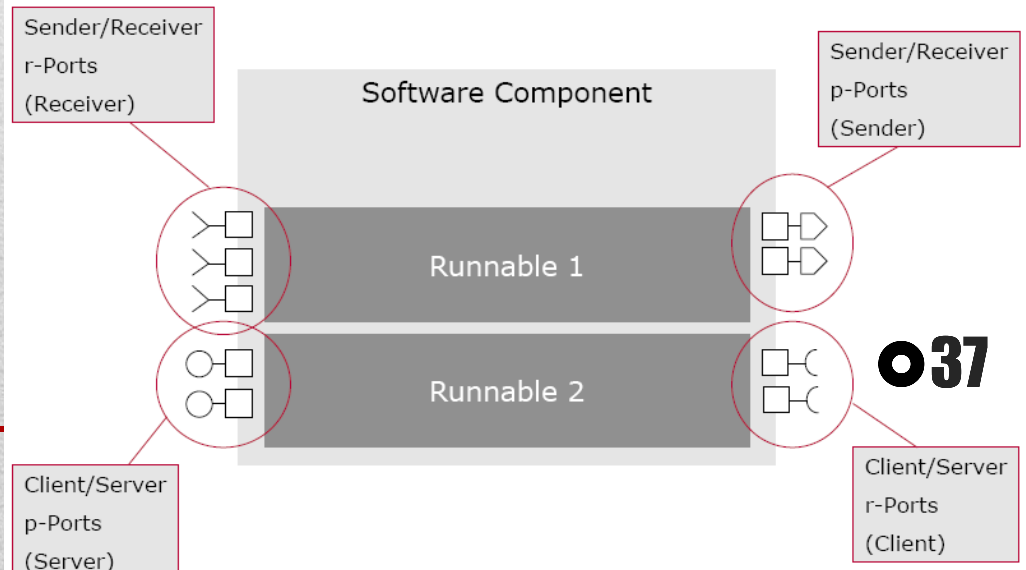
<u>AUTOSAR SW-Cs</u> (Software Components)

They **encapsulate an application** which runs on the AUTOSAR infrastructure. And they have well-defined interfaces, which are described and standardized by SW-C Description (Provides a standard description format needed for the integration of the SW-C)

- **Consist of**
  - Runnable Entities (or Runnables) procedures which contain the actual implementation triggered cyclically or on event (e.g. data reception)
  - Composite components for hierarchical design
  - Ports Interface to other SW-Cs
  - Other software components
- **Application** is divided into SW-Cs.

There are **three types** of SW-C

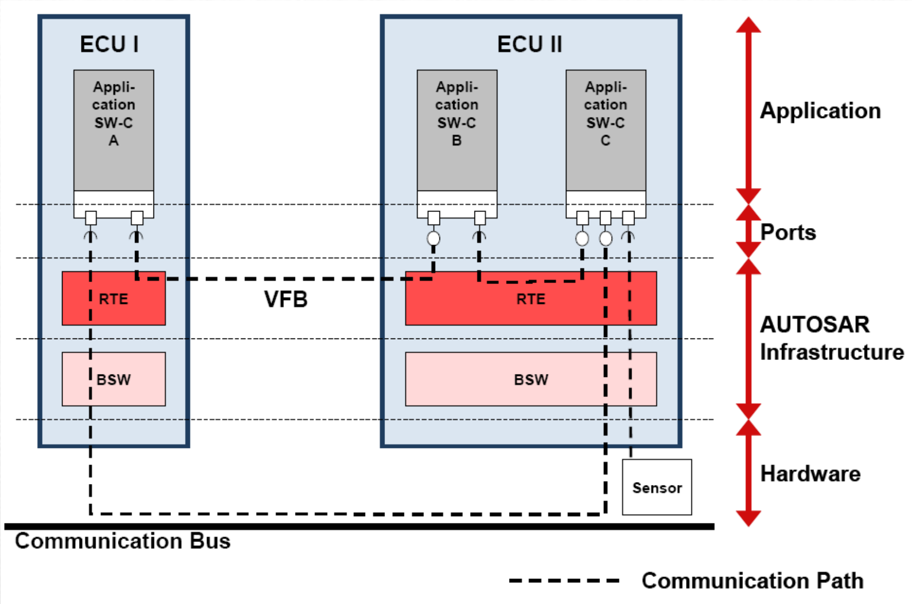
- Atomic SW-C
- Composite SW-C
- Sensor/Actuator SW-C



Main Concepts: Architecture

Intra- and Inter-ECU Communication

- **Ports** are the **interaction points** of a component and they **implement the interface** according to the communication paradigm (client-server based).
- **The communication layer** in the basic software is **encapsulated** and not visible at the application layer.



Communication is channeled via the

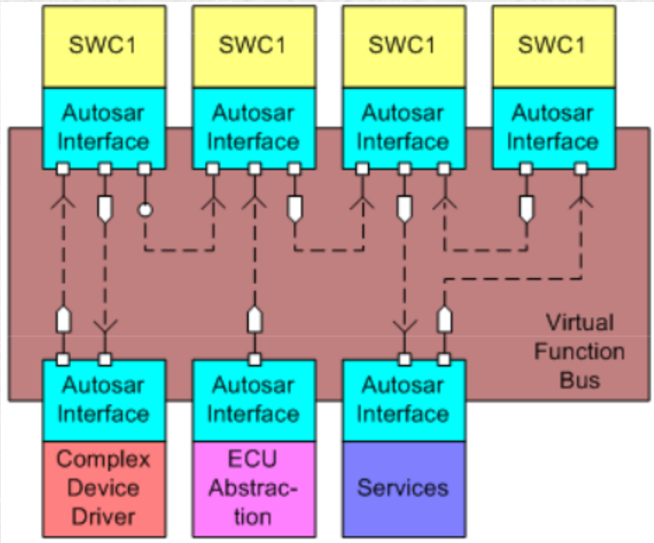
AutoSar RTE (RunTime Environment)

- Implementation of Virtual Functional Bus
- **Interface** between SW-Cs and Basic Software, therefore **it frees SW-Cs** from the hardware, BSW and from each other components.
- **Every ECU** in a AutoSar system must **implement a RTE**
- **All calls** to basic software **pass through the RTE,** so all software components can communicate without being mapped to specific hardware or ECU.
- **RTE uses** the hardware **MCAL** (MicroController Abstraction Layer)
- **Communication method** : Send/Receive signals, Client/Server functionality
- **Triggering of runnables** : Cyclically or On event

Main Concepts: Architecture

<u>The Virtual Functional Bus (VFB)</u>

- The VFB is the **sum of all communication mechanisms** (and interfaces to the basic software) provided by AUTOSAR on an abstract (technology independent) level. <u>When the connections for a concrete system are defined</u>, the VFB **allows a virtual integration** in an early development phase.
- The <u>application components</u> are **linked and communicate** through VFB.
- VFB is a **visualization of all** <u>hardware</u> and system <u>services</u> that the vehicle system supplies.
- Through VFB a **software component doesn´t need to know** <u>which</u> components it is communicating with and <u>on which</u> ECU these components are implemented.
- The VFB is **implemented by** the AUTOSAR Runtime-Environment (<u>RTE</u>) and underlying <u>Basic-SW</u> layer.



Main Concepts: Architecture

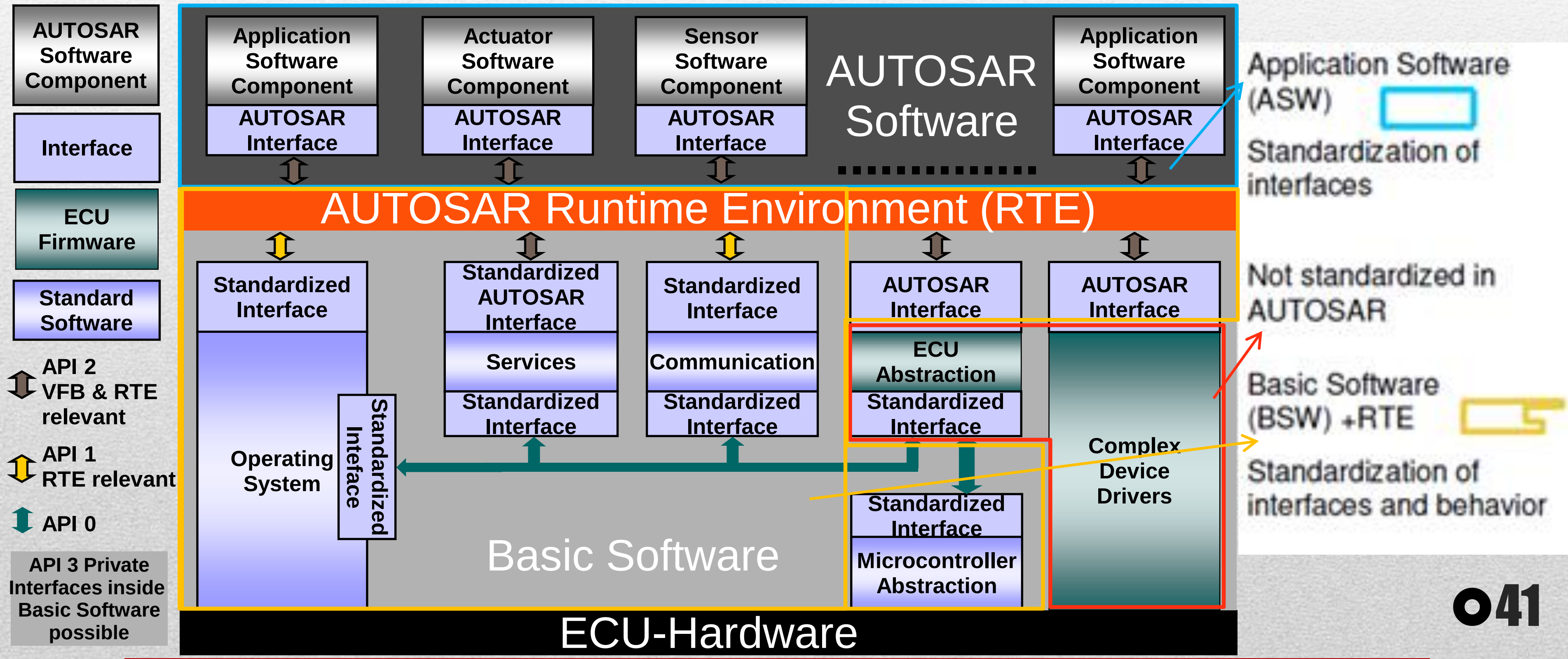
- Communication between software components

A SWC can communicate in two different ways:

Client/server:	Sender/receiver:
➢ **The client** initiates the communication and requests a service from the server ➢ **The client** could be locked while it is waiting for an answer from the server. ➢ **The Client/server** roles are defined by who is initiating the communication and could be switched. → ➢ **A SWC** can at the same time act as both client and server in different communications	➢ **The sender** expects no answer from the receiver as there will no be any. ➢ **The sender** is not blocked. ➢ **The receiver** decides on it´s own how and when to act on the information. ➢ **The interface** structure is responsible for the communication. ➢ **The sender** doesn´t know ❖ Who the receiver is, ❖ if there are More than one receiver, ❖ or in What ECU the receiver is situated

Main Concepts: Architecture

- AUTOSAR Layered Architecture & ECU Software Architecture



Objectives: Basic SW: Decoupling of Hardware and Application Software
Application SW: Relocation / Reuse of SW-Components between ECUs

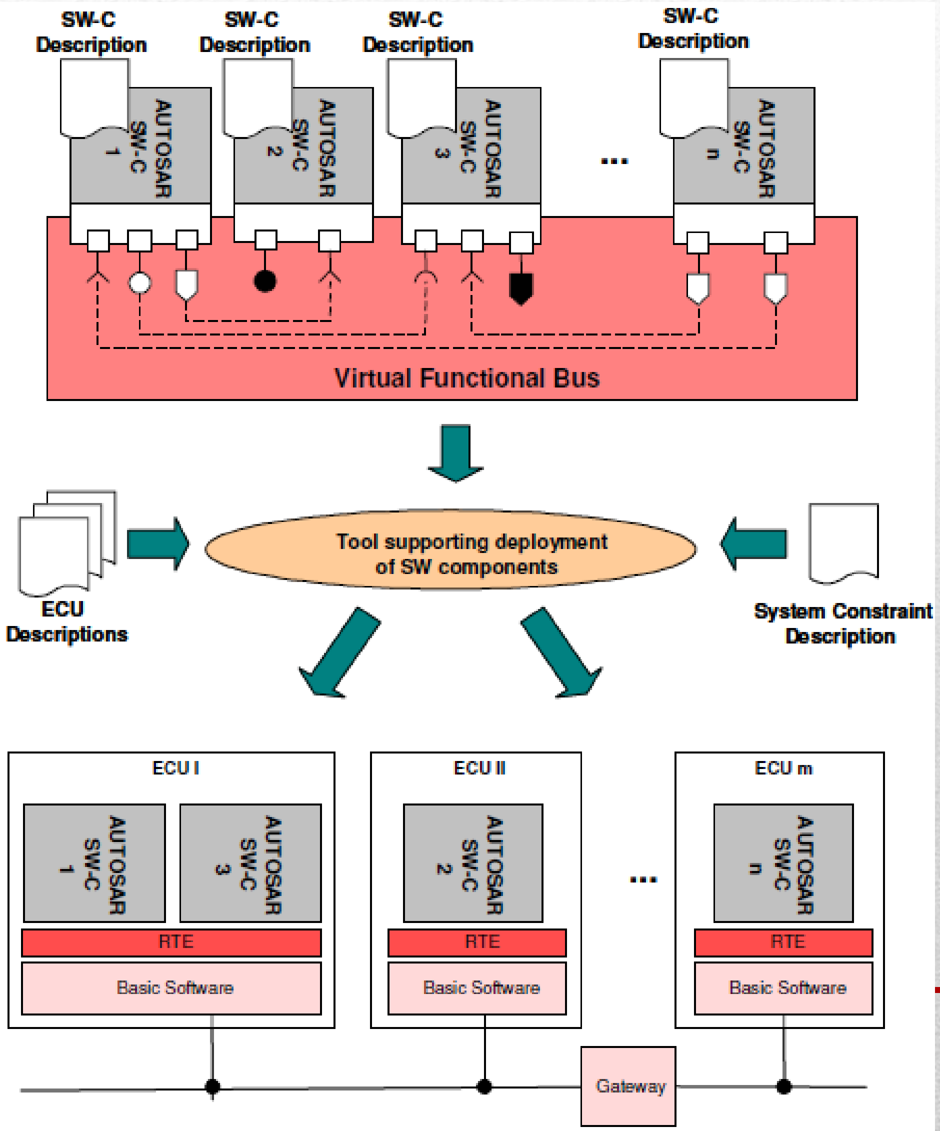
Main Concepts: Architecture

Conclusion

- AUTOSAR **harmonizes** already **existing basic software** solutions and **closes gaps** for a seamless basic software architecture.
- **The decomposition** of the AUTOSAR layered architecture into some 80 modules **has proven to be functional and complete.**
- AUTOSAR **aims at finding the best solution** for each requirement and **not finding the highest common multiple**.

Main Concepts: Methodology

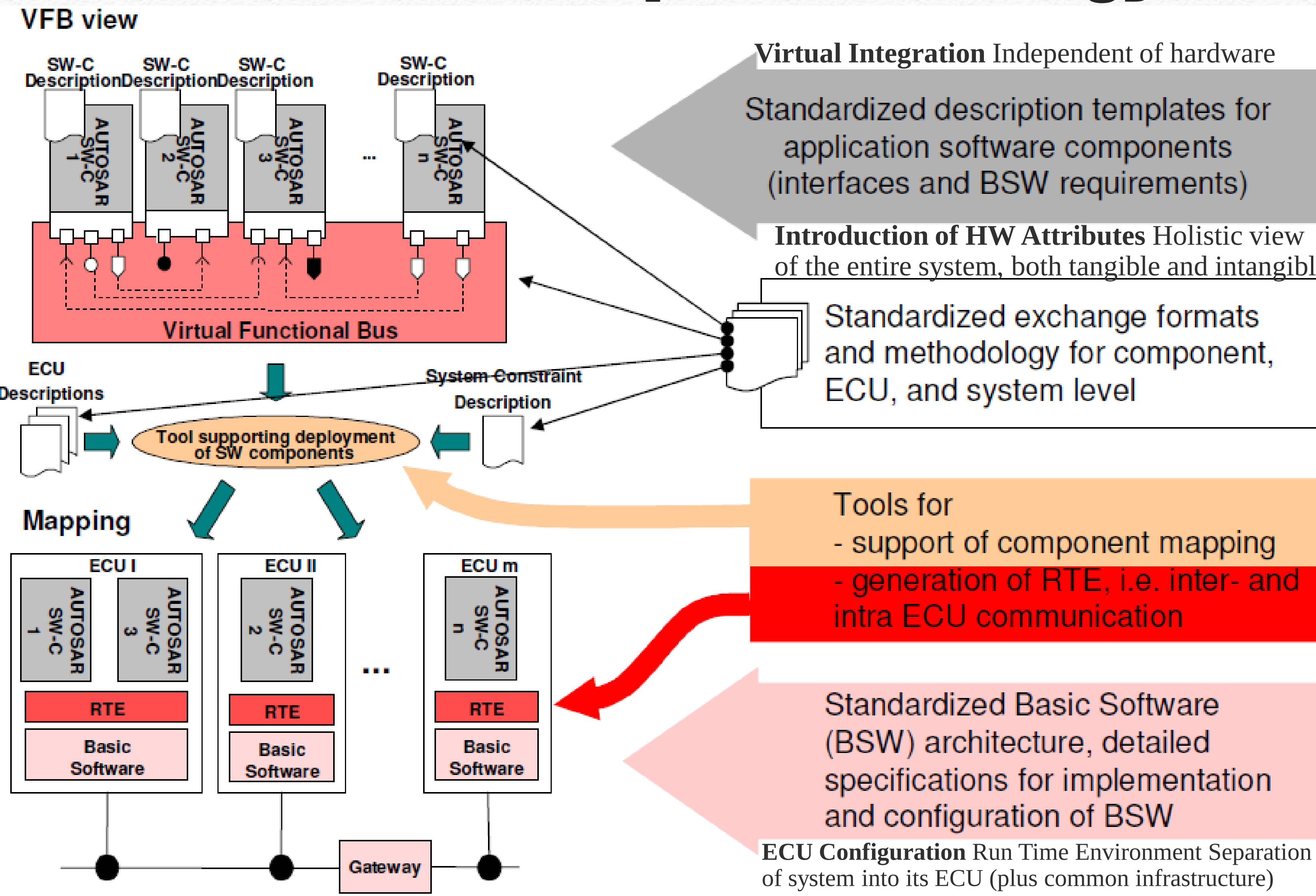
Main Concepts: Methodology



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

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

Main Concepts: Methodology



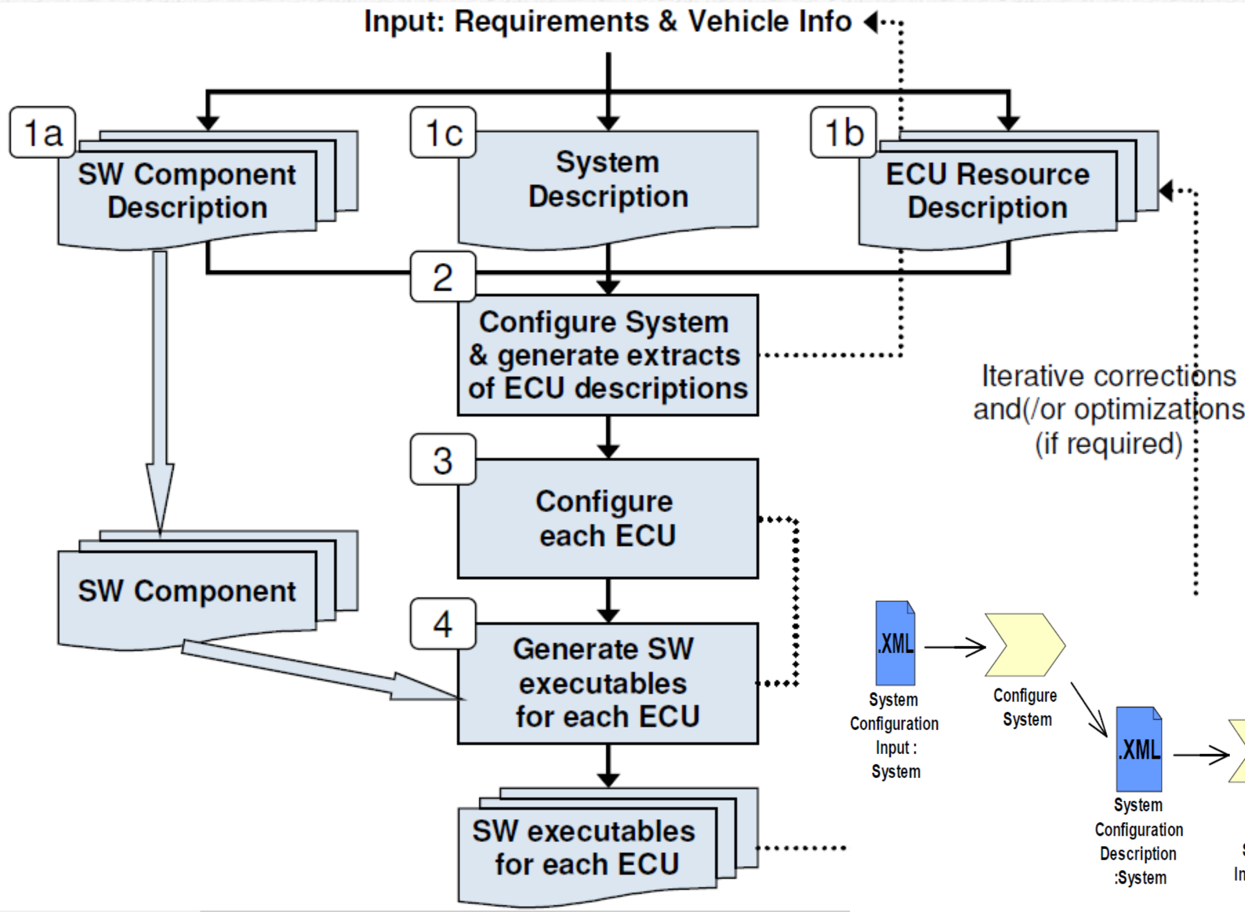
Virtual Integration Independent of hardware

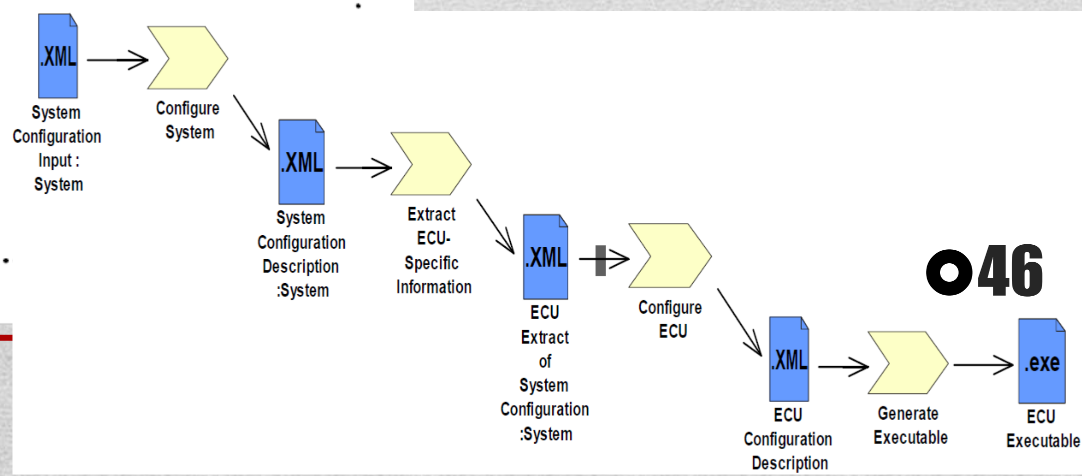
Introduction of HW Attributes Holistic view of the entire system, both tangible and intangible

ECU Configuration Run Time Environment Separation of system into its ECU (plus common infrastructure)

Main Concepts: Methodology

- AUTOSAR System - Design Process - Implementation Process





Main Concepts: Methodology

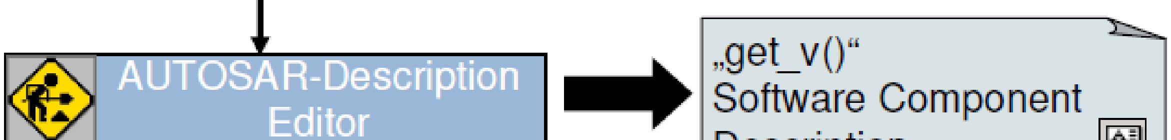
- ***Step 1a):*** Input Description (1 of 3) of ***SW-Components*** independently of hardware

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

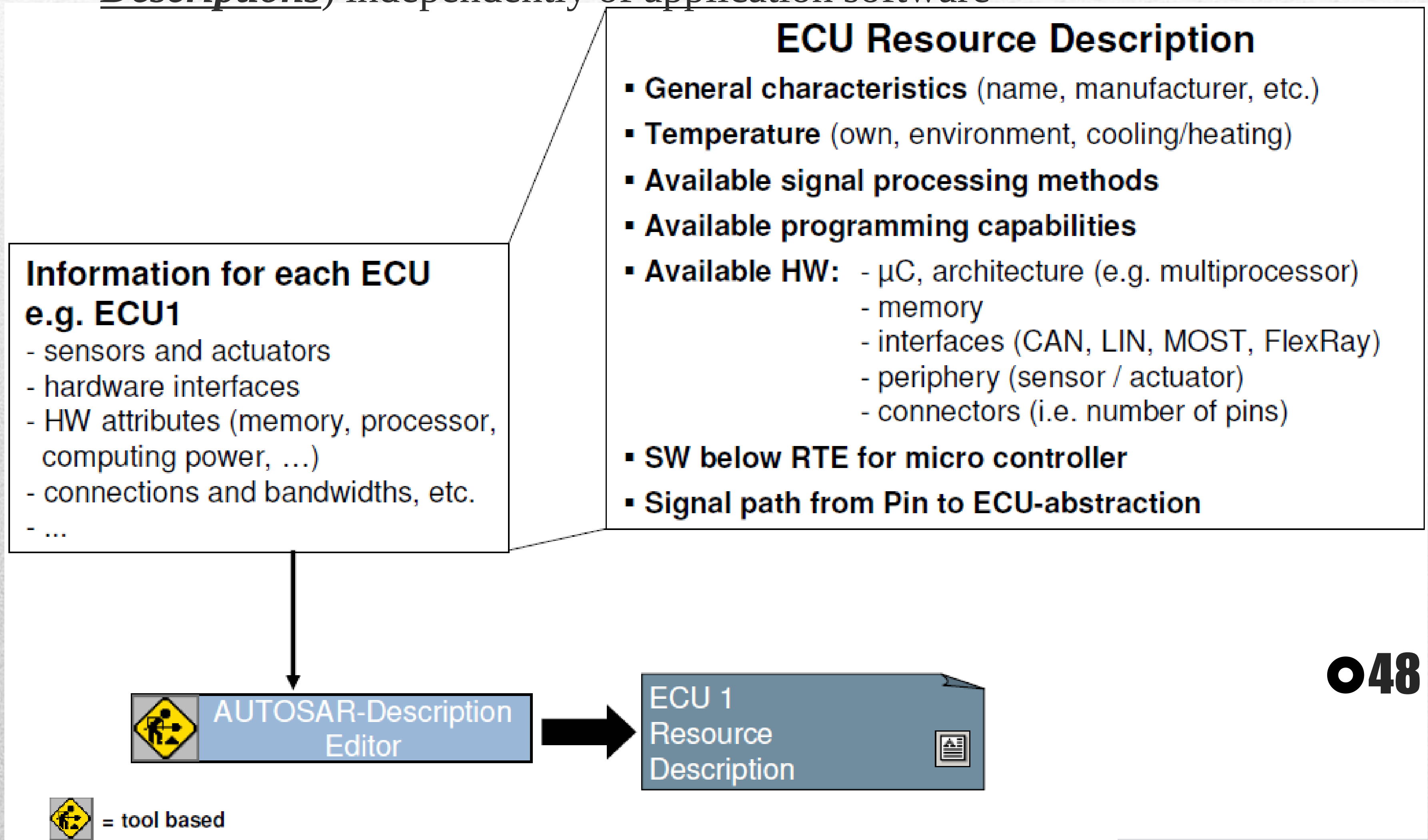
  inner structure (composition)
  - sub-components
  - connections
- **required HW resources:**
  - processing time
  - scheduling
  - memory (size, type, etc.)



= tool based

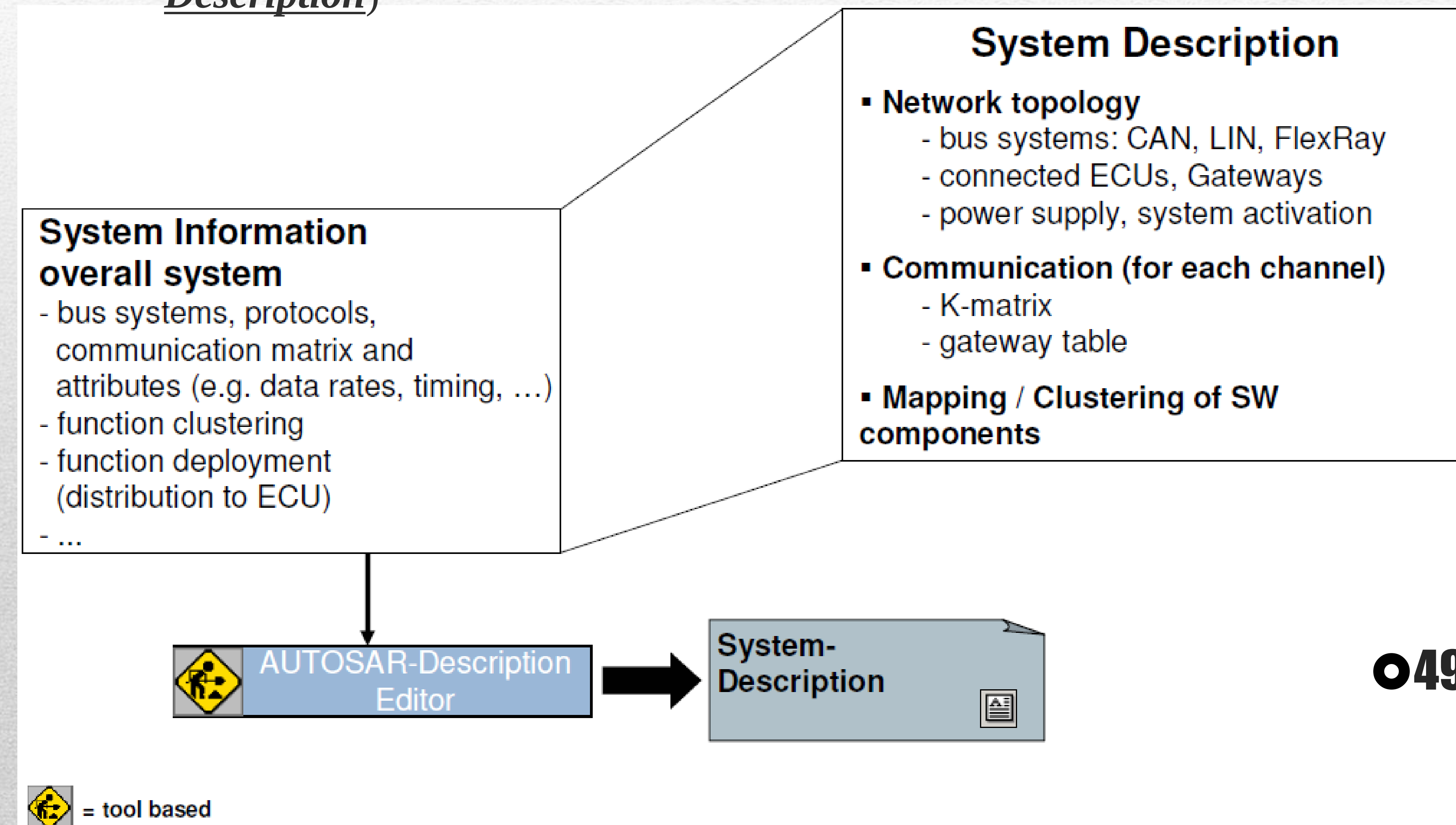
Main Concepts: Methodology

- ***Step 1b):*** Input Description (2 of 3) of hardware (***ECU-Resource-Descriptions***) independently of application software



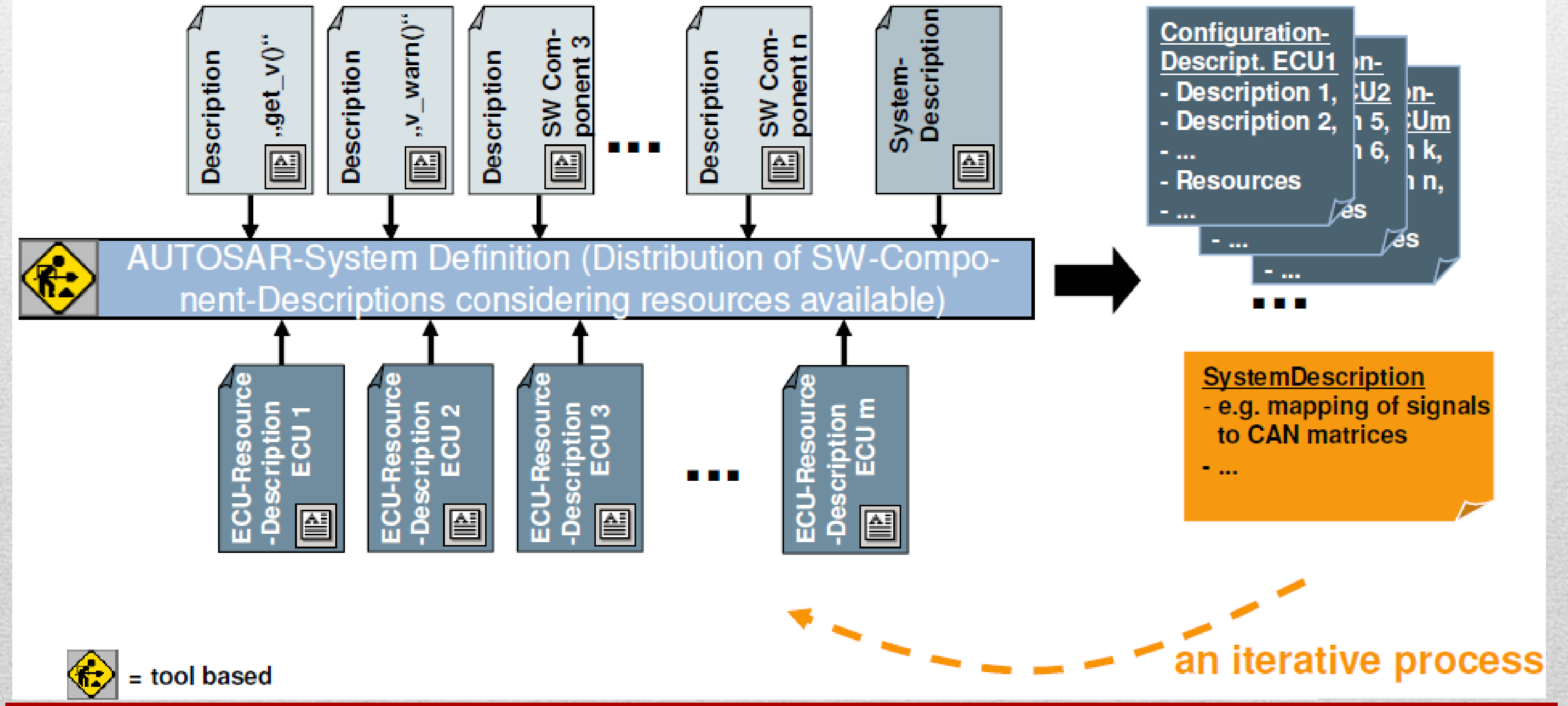
Main Concepts: Methodology

- ***<u>Step 1c):</u>*** Input Description (3 of 3) of system (***<u>System-Constraint-Description</u>***)



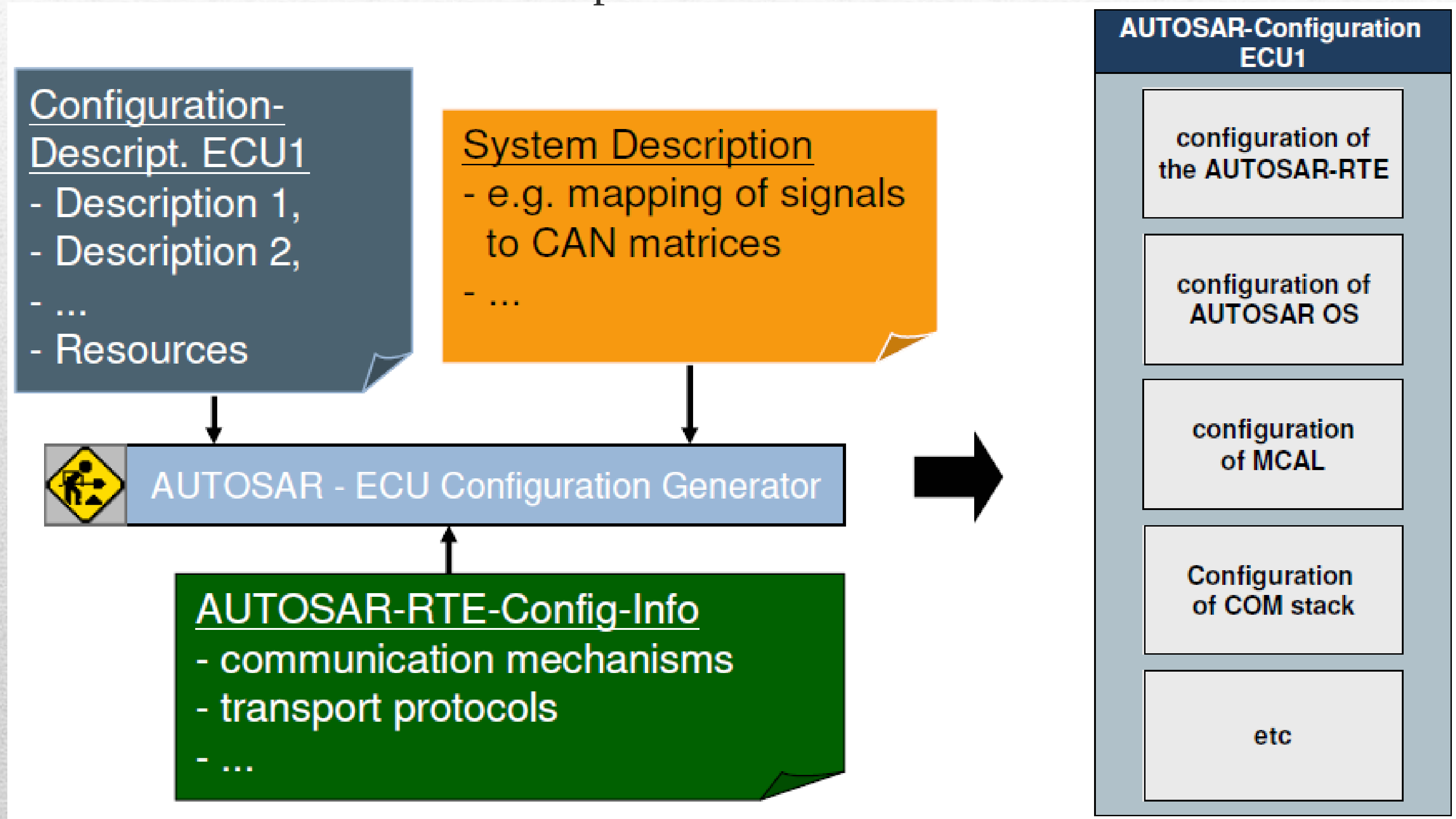
Main Concepts: Methodology

- ***Step 2:*** Distribution of SW-Component-Descriptions to all ECU with
  - System Configuration on the basis of descriptions (not of implementations!) of SW Components, ECU-Resources and System-Description.
  - Consideration of ECU-Resources available and constraints given in the Syste m-Description.



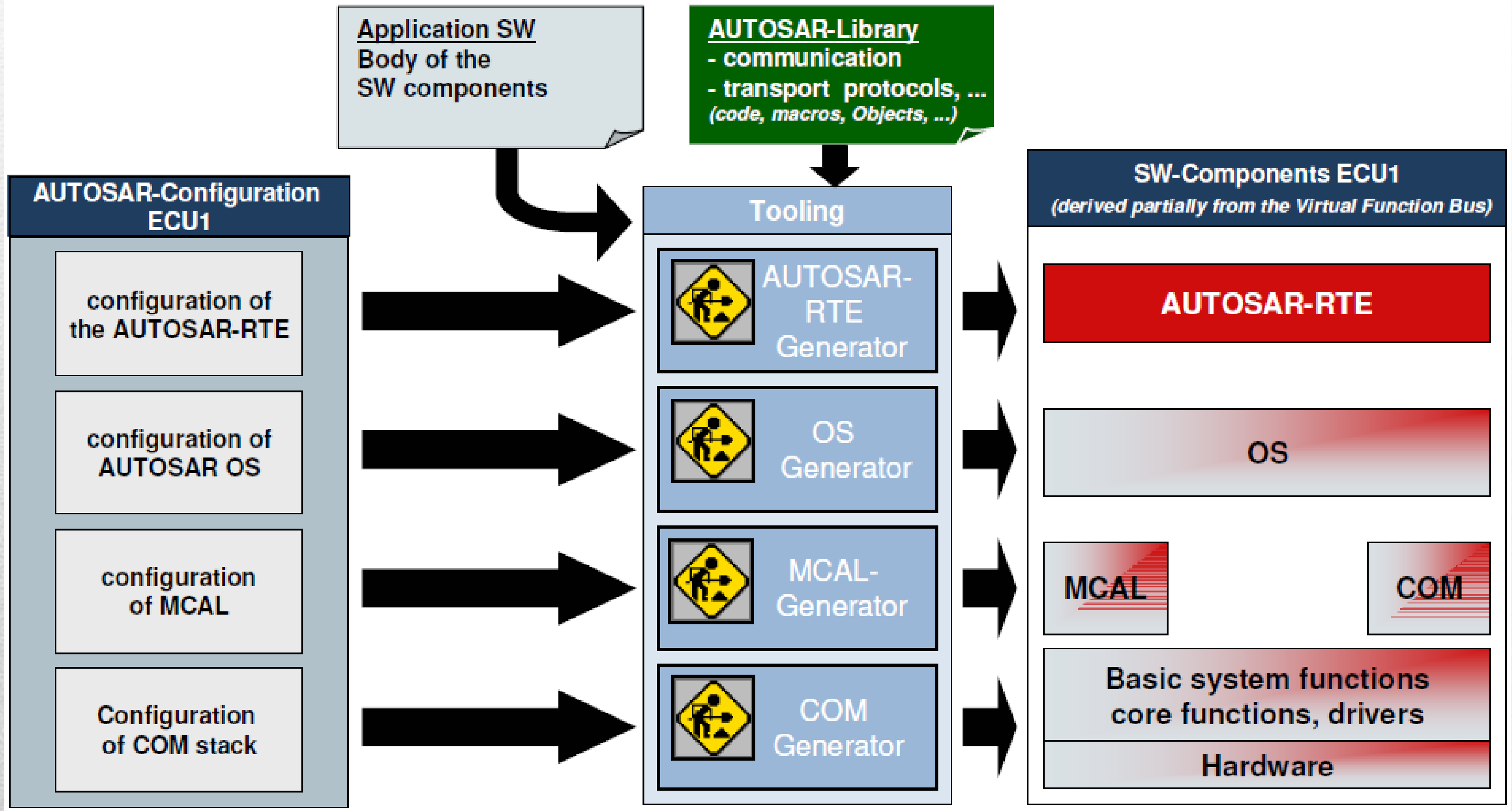
Main Concepts: Methodology

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



Main Concepts: Methodology

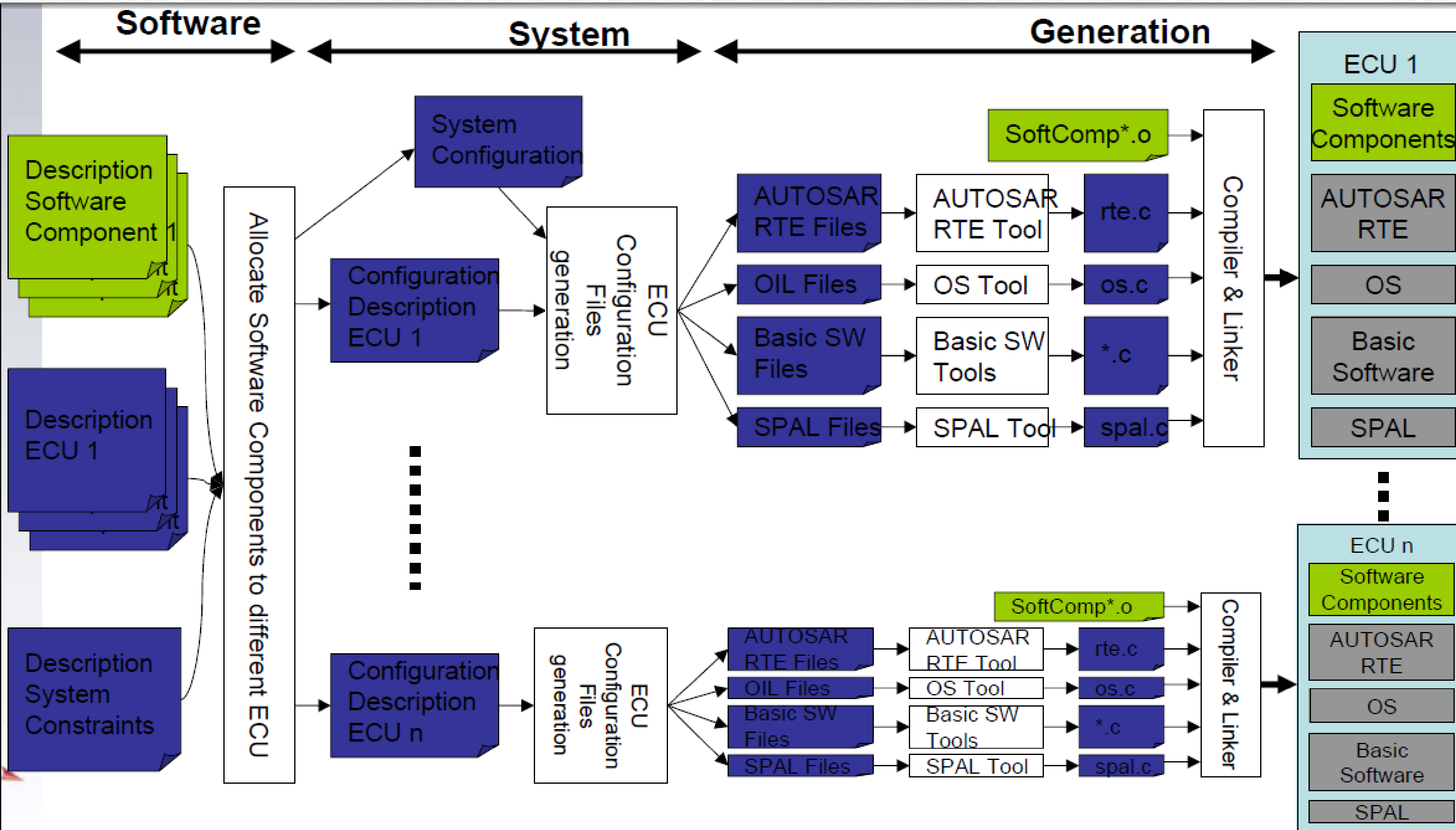
- ***Step 4:*** ***Generation of Software Executables*** Based on the configuration information for each ECU (example ECU1)



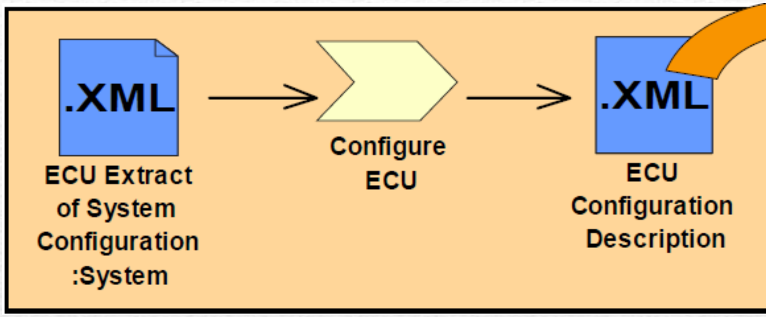
1) If need be, extract for ECU1 only

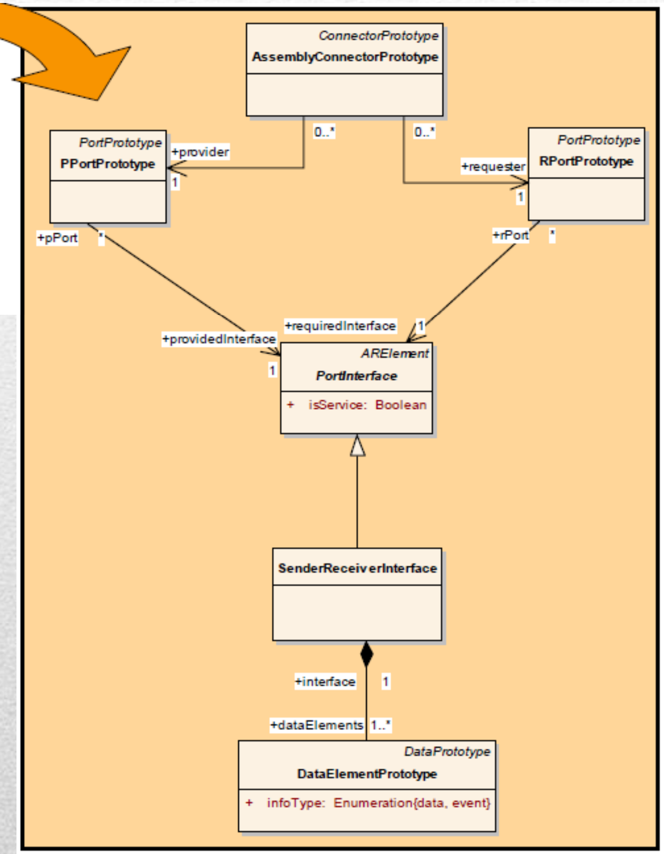
2) SPAL: Standardized Peripheral Abstraction Layer

Main Concepts: Methodology



Main Concepts: Methodology





AutoSar Development process

- AutoSar has given a **method for creating the system** architecture that **starts in the design model**.
- The **model descriptions** within AutoSar are standardized to become **tool independent**.
- The **descriptions have UML syntax** (Unified Modeling Language).
- The **basic system descriptions** are **independent** of how they are to be **implemented**.
- Necessary **data** are among others **interface and hardware** demands **Standard interfaces are described in XML** (eXtendable Mark-up Language).

Main Concepts: Methodology

AutoSar MetaModel

- is modeled in UML
- is the **backbone of the AutoSar architecture** definition.
- **contains complete specification**, how to model AUTOSAR systems
- **Integrates methodology** which defines activities and work-products

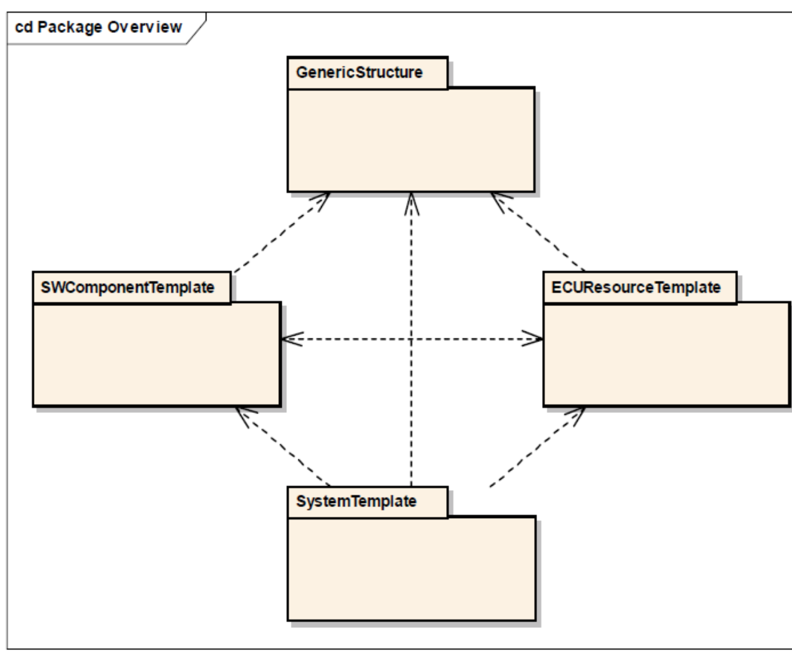
- **Defines content of work-products**, Formal description of all the information that is produced or consumed in the AUTOSAR methodology
- Has benefits as:
  - The **structure of the information** can be clearly visualized and easily maintained.
  - The **consistency of the information and terminology** is guaranteed
  - Using XML, a data exchange format can be generated automatically out of the MetaModel

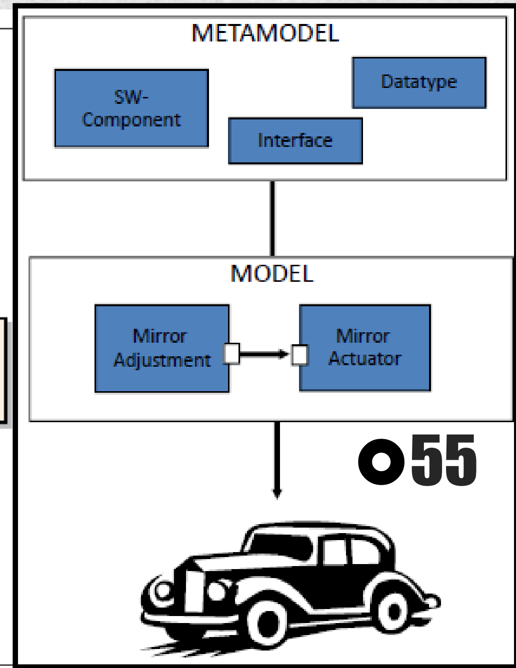
M3: Model of the Metamodel (Meta-Metamodel) *(Defines UML Modeling Elements)*

M2: Model of the model (Metamodel) *(Defines AUTOSAR Modeling Elements)*

M1: Model of the system *(Defines a real system)*

M0: Realized System in the car *(Implements a real system)*

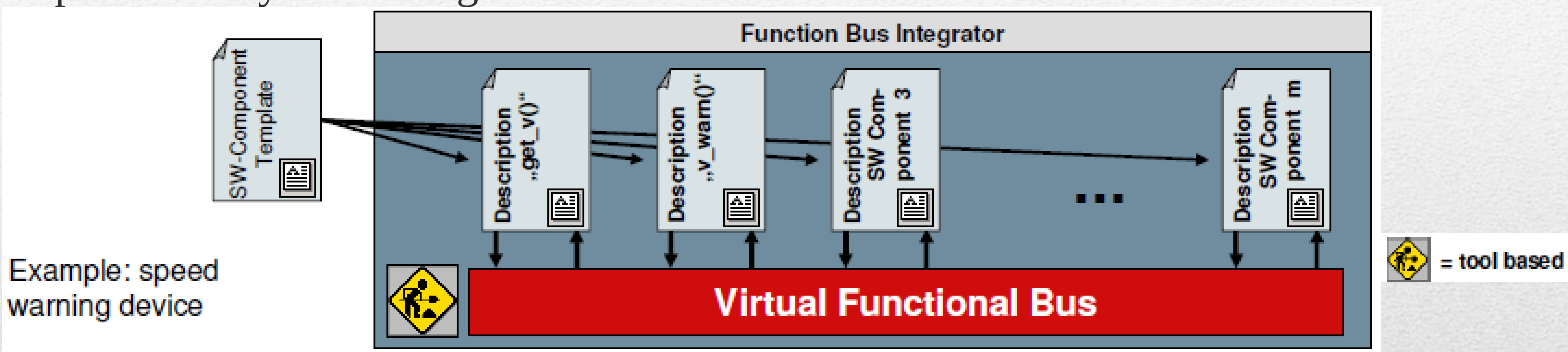




Main Concepts: Methodology

Example 1: The Virtual Functional Bus (VFB)

- Input to the System Design on an abstract level

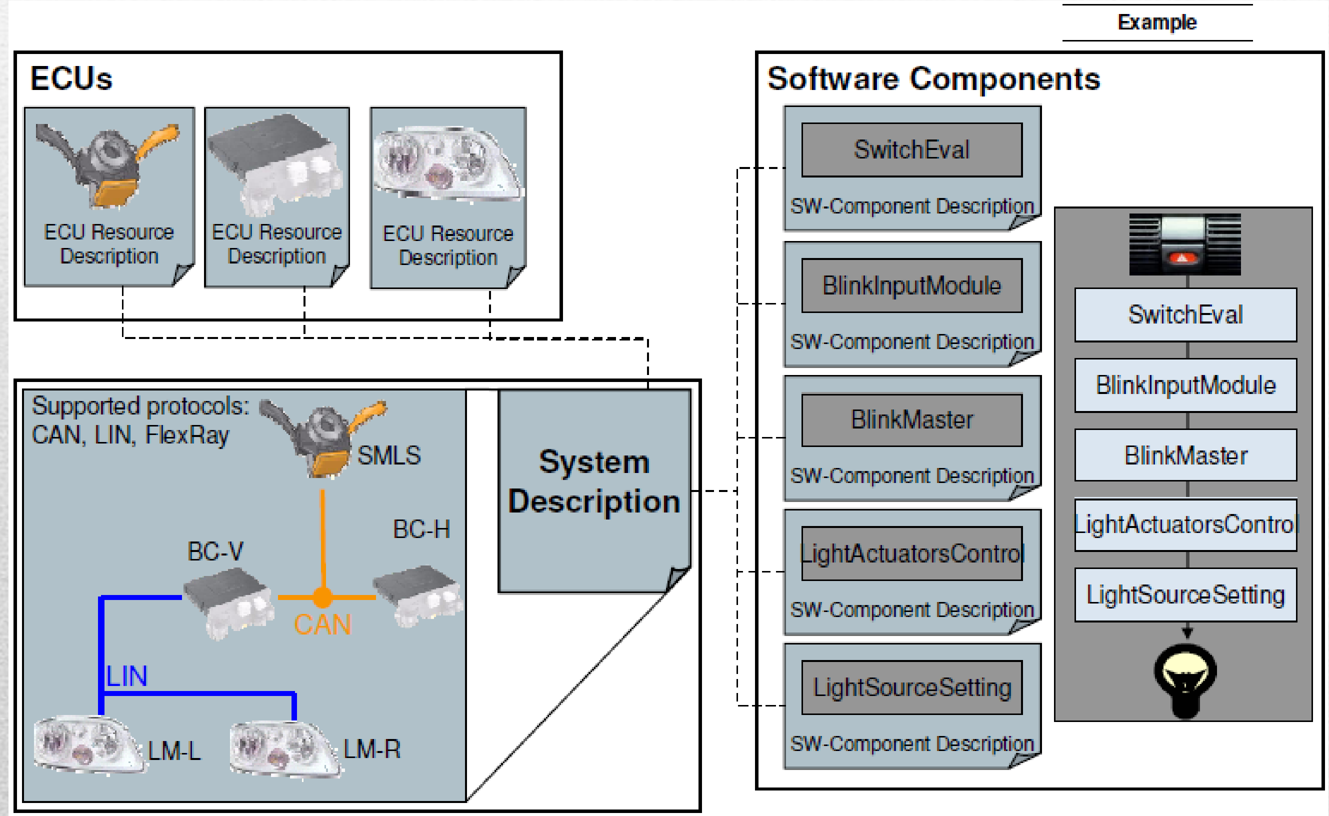


- **SW-Component-Description “get_v()“ describes a function to acquire** the current **vehicle speed** and defines the **necessary resources** (such as memory, run-time and computing power requirements, etc.)
- Function “v_warn()“ makes use of “get_v()“
- “ Virtual Integration“ by checking of
  - **Completeness** of SW-Component-Descriptions (entirety of interconnections)
  - **Integrity/correctness** of interfaces

Main Concepts: Methodology

Example 2: AutoSar Descriptions:

To **configure the system**, **input descriptions** of all software components, ECU resources and system constraints **are necessary**.

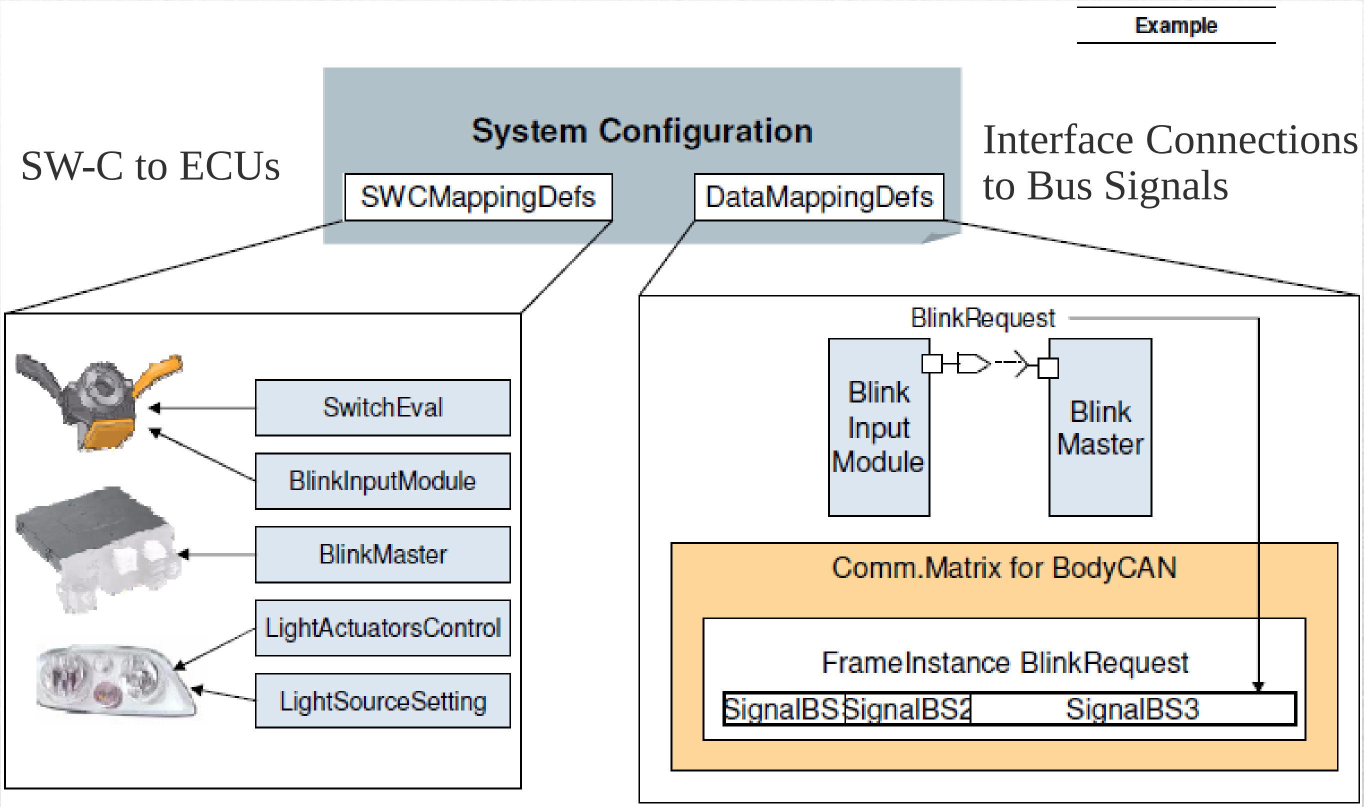


Description of Topology

Main Concepts: Methodology

Example 2: System Configuration

Maps SW-C to ECUs **and** links interface connections to bus signals.



Main Concepts: Methodology

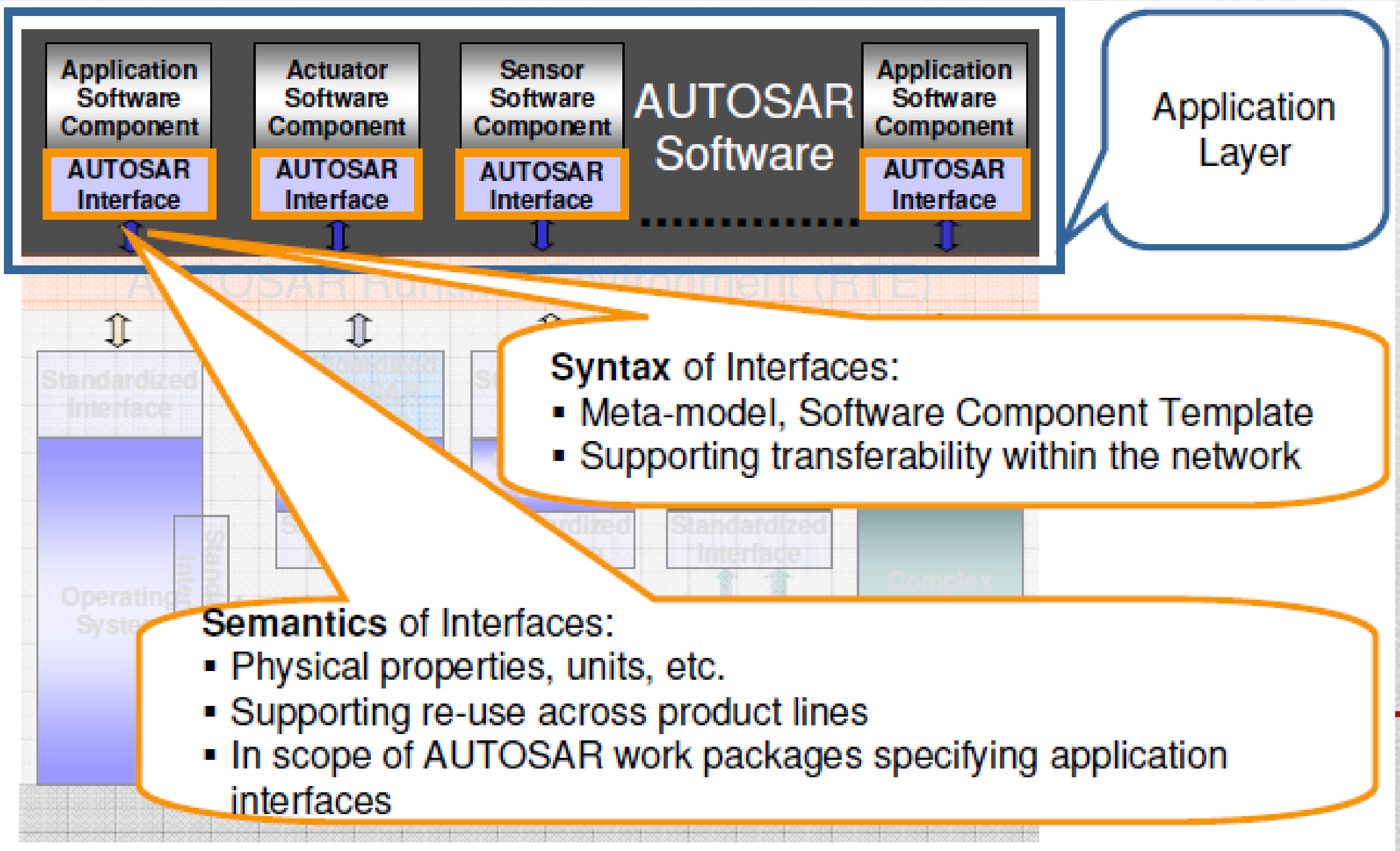
Conclusion

- The E/E system architecture can be **described by means of AUTOSAR.**
- A methodology to **integrate AUTOSAR software modules** has been designed.
- The meta model approach and the tool support for specifying the AUTOSAR information model allow **working at the right level of abstraction.**

Main Concepts: Application Interfaces

Main Concepts: Application Interfaces

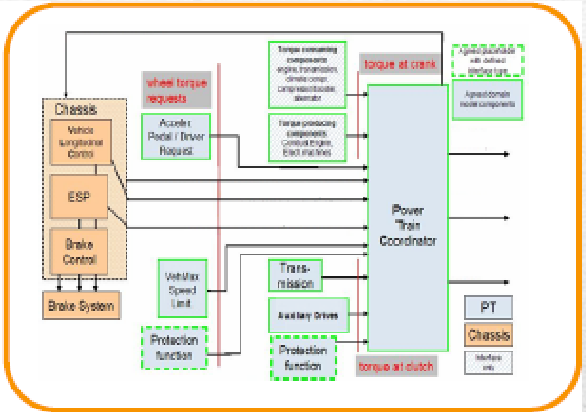
- Standardized AUTOSAR interfaces approach will
  - **support HW independence,**
  - **enable the standardization of SW** components and
  - **ensure the interoperability of functional SW-C** (applications) from different sources.



Main Concepts: Application Interfaces

- 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
  - Exterior Lights
  - Defrost Control
  - Seat climatization
  - Cabin climatization
  - Steering wheel climatization
  - Window Control
  - Sunroof/Convertible control
  - Steering column adjustment
  - Roller blind control

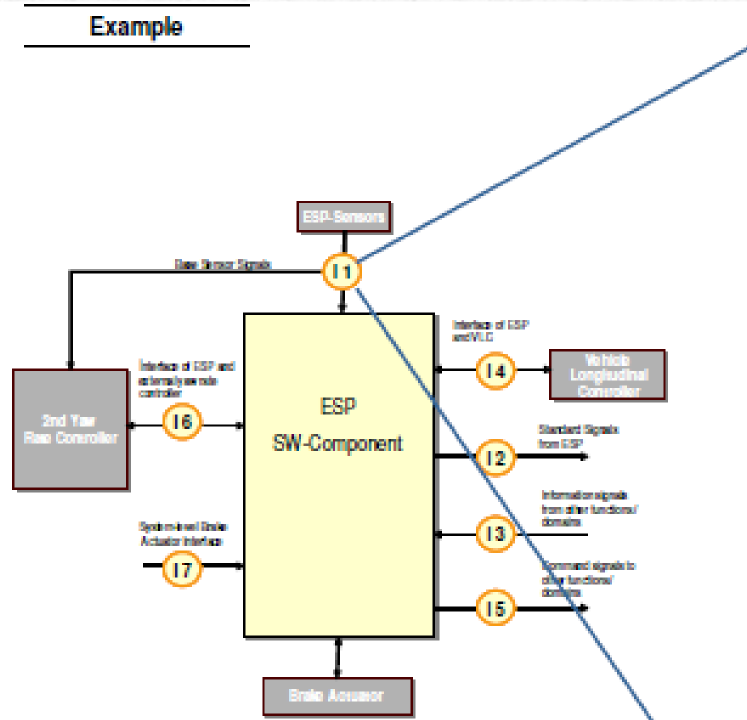


- **Chassis Control Domain**
  - Vehicle Longitudinal Control
  - Electronic Stability Program
  - Electronic Parking Brake
  - Adaptive Cruise Control
  - Roll Stability Control
  - Steering System
  - Suspension System
  - Stand Still Manager
  - High Level Steering
    - Vehicle Stability Steering
    - 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

Main Concepts: Application Interfaces

- **To ease the re-use of SW-C** across several OEMs.

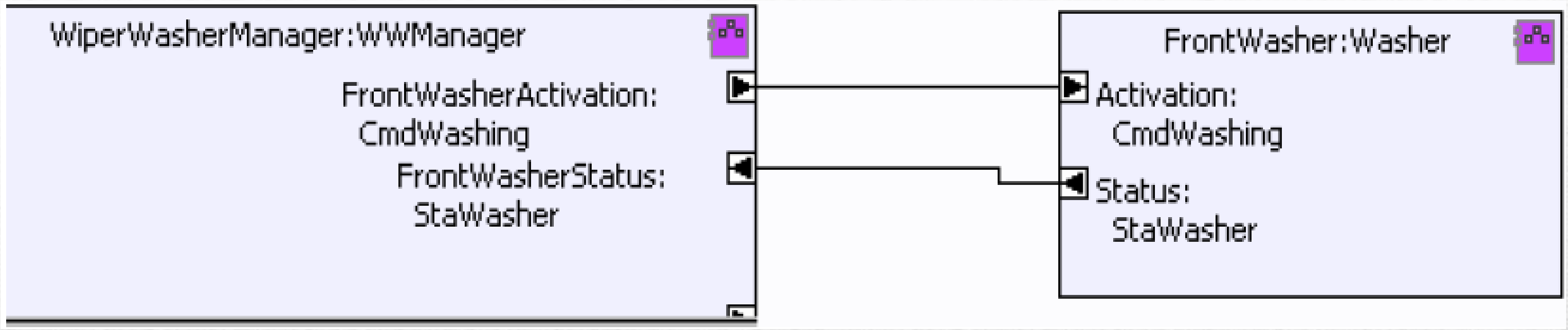


Standardized application interfaces on system level
(ESP-system, chassis domain)

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

Main Concepts: Application Interfaces

- Glance on Application Interfaces – Body Domain

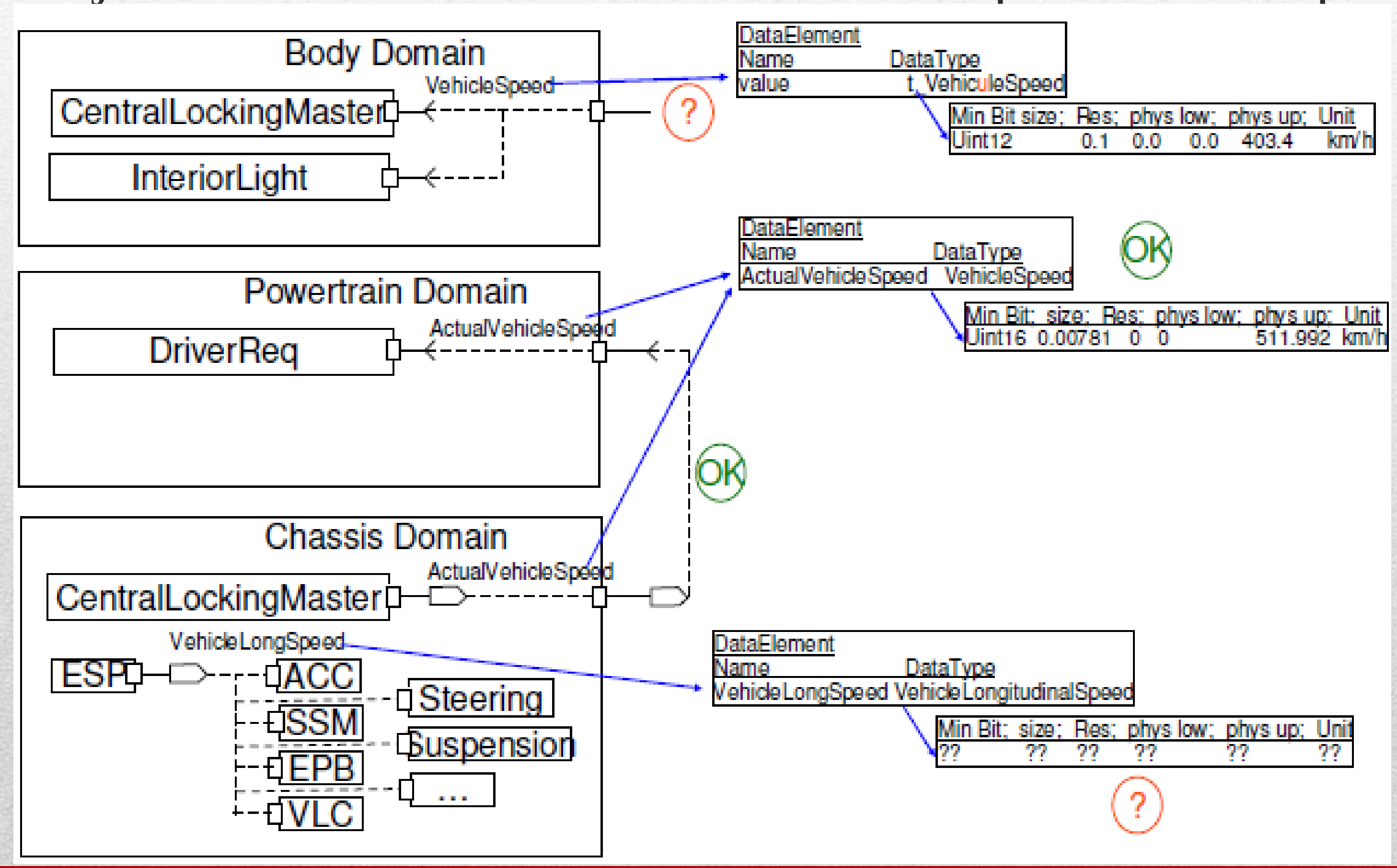


CmdWashing is the interface defined by following information:

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

Main Concepts: Application Interfaces

- Major task: Conflict Resolution – Example Vehicle Speed



Main Concepts: Application Interfaces

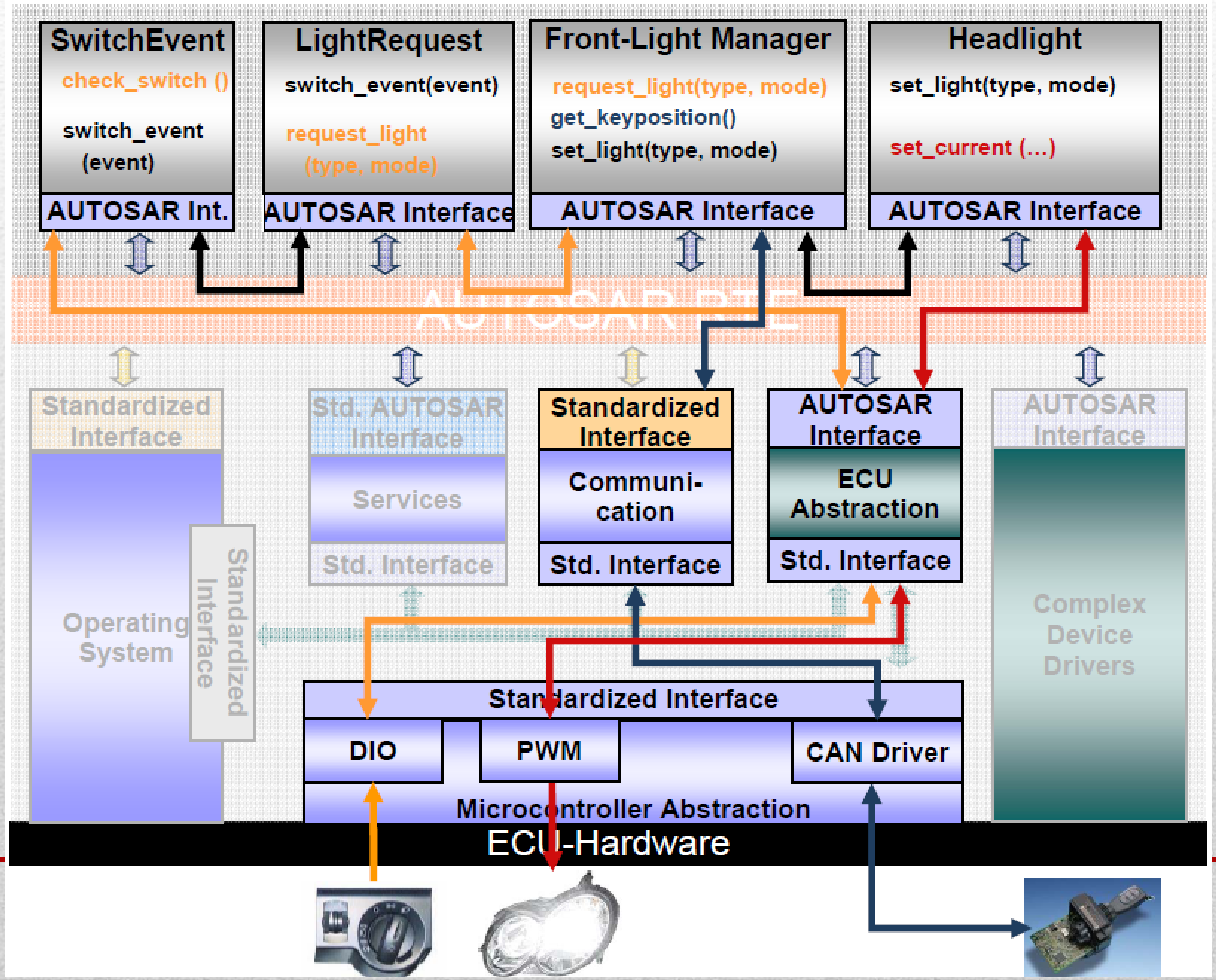
Conclusion

- For several domains a subset of **application interfaces has been standardized** to agreed levels.
- It is a **challenge to align** standardization with the pace of application development.

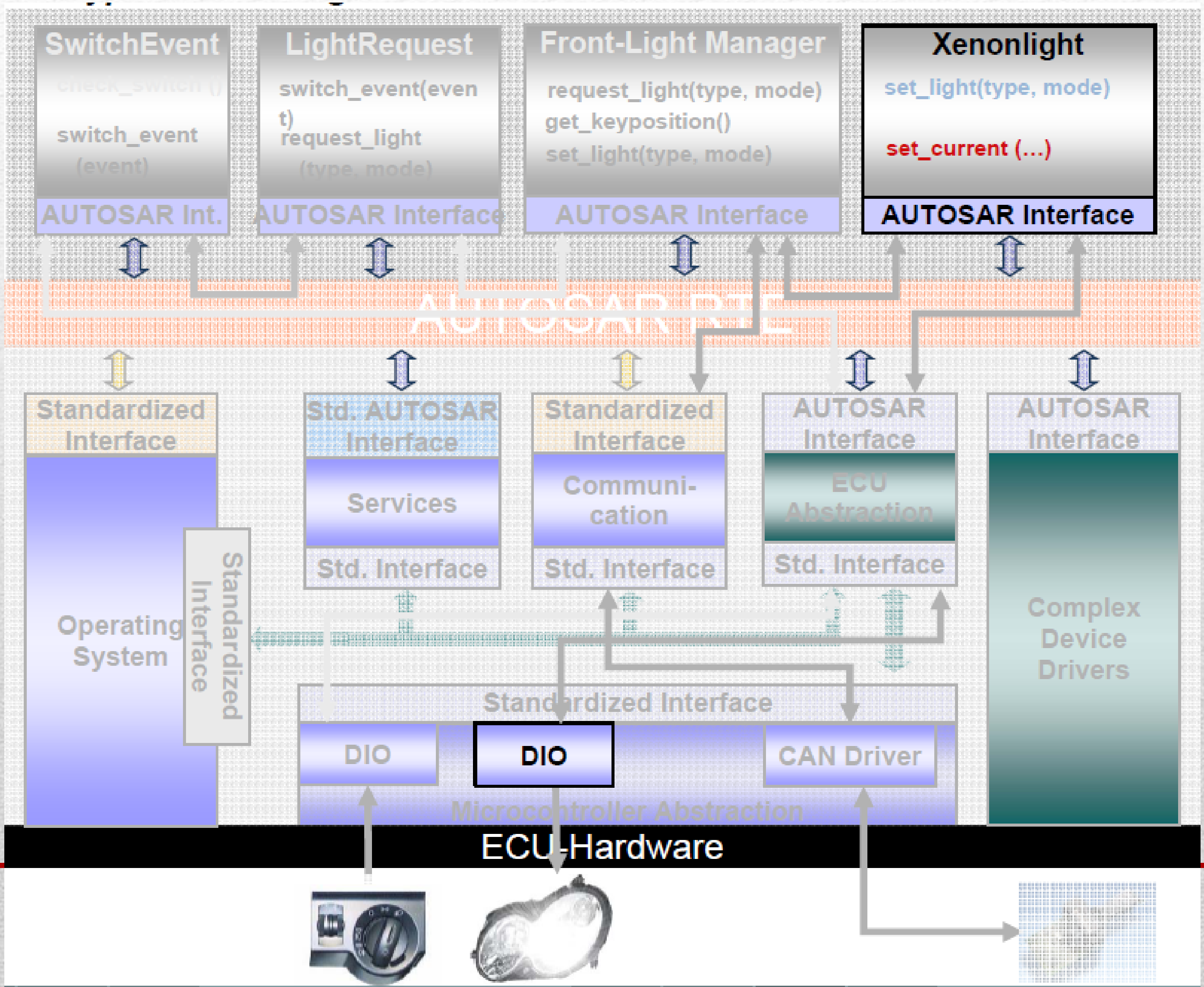
Contents

- What is AUTOSAR
- Project Objectives & Benefits
- Use case “ Front-Light Management”
- AUTOSAR Main Concepts
    - Architecture
    - Methodology
    - Application Interfaces
- Example of AUTOSAR System
- Conclusion

Use case " Front-Light Management"

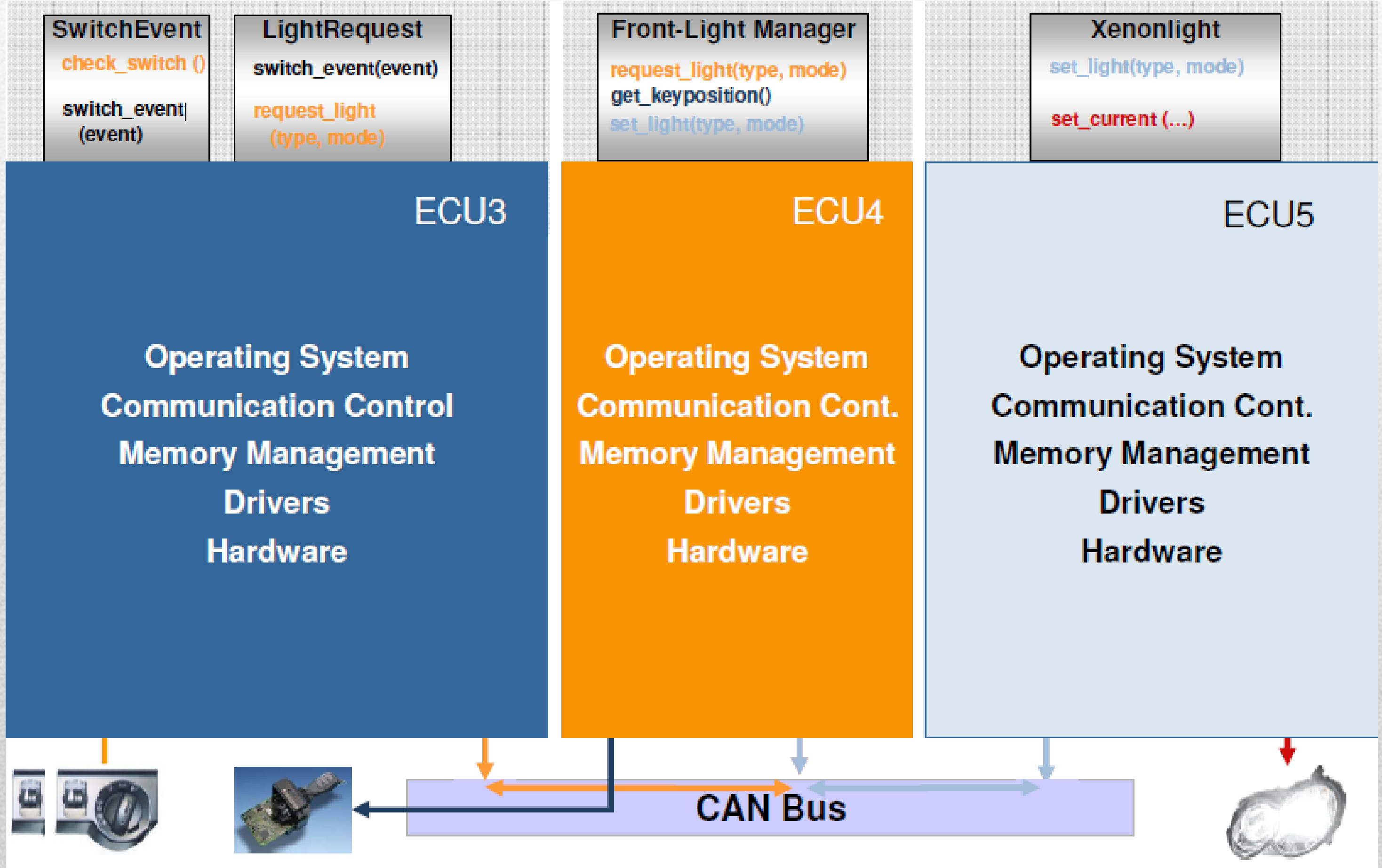


Use case " Front-Light Management"



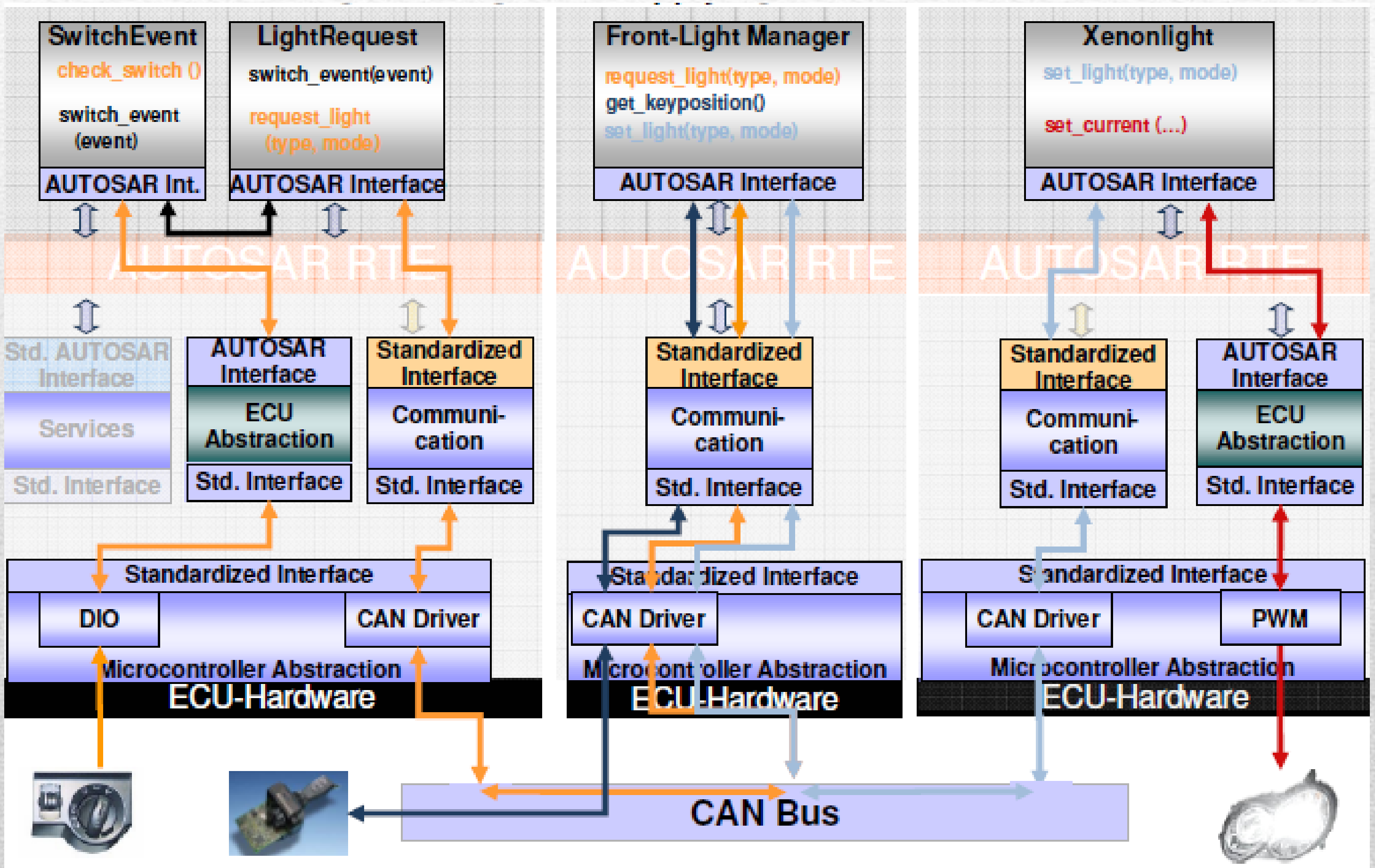
Use case " Front-Light Management"

- Multiple ECUs



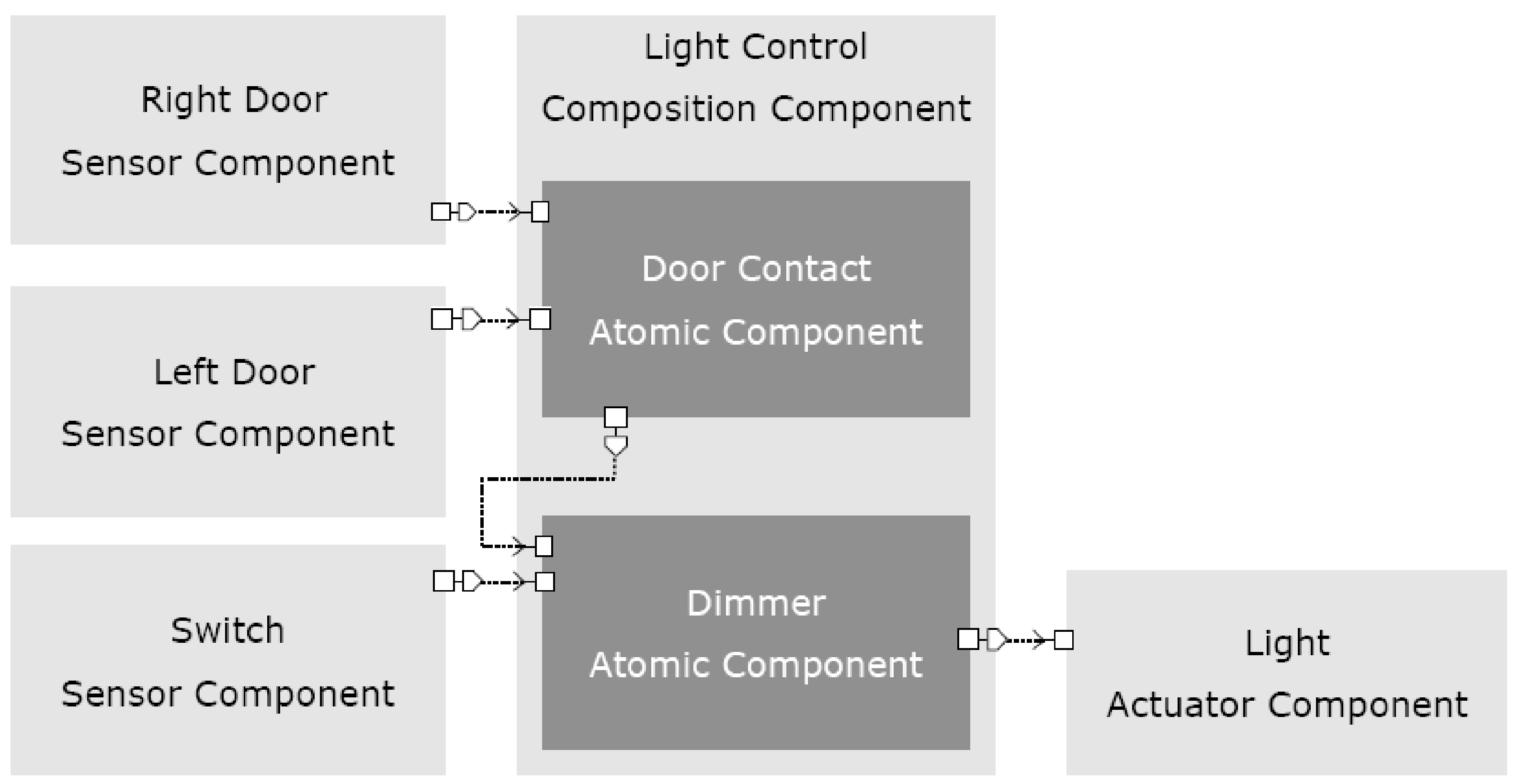
Use case " Front-Light Management"

- Applying AUTOSAR



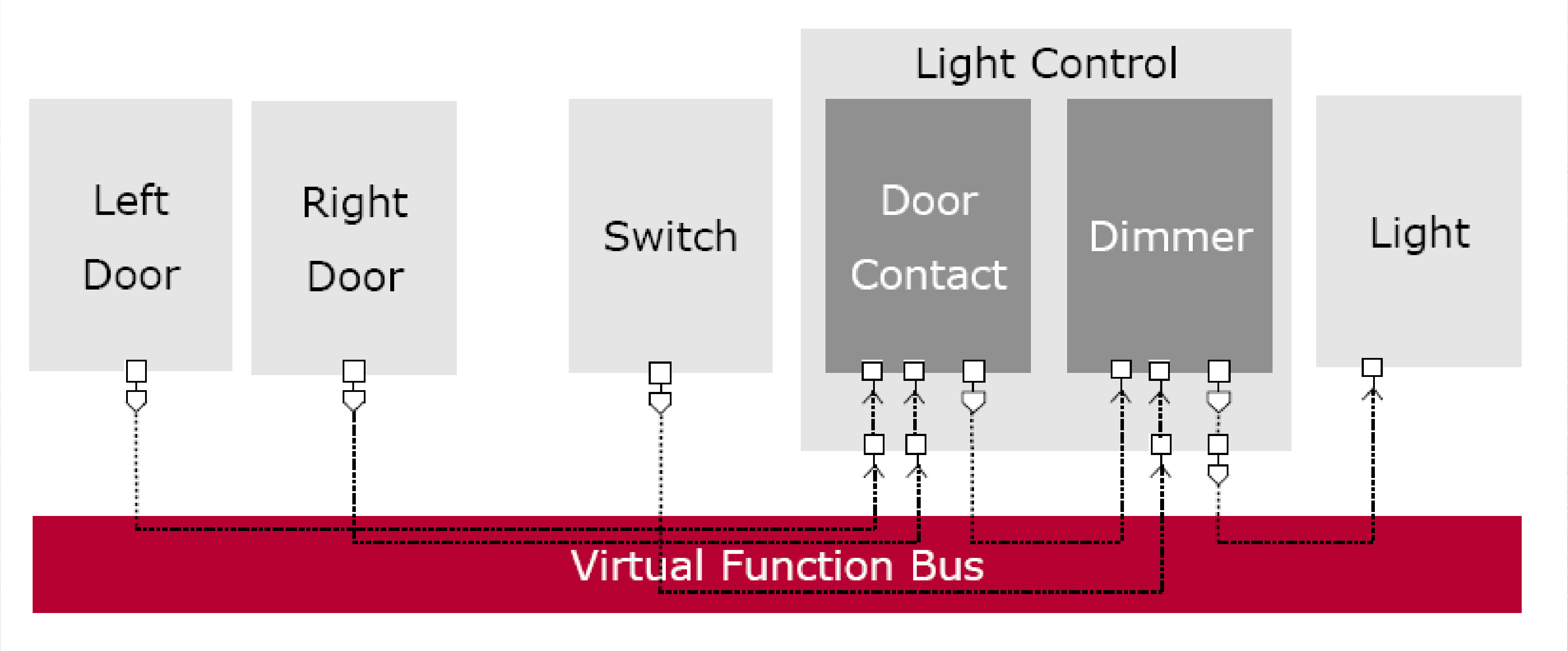
Example AUTOSAR System : Lighting System

- Software Component View



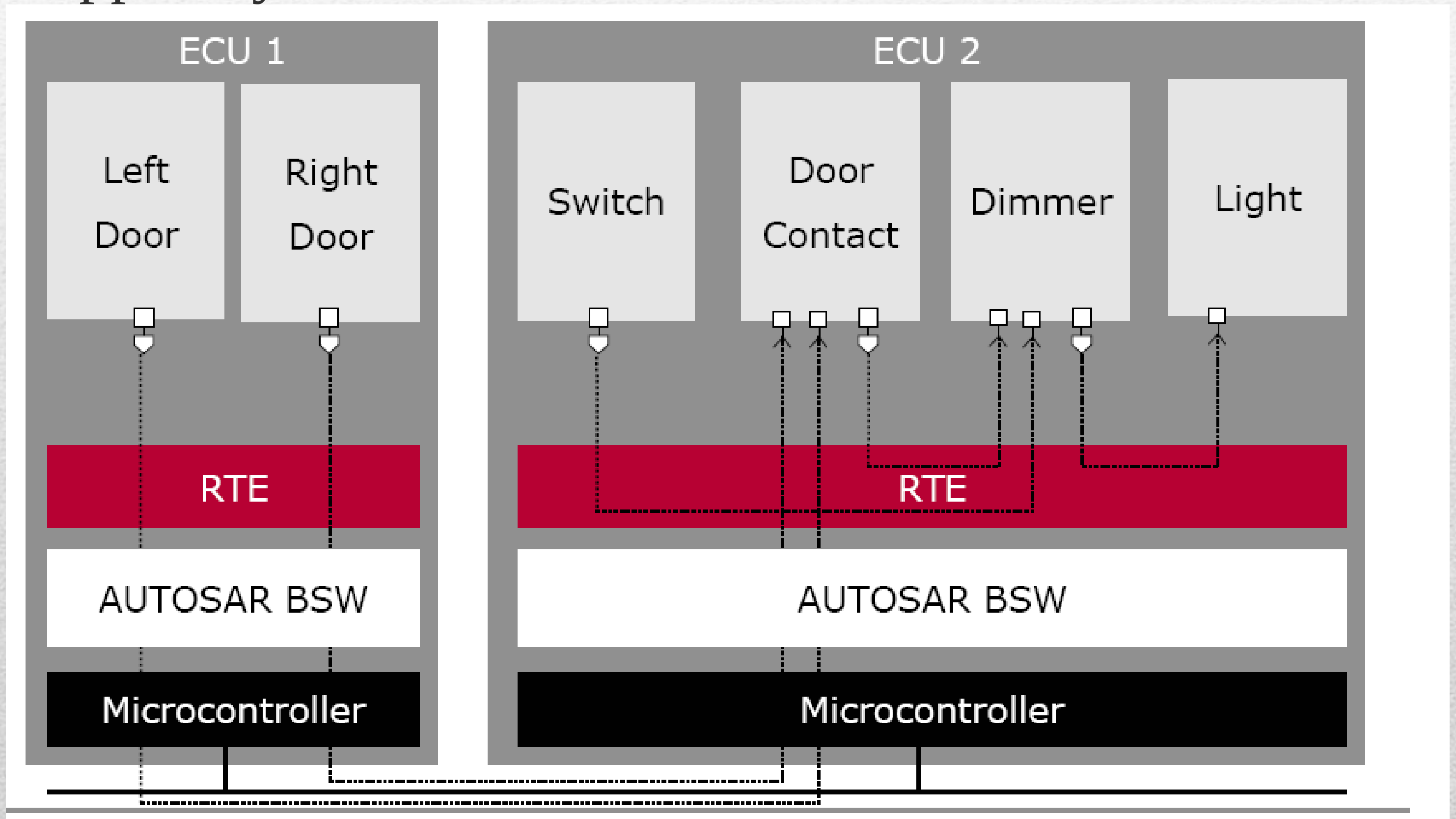
Example AUTOSAR System : Lighting System

- Virtual Functional Bus View



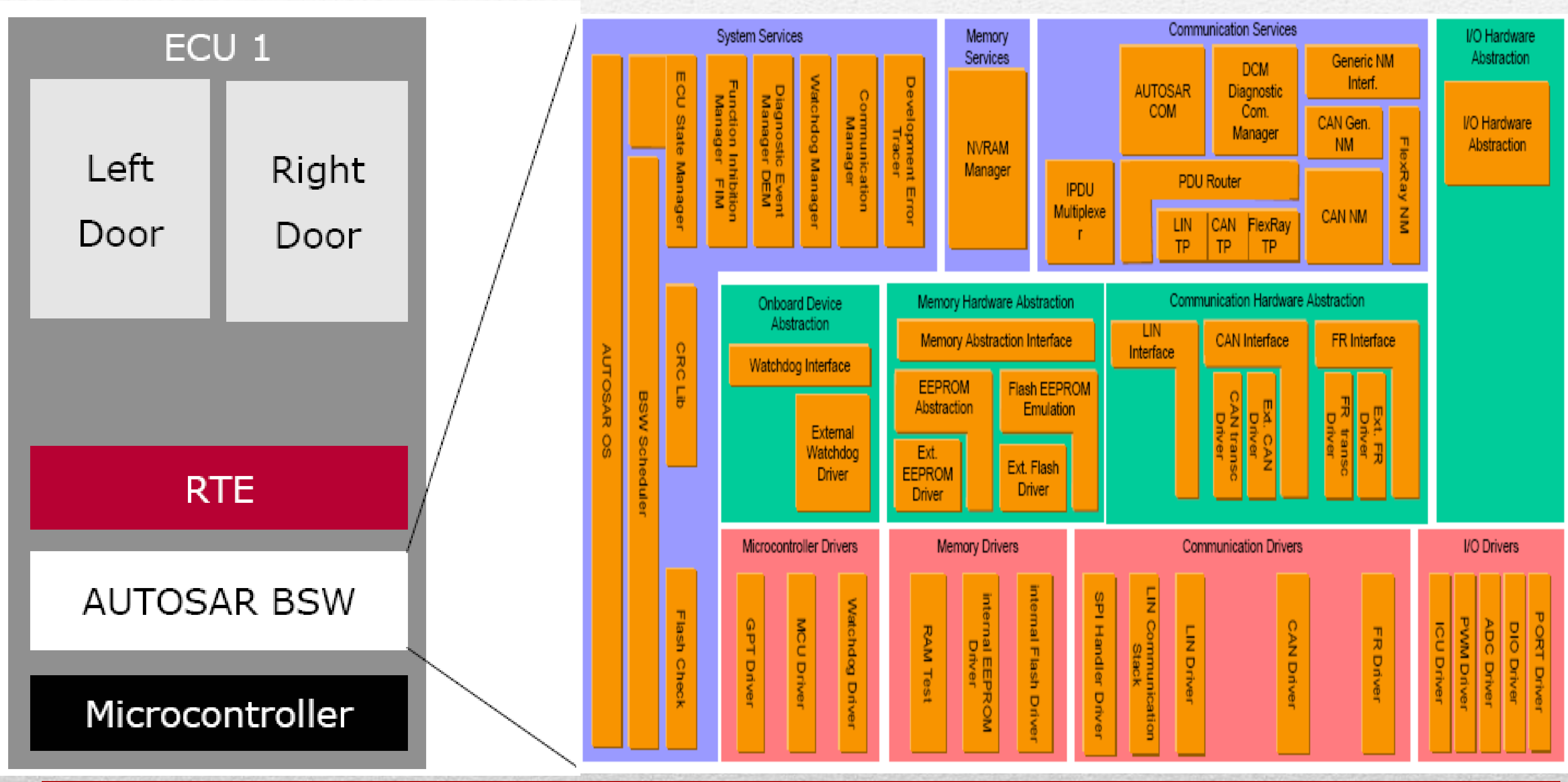
Example AUTOSAR System : Lighting System

- Mapped System



Example AUTOSAR System : Lighting System

- Basic Software Architecture



Contents

- What is AUTOSAR
- Project Objectives & Benefits
- Use case “ Front-Light Management”
- AUTOSAR Main Concepts
  - Architecture
  - Methodology
  - Application Interfaces
- Example of AUTOSAR System
- Conclusion

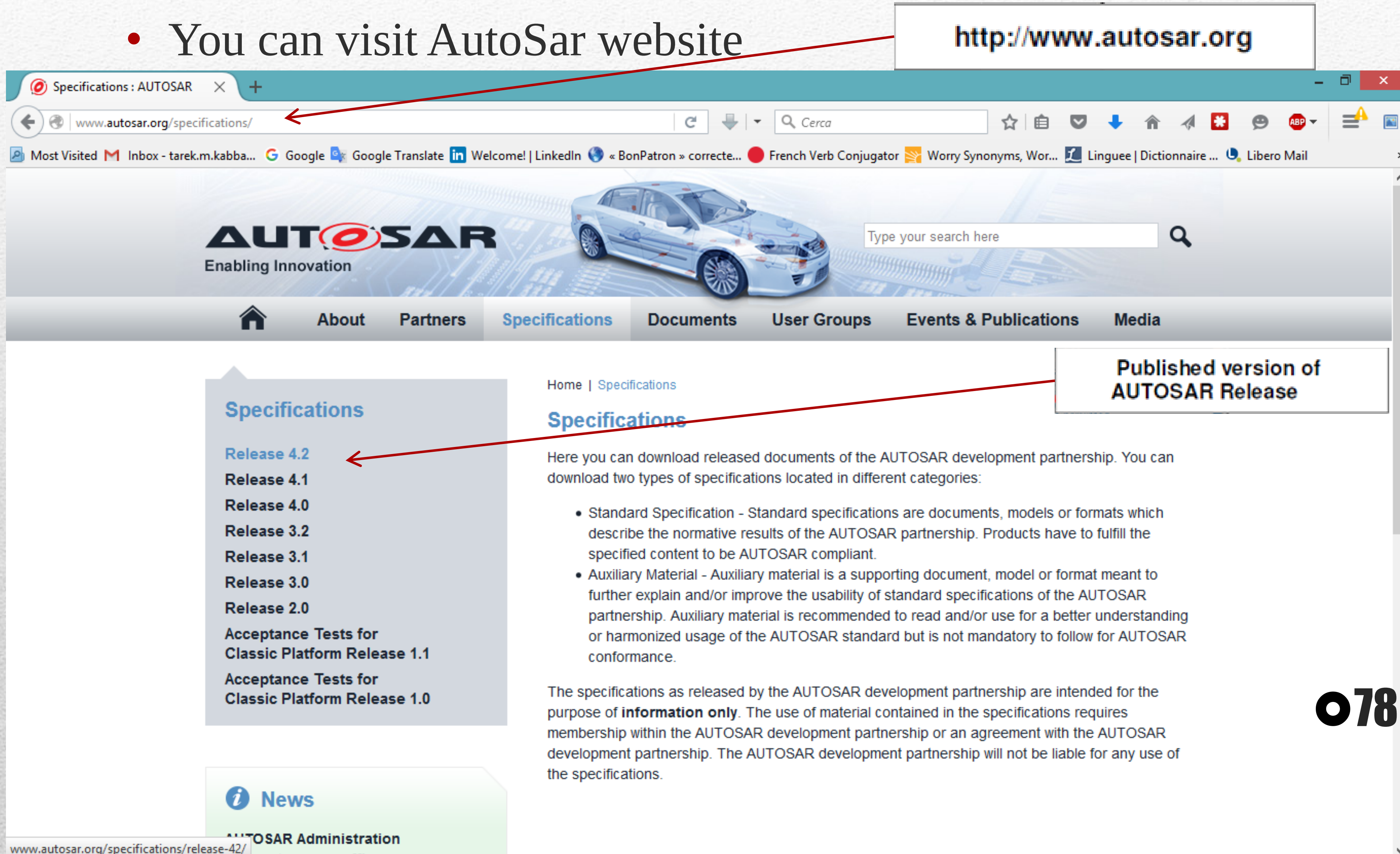
Conclusion

AUTOSAR

- **Leverages model-based engineering of automotive embedded software** to whole systems.
- **Enables management of the growing E/E complexity** with respect to technology and economics.
- Standardization itself is **highly formalized** and so supports formal system development.
- **Shifts implementation efforts to configuration.**
- Pushes the paradigm **shift from an ECU to a function based** approach in automotive software development.
- Through interconnection of subsystems, **new system properties emerge** which have to be understood and controlled.

Further Information

- You can visit AutoSar website



Thank you >>>

Any questions …

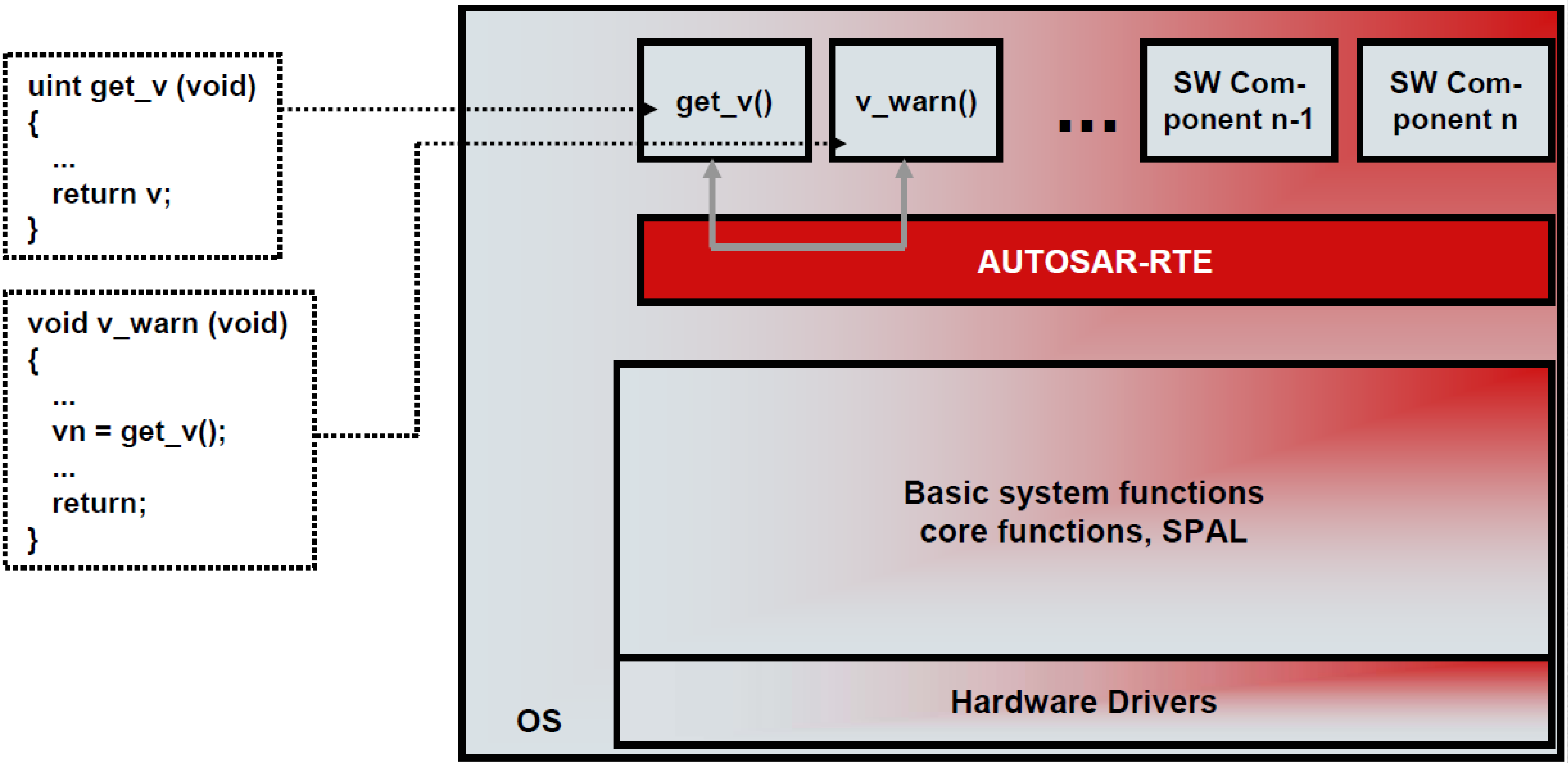
Backup Slides

AUTOSAR –Implementation

AUTOSAR – Implementation (1 of 2)

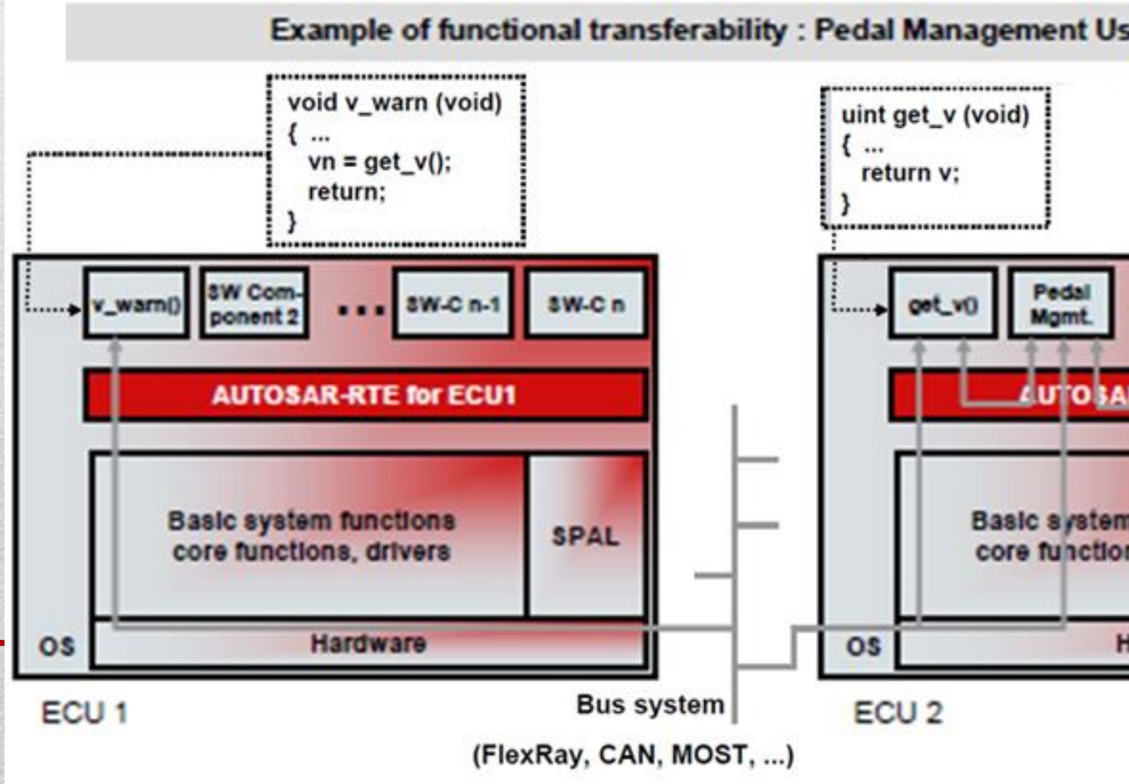
- Implementation of functions independent on distribution on different ECU as communication will be done via ECU-individual AUTOSAR-RTE exclusively

Example: view for one ECU



AUTOSAR – Implementation (2 of 2)

- The ability to transfer functions or SW modules (AutoSar Central Objective: Transferability) supports the following technical benefits
  - Reuse of Intellectual Property (reuse of IP)
  - Increase in design flexibility
  - Simplification of the integration task
  - Reduction of SW development costs



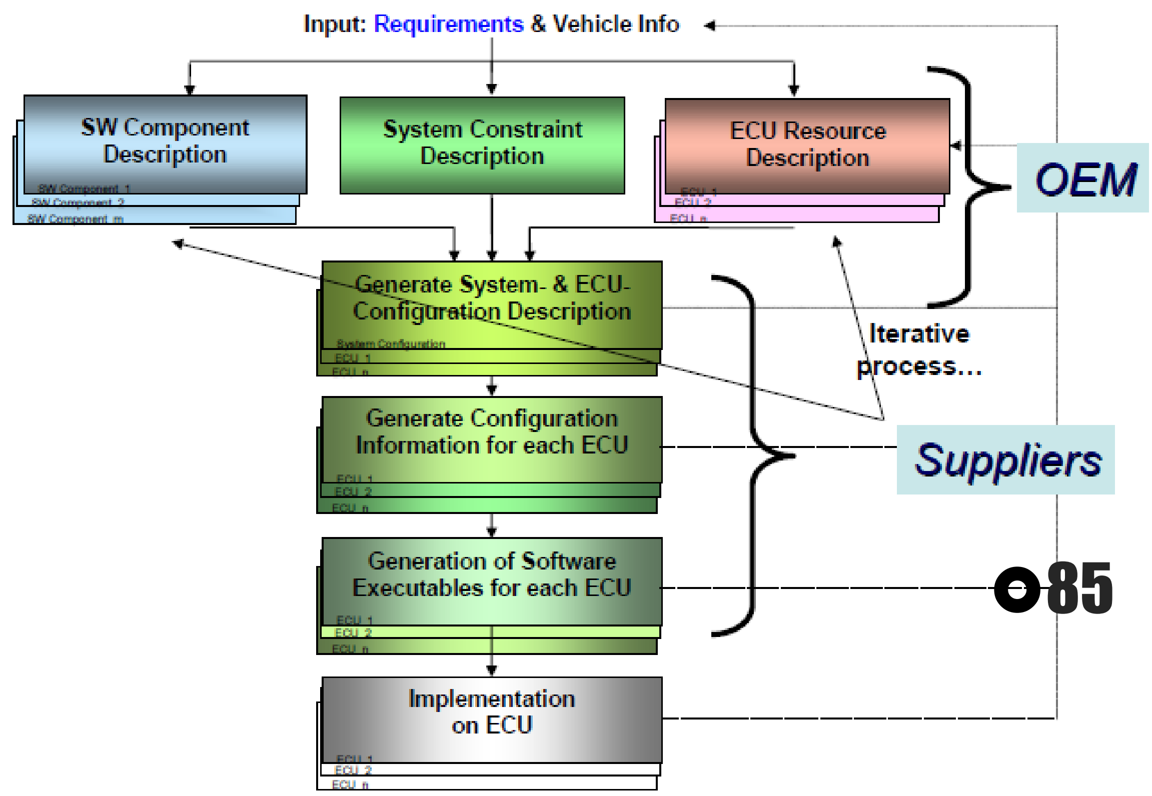
Example: view for two ECU

AUTOSAR Tools

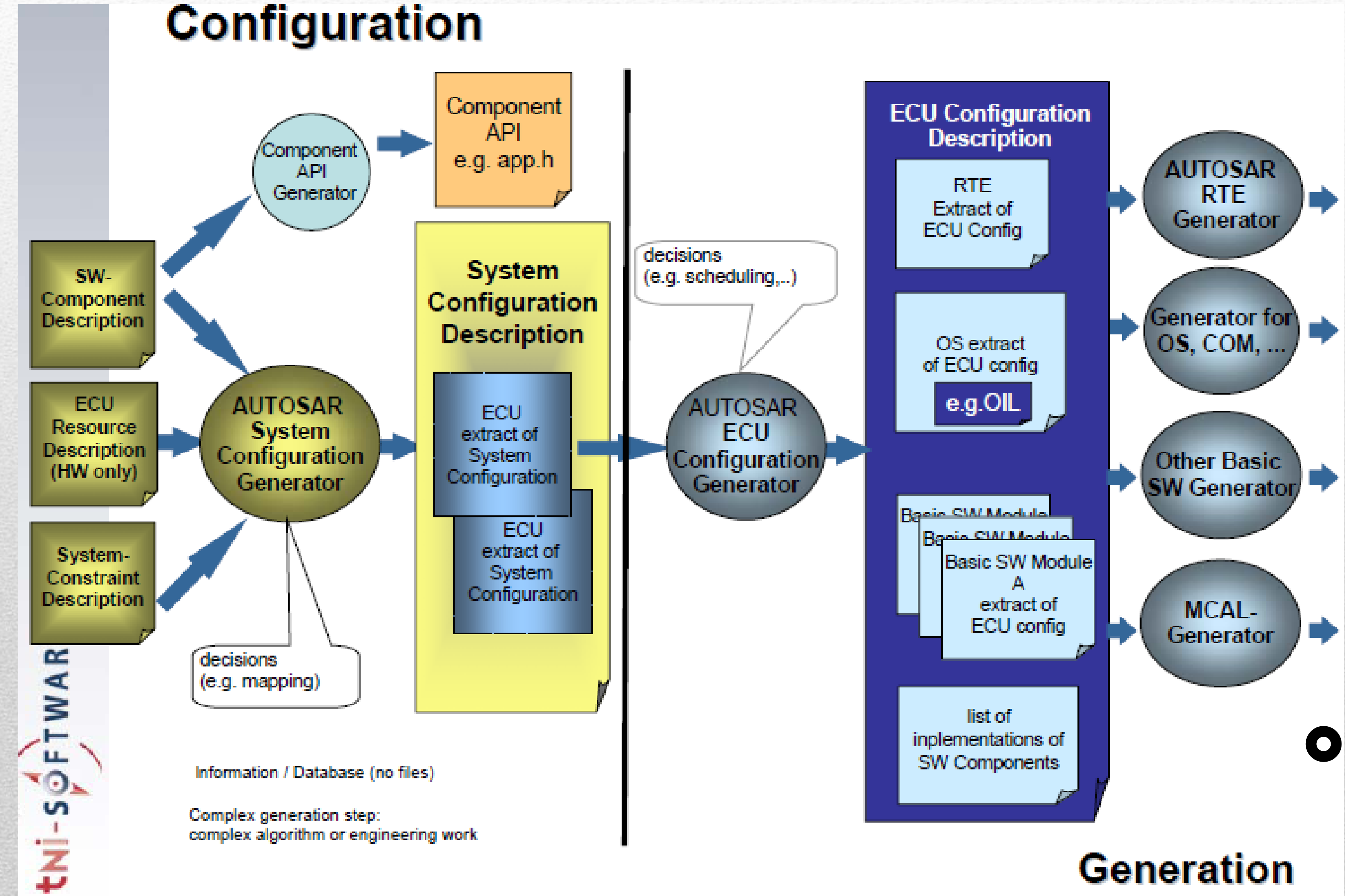
AUTOSAR Tools

AutoSar is not manual - Scope of the standard support

- AutoSar MetaModel: 800 classes based of MOF with stereotypes extensions
- The standardization is based on exchanging XML at every steps
- Must be tooled:
  - Manage (rights, configuration, changes,..)
  - Import
  - Design
  - Validate



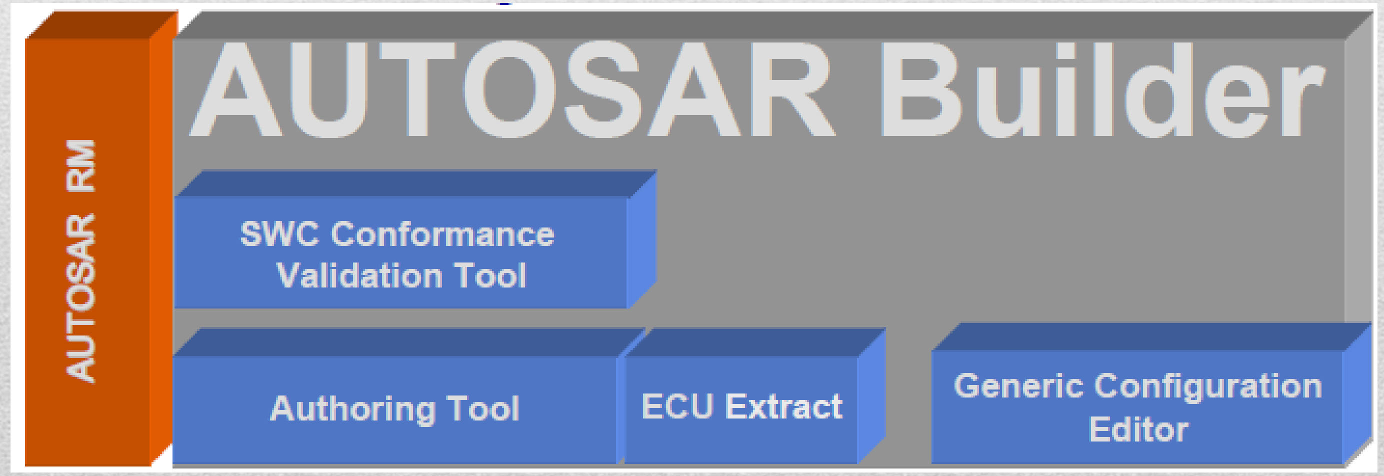
AUTOSAR Tools



AUTOSAR Builder Tool

Components

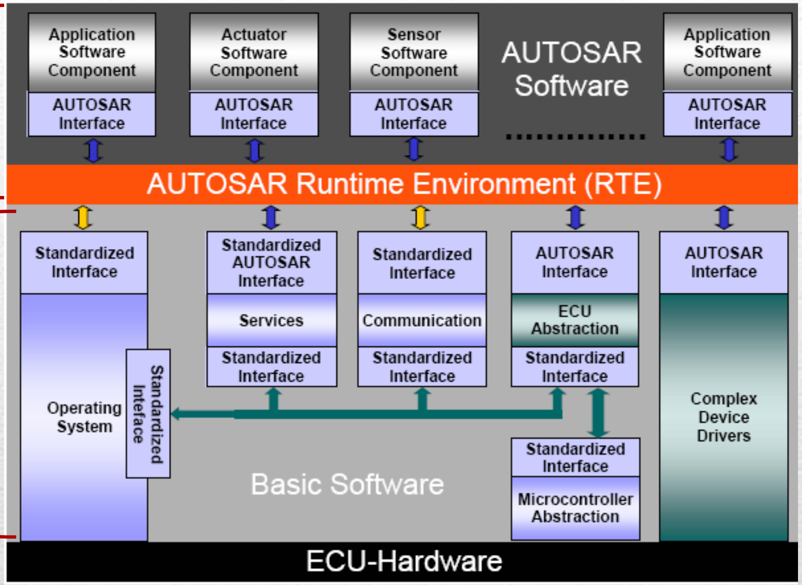
- AutoSar Requirement Management
- AUTOSAR Authoring Tool, **AAT.**
- ECU Extract.
- SWC Conformance Validation Tool, **SCVT.**
- Generic Configuration Editor, **GCE.**



AUTOSAR Builder Tool

Authoring Tool – SWC Conformance Validation

Application Level, Description and Validation



GCE,

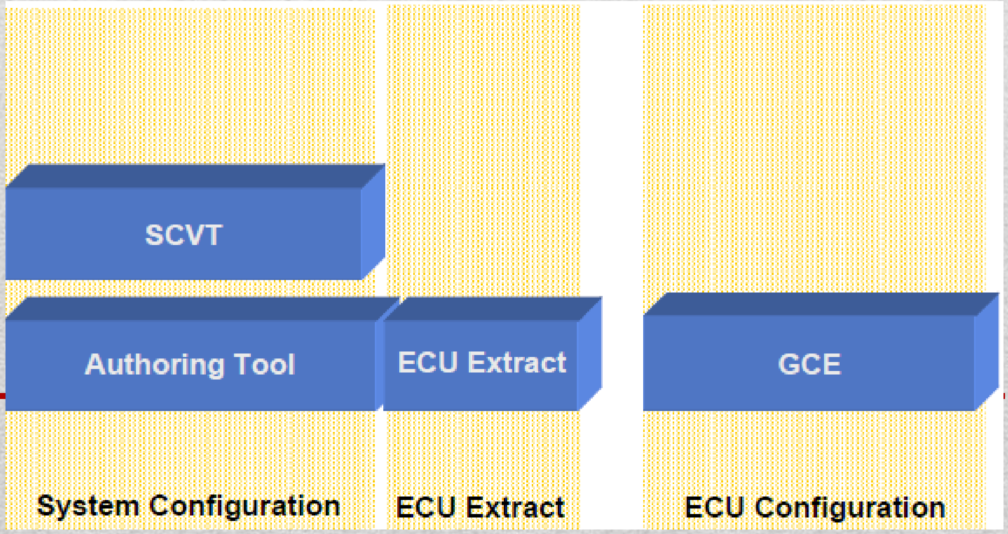
BSW Level,

Configuration

In AutoSar Process

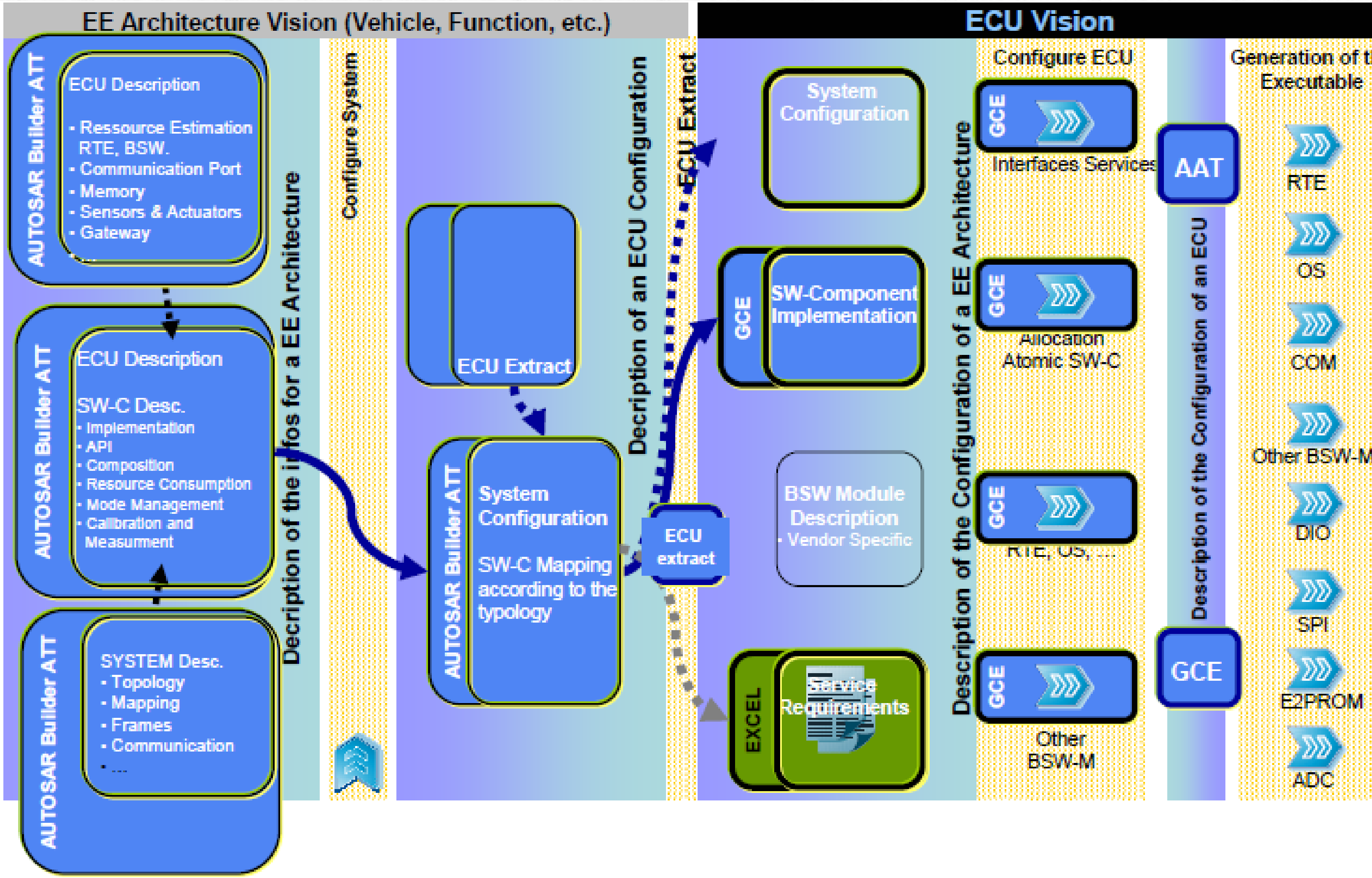
System Configuration & ECU Extract, Description and Validation

ECU Configuration, Configuration

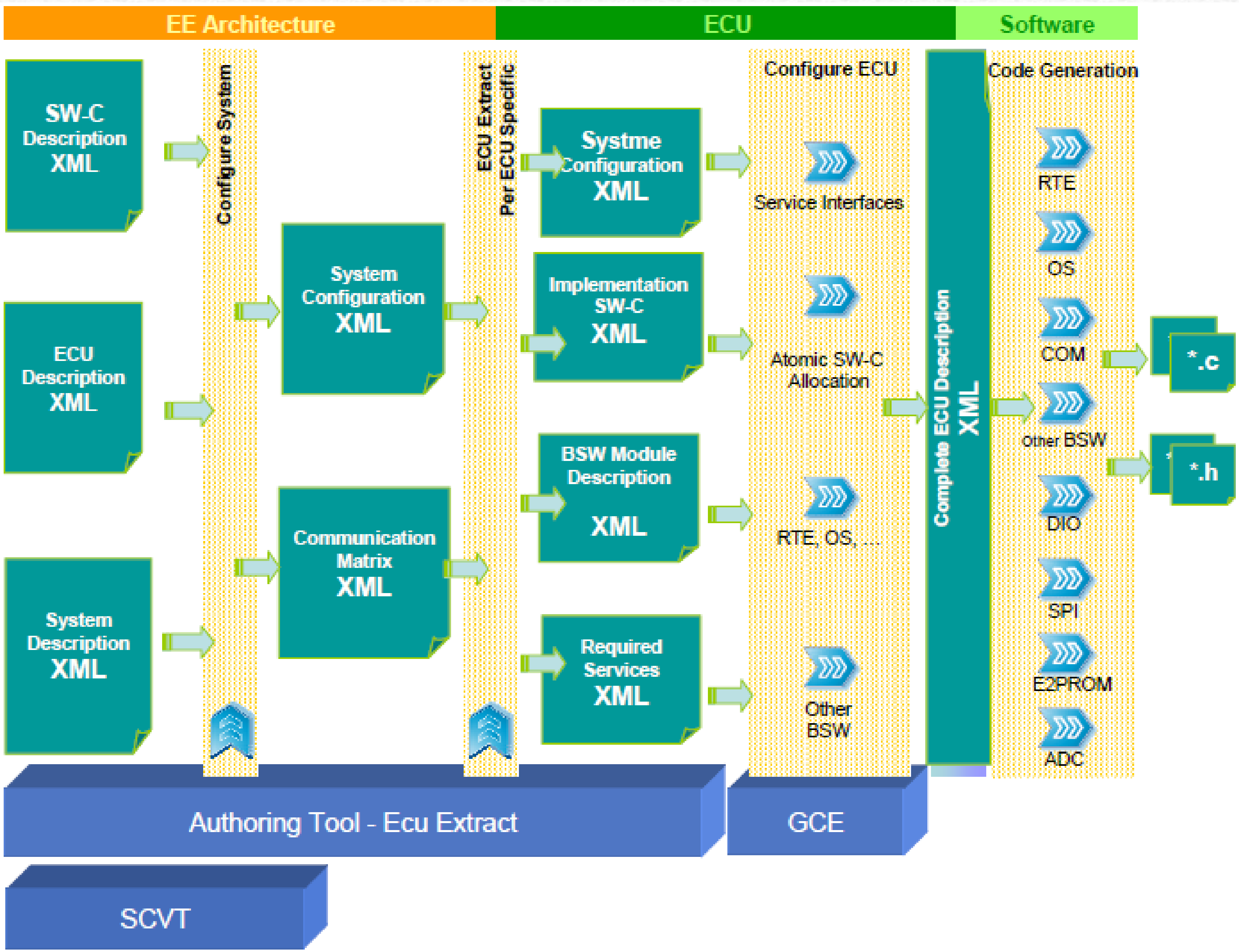


AUTOSAR Tools

- Integration of AUTOSAR Tools in AutoSar Process



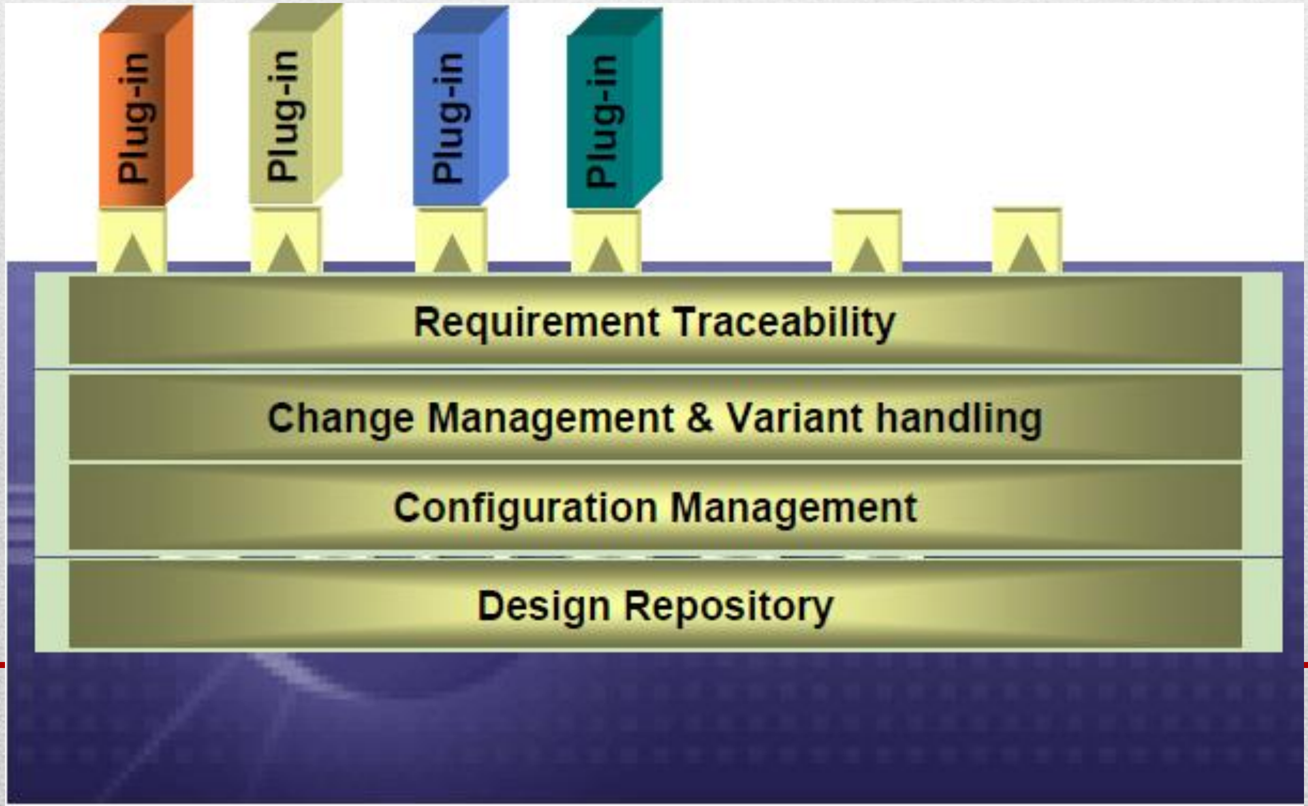
AUTOSAR Tools



AutoSar Builder

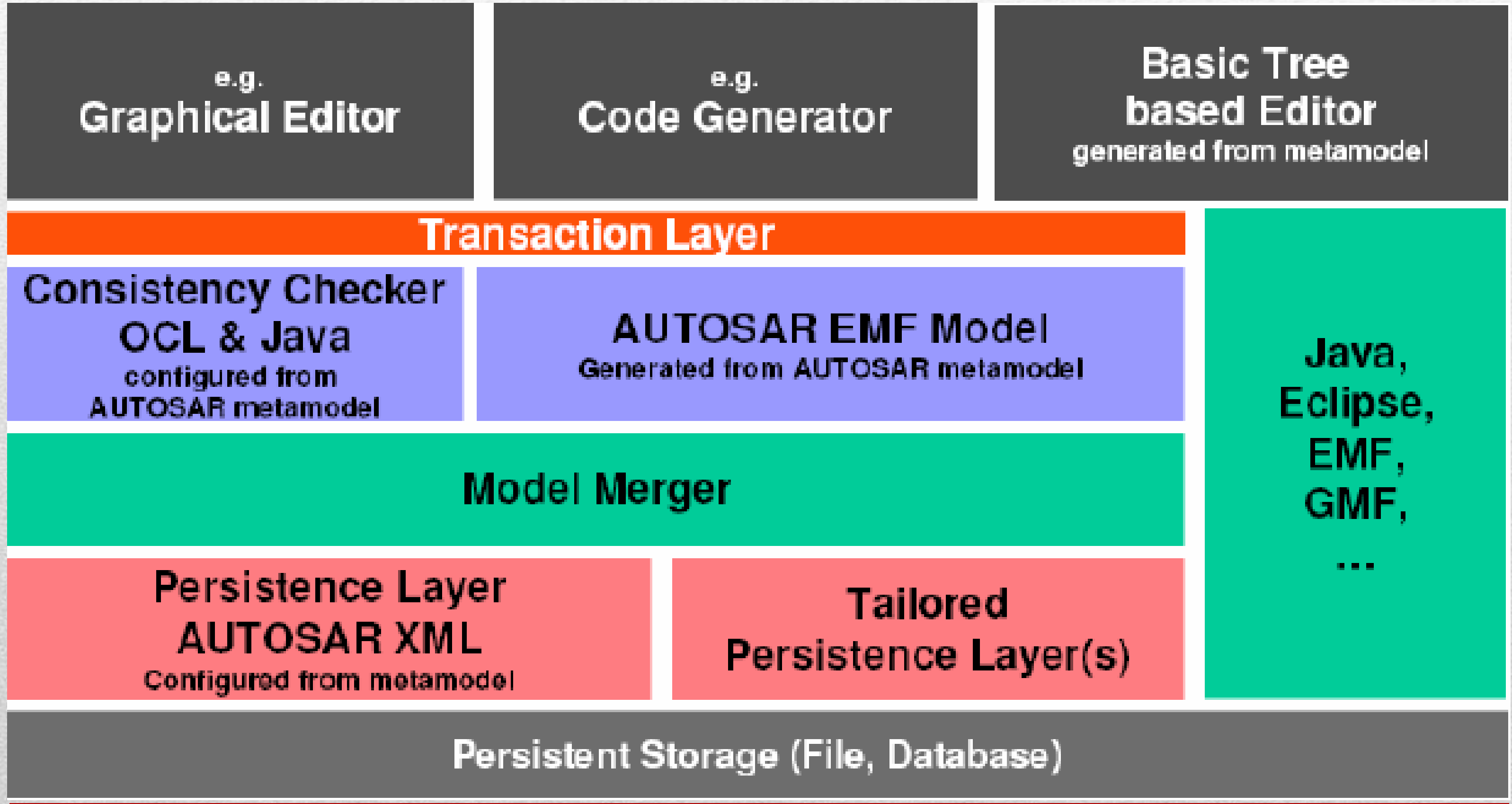
Platform Architecture

- Eclipse Plug-in mechanism
- Leverage on the mature existing tools in the market
- Open Framework adapted to System Engineering



Tool Architecture

- Starting Points: ECLIPSE



Technical infrastructure

- Model management
  - UML/MOF/MDS (model driven schema) to EMF
  - Multi resources support (files, database)
  - Model validation
  - Model extension, AUTOSAR profiles
- Model editing
  - Tree view, forms, XML model of GUI, EMF methods generation
  - Graphical editor, Topcased
- Collaborative support
  - Svn integration
- Documentation
  - Jet & Birt technologies,
  - Creating jet code from the meta model
- Code Generation
  - Jet
- External Tool integration

AUTOSAR – Drivers and Interfaces

AutoSar - Drivers

An driver **contains the functionality to control and access** an internal or an external device.

- **Internal devices** are located inside the microcontroller. Examples for internal devices are:
  - Internal EEPROM
  - Internal CAN controller
  - Internal ADC

A driver for an internal device is called **internal driver** and is located in the MCAL "Microcontroller Abstraction Layer ".

AutoSar - Drivers

- **External devices** are located on the ECU hardware outside the microcontroller. Examples for external devices are:
  - External EEPROM
  - External watchdog
  - External flash

A driver for an external device is called **external driver** and is located in the EAL "ECU Abstraction Layer". It accesses the external device via drivers of the MCAL. This way also components integrated in SBCs (System Basis Chips) like transceivers and watchdogs are supported by AUTOSAR.

Example: a driver for an external EEPROM with SPI interface accesses the external EEPROM via the handler/driver for the SPI bus.

Exception: The drivers for memory mapped external devices (e.g. external flash memory) may access the microcontroller directly. Those external drivers are located in the MCAL because they are microcontroller dependent.

AutoSar - Interface

- An Interface (interface module) contains the functionality to abstract from modules which are architecturally placed below them. E.g., an interface module which abstracts from the hardware realization of a specific device. It provides a generic API to access a specific type of device independent on the number of existing devices of that type and independent on the hardware realization of the different devices.
- The interface does not change the content of the data.
- In general, interfaces are located in the ECU Abstraction Layer.
- Example: an interface for a CAN communication system provides a generic API to access CAN communication networks independent on the number of CAN Controllers within an ECU and independent of the hardware realization (on chip, off chip).

AutoSar – Handler and Manager

<u>A handler</u>

- Is a specific interface which controls the concurrent, multiple and asynchronous access of one or multiple clients to one or more drivers. i.e. it performs buffering, queuing, arbitration, multiplexing.
- Does not change the content of the data.
- Functionality is often incorporated in the driver or interface (e.g. SPIHandlerDriver, ADC Driver).

<u>A manager</u>

- offers specific services for multiple clients. It is needed in all cases where pure handler functionality is not enough to abstract from multiple clients.
- Besides handler functionality, a manager can evaluate and change or adapt the content of the data.
- In general, managers are located in the Services Layer

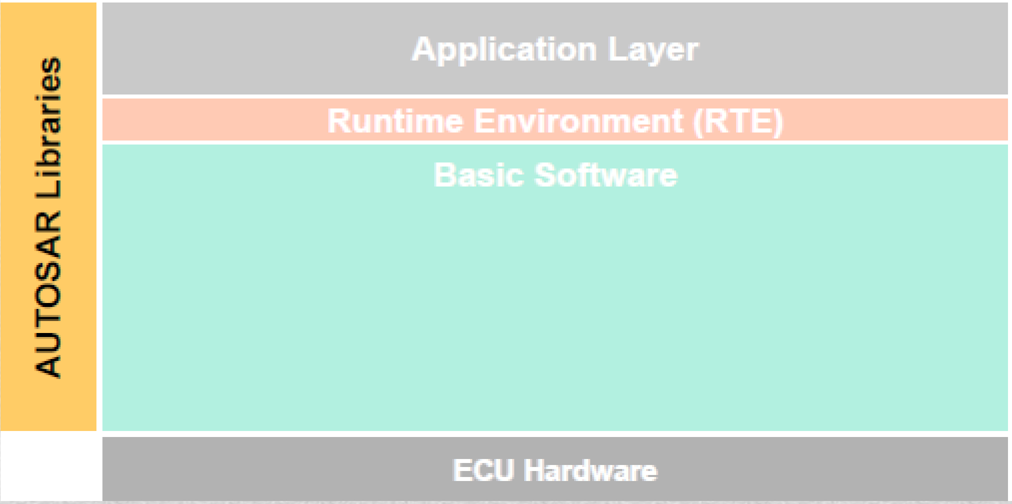
Example: The NVRAM manager manages the concurrent access to internal and/or external memory devices like flash and EEPROM memory. It also performs distributed and reliable data storage, data checking, provision of default values etc.

AutoSar - Libraries

Libraries are a collection of functions for related purposes. They:

- Can be called by BSW modules (that including the RTE), SW-Cs, libraries or integration code
- run in the context of the caller in the same protection environment
- can only call libraries
- are re-entrant
- do not have internal states
- do not require any initialization
- are synchronous, i.e. they do

not have wait points



The following libraries are specified within AUTOSAR:

- Fixed point mathematical,
- Floating point mathematical,
- Interpolation for fixed point data,
- Interpolation for floating point data,
- Bit handling,
- E2E communication,
- CRC calculation,
- Extended functions (e.g. 64bits calculation, filtering, etc.) and
- Crypto

AUTOSAR - Methodology and Application Interfaces Use cases

Sensor/Actuator AUTOSAR SW-C

The **Sensor/Actuator AUTOSAR Software Component** is a specific type of AUTOSAR Software Component for sensor evaluation and actuator control. Though not belonging to the AUTOSAR Basic Software, it is described here due to its strong relationship to local signals. It has been decided to locate the Sensor/Actuator SW Components above the RTE for integration reasons (standardized interface implementation and interface description). Because of their strong interaction with raw local signals, relocatability is restricted.

Task:

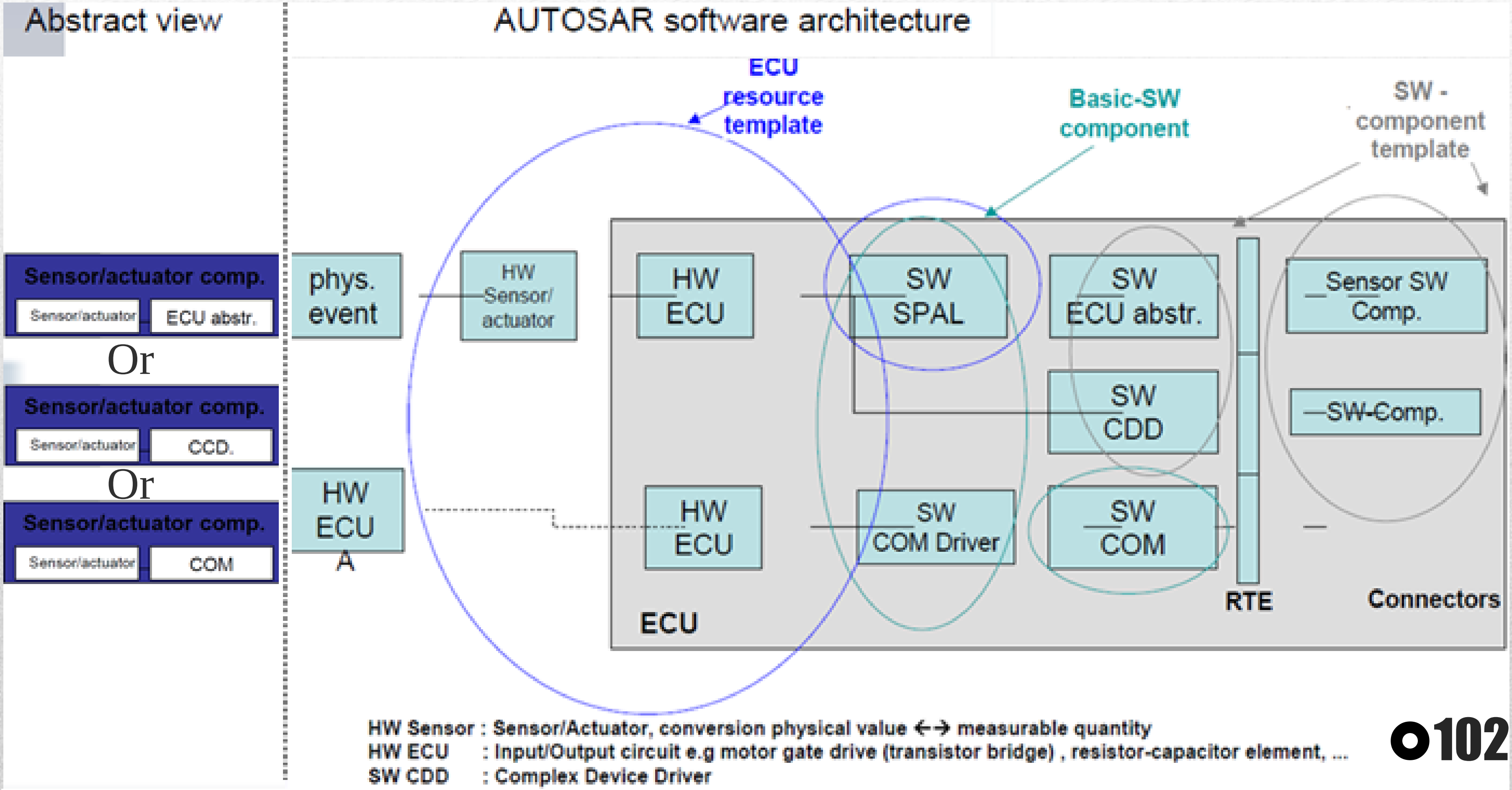
Provide an abstraction from the specific physical properties of hardware sensors and actuators, which are connected to an ECU.

Properties:

Implementation: µC and ECU HW independent, sensor and actuator dependent

Main Concepts: Methodology

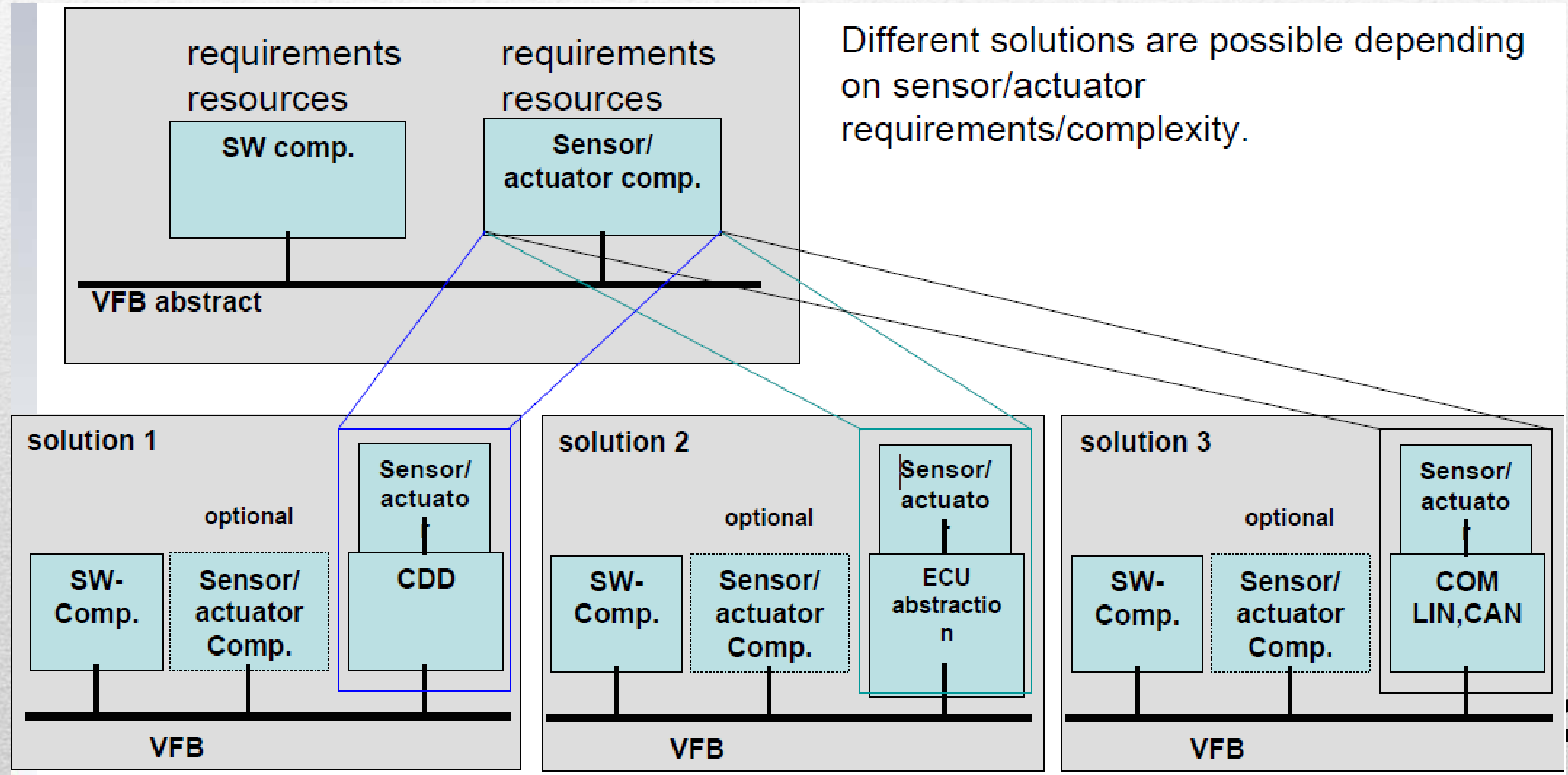
- AUTOSAR – Assignment of Basic SW Components



Main Concepts: Methodology

- AUTOSAR – System View

Sensor/Actuator Component Implementation possibilities

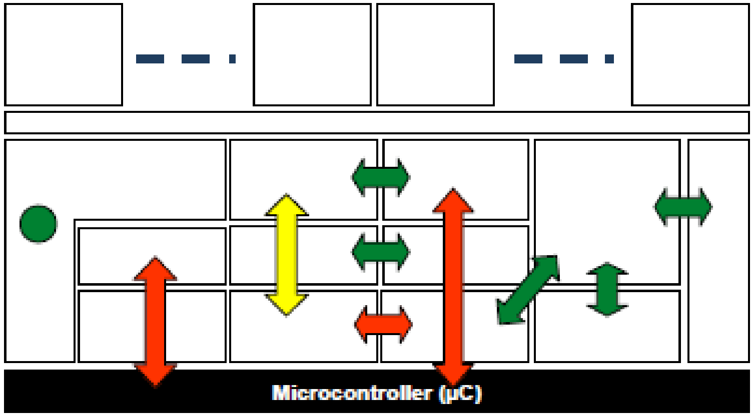


Interfaces: General Rules

- **General Interfacing Rules**

Horizontal Interfaces

- Services Layer: horizontal interfaces are allowed
  Example: Error Manager saves fault data using the NVRAM manager
- ECU Abstraction Layer: horizontal interfaces are allowed
- A complex driver may use selected other BSW modules
- µC Abstraction Layer: horizontal interfaces are not allowed. Exception: configurable notifications are allowed due to performance reasons.



Vertical Interfaces

- One Layer may access all interfaces of the SW layer below
- Bypassing of one software layer should be avoided
- Bypassing of two or more software layers is not allowed
- Bypassing of the µC Abstraction Layer is not allowed
- A module may access a lower layer module of another layer group (e.g. SPI for external hardware)
- All layers may interact with system services.

Interfaces: General Rules

- **Layer Interaction Matrix**

This matrix shows the possible interactions between AUTOSAR Basic Software layers

✓ "is allowed to use"
✗ "is not allowed to use"
Δ "restricted use (callback only)"

The matrix is read **row-wise**:
Example: "I/O Drivers are allowed to use System Services and Hardware, but no other layers".

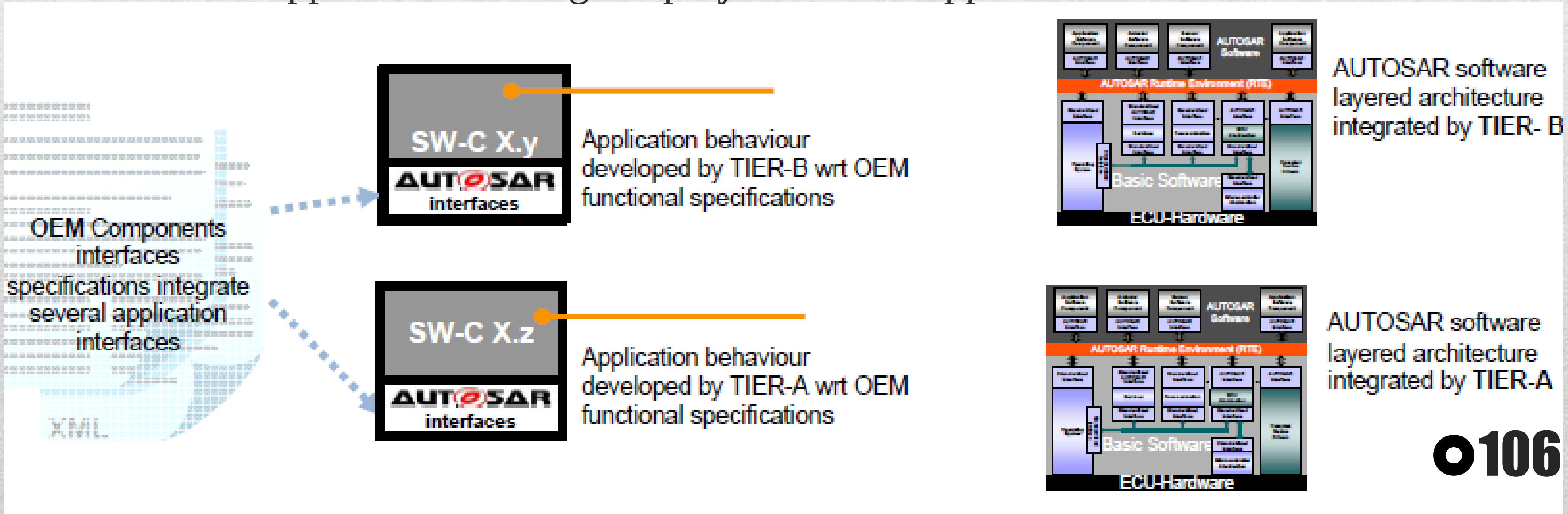
(gray background indicates "non-Basic Software" layers)

uses

	System Services	Memory Services	Communication Services	Complex Drivers	I/O Hardware Abstraction	Onboard Device Abstraction	Memory Hardware Abstraction	Communication Hardware Abstraction	Microcontroller Drivers	Memory Drivers	Communication Drivers	I/O Drivers
AUTOSAR SW Components / RTE	✓	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
System Services	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓
Memory Services	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Communication Services	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗
Complex Drivers	restricted access -> see the following two slides											
I/O Hardware Abstraction	✓	✗	✗	✗	✓	✓	✗	✓	✓	✗	✓	✓
Onboard Device Abstraction	✓	✗	✗	✗	✗	✓	✗	✓	✓	✗	✓	✓
Memory Hardware Abstraction	✓	✓	✗	✗	✗	✓	✓	✓	✗	✓	✓	✗
Communication Hardware Abstraction	✓	✗	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓
Microcontroller Drivers	✓	✗	✗	✗	✓	✓	✗	✗	Δ	✗	✗	Δ
Memory Drivers	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Communication Drivers	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓
I/O Drivers	✓	✗	✗	✗	✓	✓	✗	✗	Δ	✗	✗	Δ

AUTOSAR Application Interfaces - OEM Use case

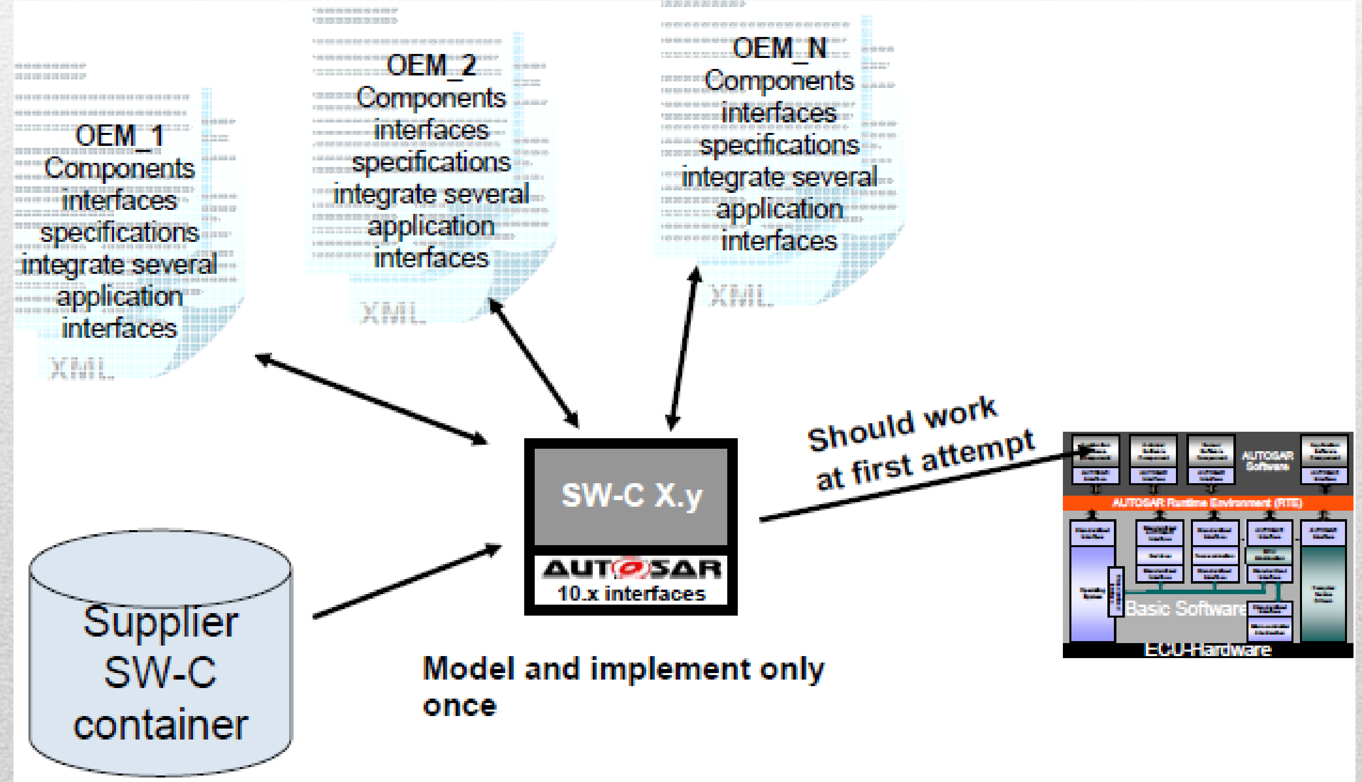
- SHORT TERM: OEM is applying AUTOSAR Naming Convention more than 10.000 interfaces and calibrations data for industrial purposes after two years of intensive work on the specification of the naming convention
- Middle Term: Results are foreseen as an "AUTOSAR Application Interfaces Handbook" to support 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.

AUTOSAR Application Interfaces - 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.

AUTOSAR –challenges

Challenges in Automotive E/E Development

- Extend product offering and increase product differentiation
  - Stable or decreasing development costs
- Strengthen brand image in the market
  - Propose specific features and functions across the product range
- Ensure long term competitiveness, as well as presence in emerging markets, through cost reduction
- Increase quality and reduce “non quality” costs
- Increasing share of electronics in vehicle value
  - Electronics share (in value): 2004: 20% 2015: 40% (McKinsey, Automotive Electronics - Managing innovations on the road)
  - Software share (in value): 2000: 4,5% 2010: 13% (Mercer Consulting, Automobile technologie 2010)

AutoSar Evolution

Architecture Evolution

- The basic software architecture has reached a high level of maturity.
- Commercial implementations of the basic software modules based on Release 3.0 as well as 2.1 are available on the market.
- Major improvements were made on the wake up and start up of ECUs and networks providing both, harmonization of features and reduction of complexity.
- As an example of evolution of existing modules the approach of abstraction was refined by introducing state managers as an architectural layer for CAN (Controller Area Network), LIN (Local Interconnection Network), and FlexRay

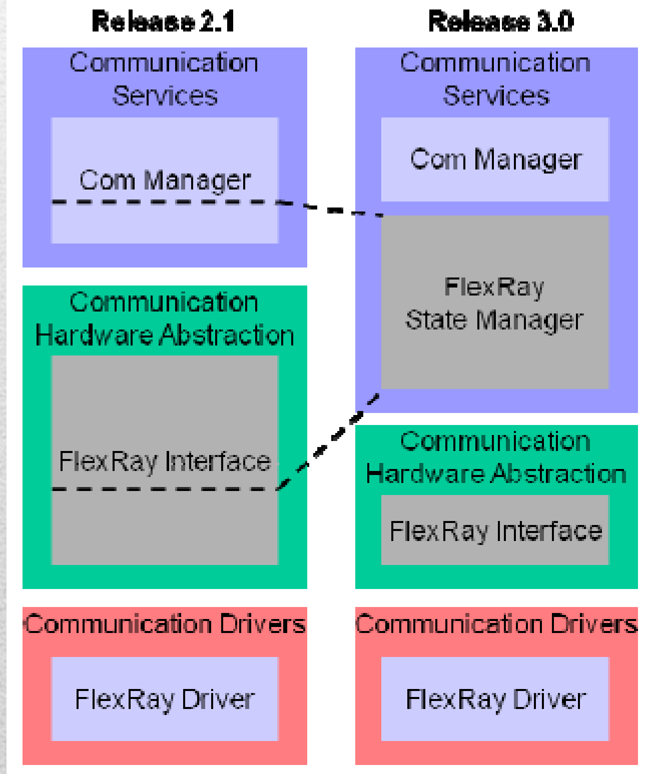
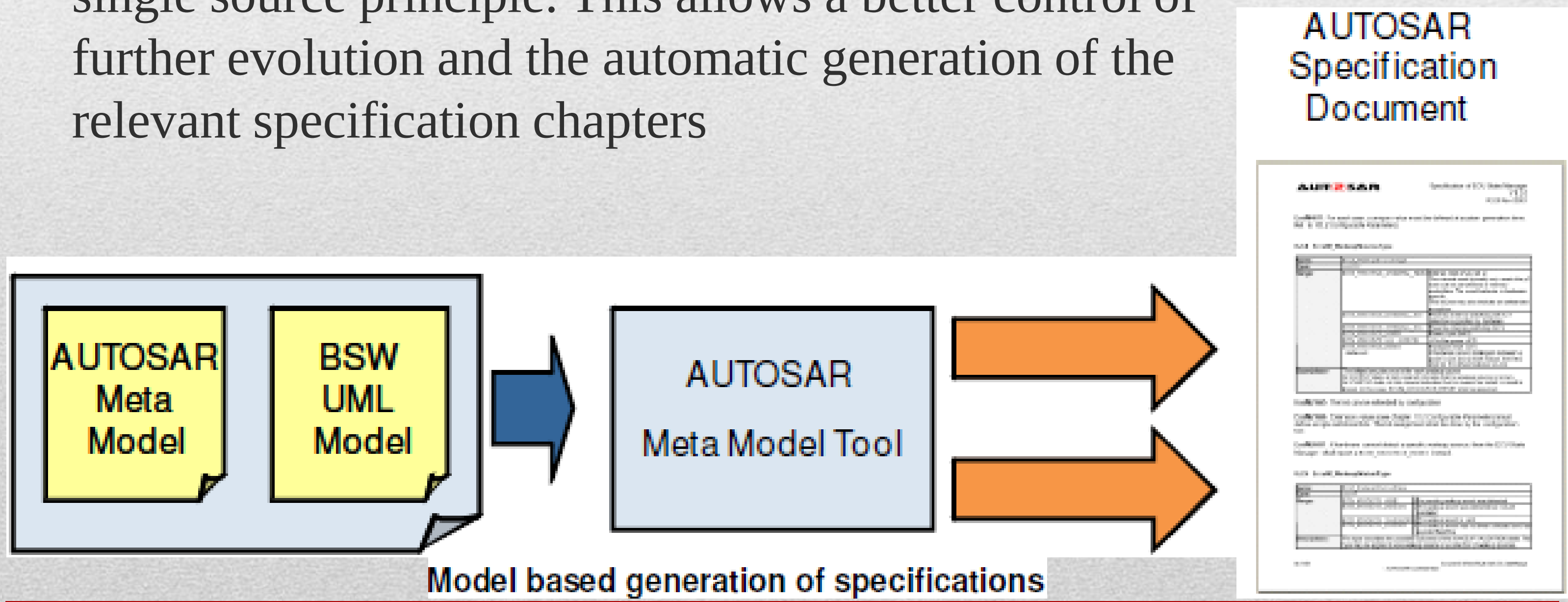


Figure 3: Evolution of the communication stack in Release 3.0, for example FlexRay

Methodology and Templates Evolution

- The improvements made on the templates ensure the consistency of the standard. Interfaces, behavior and configuration parameters of the basic software are now included in AUTOSAR models – following the single source principle. This allows a better control of further evolution and the automatic generation of the relevant specification chapters



Model based generation of specifications

Release 4 - Functional safety

- Functional safety is one of the main objectives as AUTOSAR will support safety related applications and thereby has to consider the upcoming ISO 26262 standard. Exemplarily some of the new safety features are mentioned below:
  - The memory partitioning concept will provide a fault containment technique to separate software applications from each other. This concept is allowing safety and non safety applications to be implemented on the same ECU.
  - Defensive behavior is a solution that prevents data corruption and wrong service calls on microcontrollers which have no hardware support for memory partitioning.
  - Support for dual microcontroller architectures aims on detection of faults in the core microcontroller by a secondary unit.
  - Program flow monitoring controls the temporal and logical behavior of applications by checking, at specified points of code execution, if the timing and logical order of execution requirements are met.
  - The end-to-end communication protection library is providing a state of the art safety protocol at application level.

Standardization Levels

The standardization could be developed incrementally towards:

- Level of abstraction
  - Functional aspects
  - Behavior and implementation aspects
- Level of decomposition
  - Low degree of decomposition of the functional domain
  - High degree of decomposition of the functional domain
- Level of architecture definition
  - Terminology
  - Standardized data-types
  - Partial description of interfaces (without semantics)
  - Complete description of interfaces (without semantics)
  - Complete description of interfaces (with semantics)
  - Partial definition of the functional domain
  - Complete definition of the functional domain

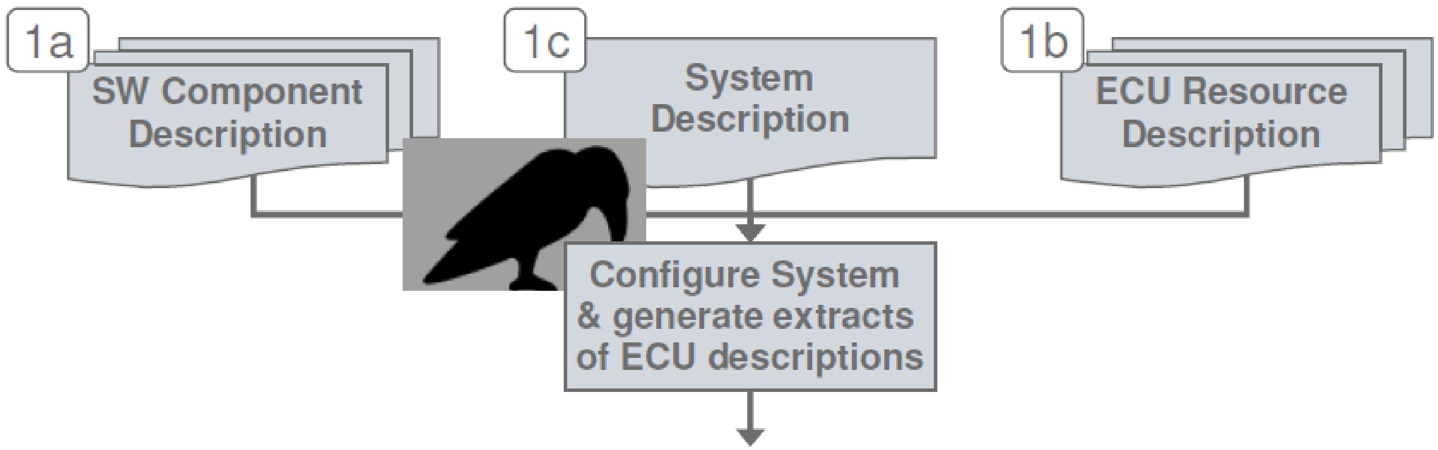
AUTOSAR – Topics for Research and Development 2008

Graph Transformation

- Is the technology for semi-automatic configuration
- Can reduce the configuration complexity
- Needs to build domain knowledge

Topics for Research and Development

- System configuration



Step 2 “System Configuration” has high complexity

Contains mappings of

Data → Signals → Network Communication

Implementations ← SW-compositions → ECU

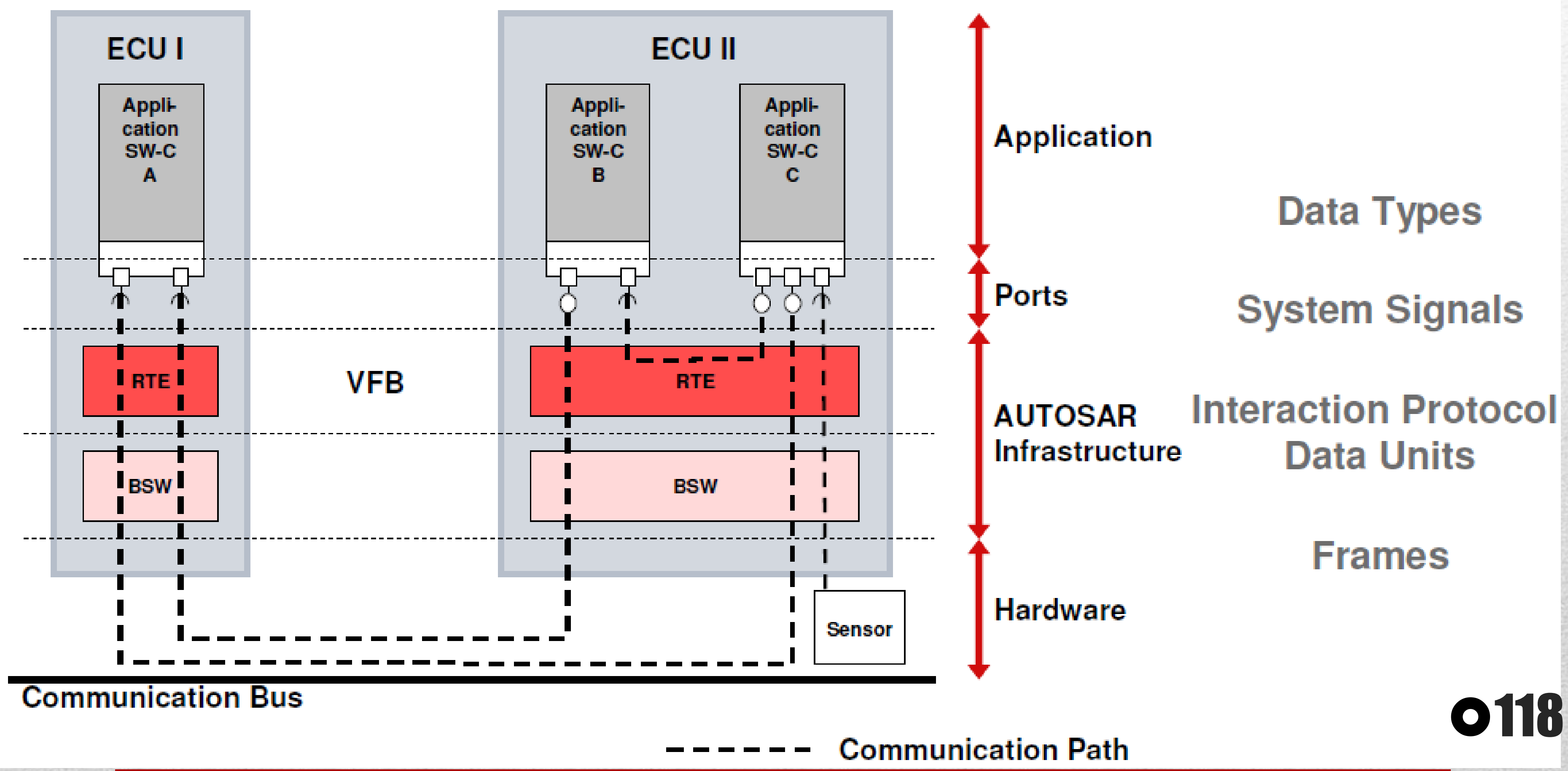
Logical HW resources → ECU

Under the existence of mappings constraints

System configuration is data structure covering the whole system description

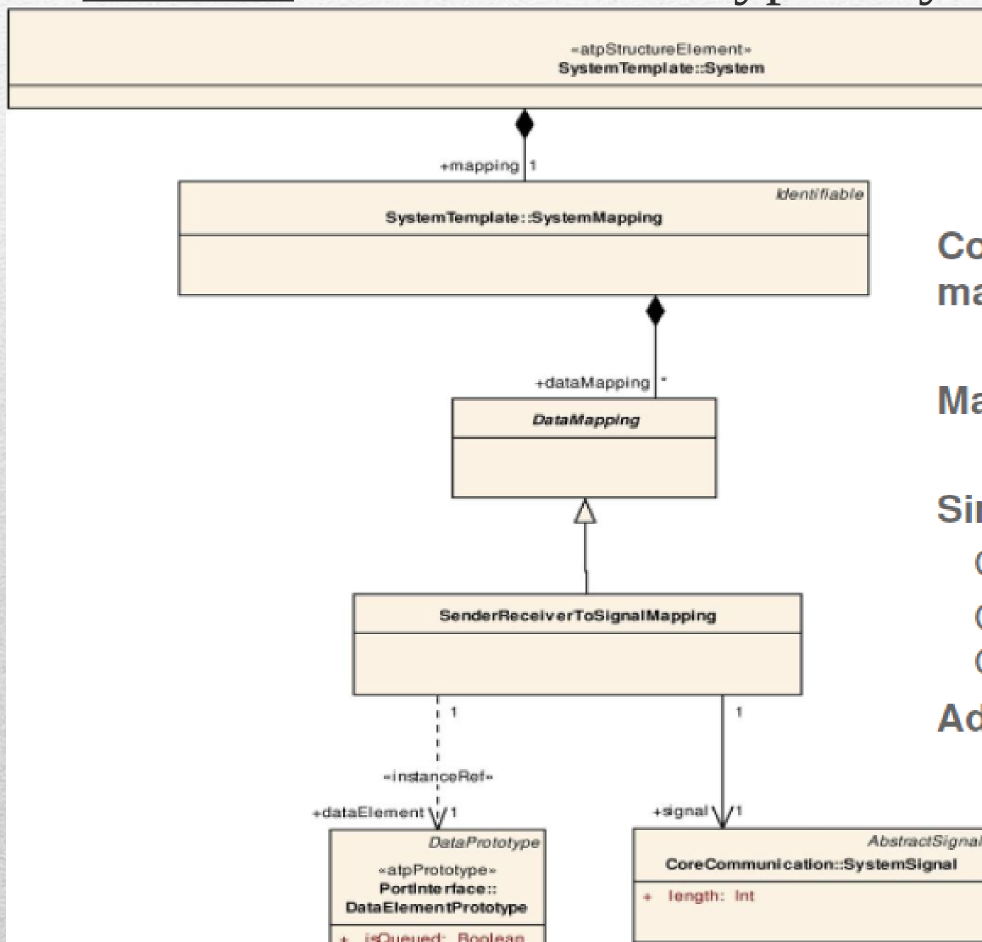
Topics for Research and Development

- System configuration/Communication Mapping.



Topics for Research and Development

- System configuration/Communication Mapping.
- Level 1: Primitive Data Types System Signals.



Communication between SW Components mapped onto Run Time Environment

Mapping defined as set of associations

Similar, but more complex mappings for

- Complex data types
- Client-server communication between SW-Components

Additionally, signal paths can be constraint

Topics for Research and Development

System configuration/

Communication Mapping.

<u>Level 2:</u>

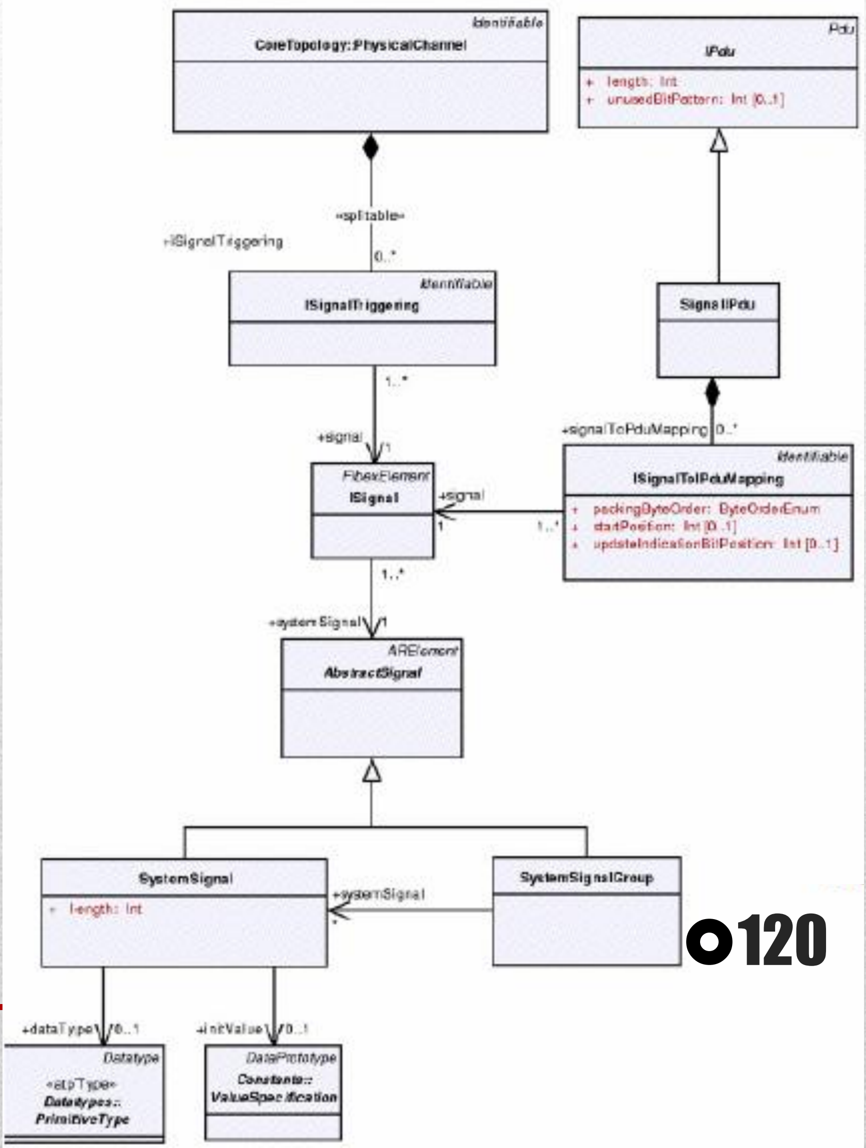
System Signals

→ Run Time Environment

→ Interaction Protocol Data Unit

- Mapping of RTE signals to Communication Manager
- Interaction layer defines also

timing and triggering of ISignals



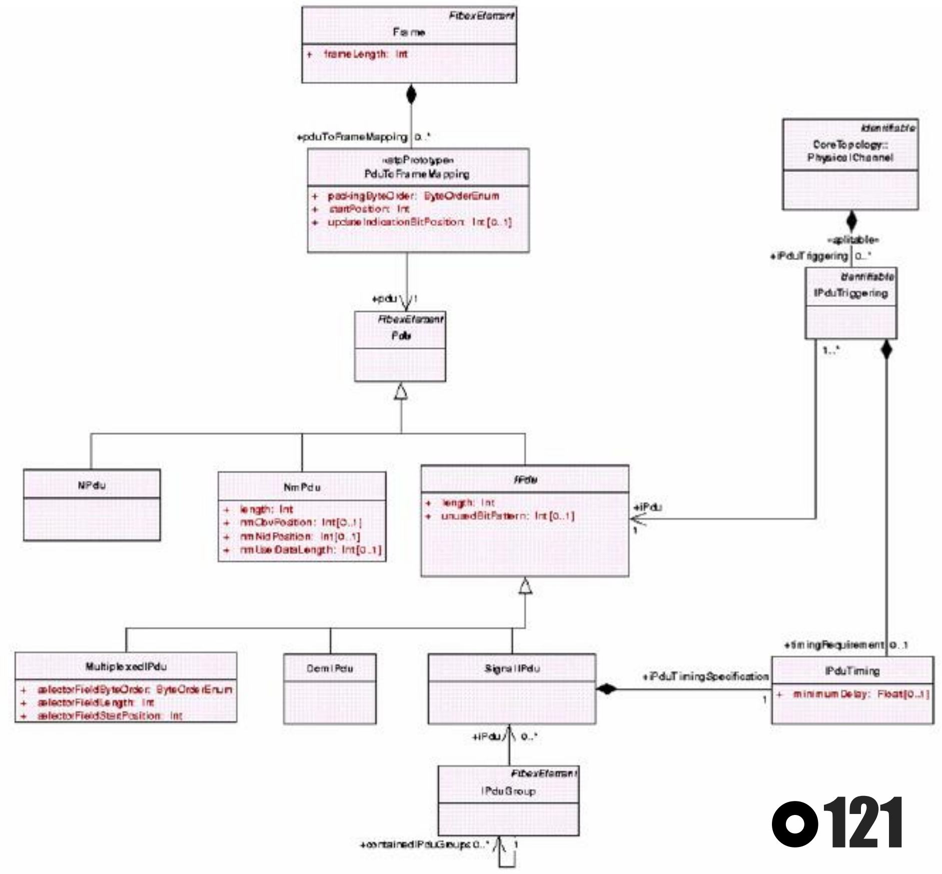
Topics for Research and Development

System configuration/

Communication Mapping

Level 3: Interaction Protocol Data Units → Frames.

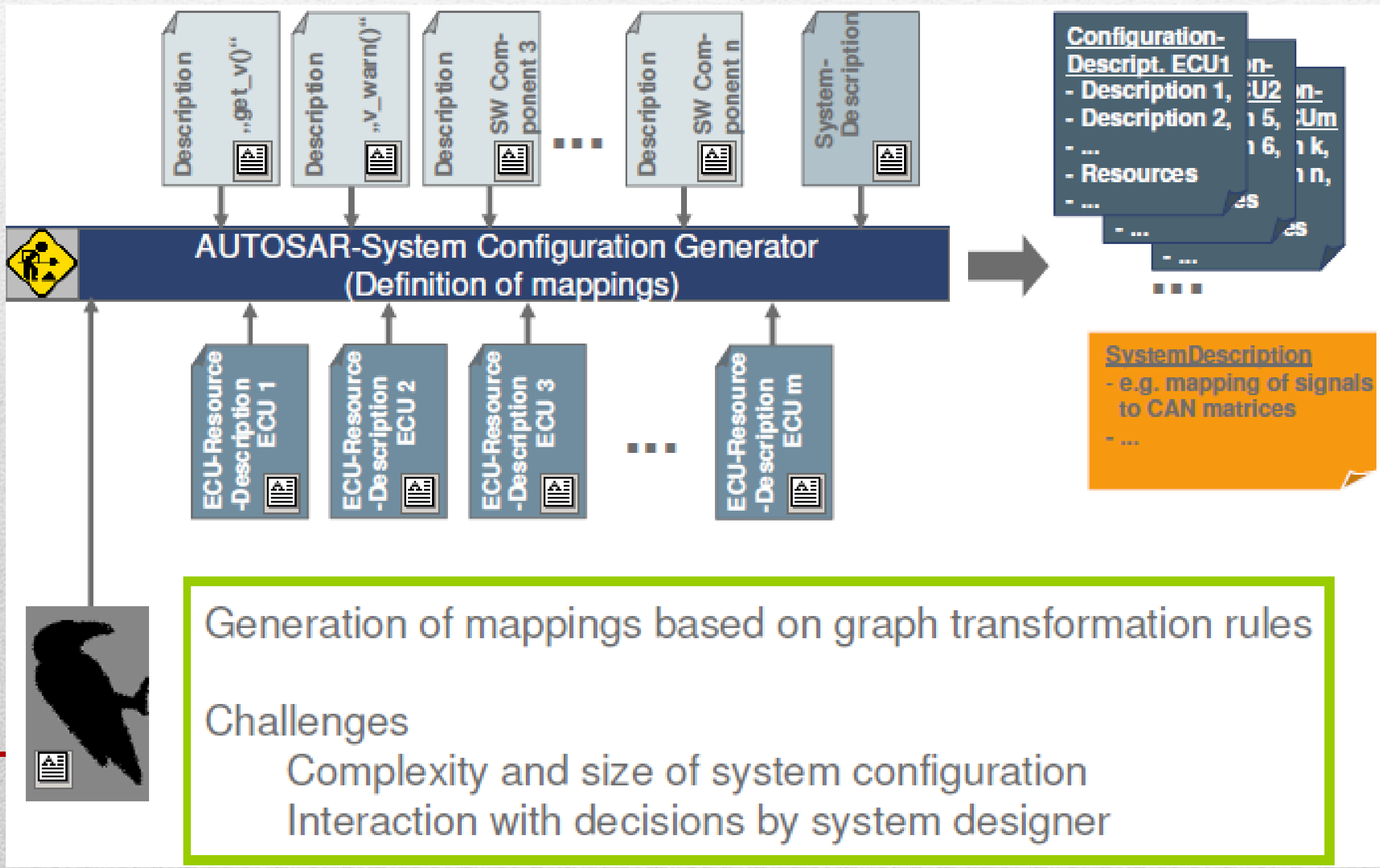
- Mapping of Communication Manager to PDU Router
- PDU Router deploys frames to different communication protocols
- Frame definitions configure all communication stacks of full network
  - Different segments of system configuration will be used to configure each communication stack at each ECU



Topics for Research and Development

System configuration/Communication Mapping/Tooling.

Semi-automatic mapping of communication

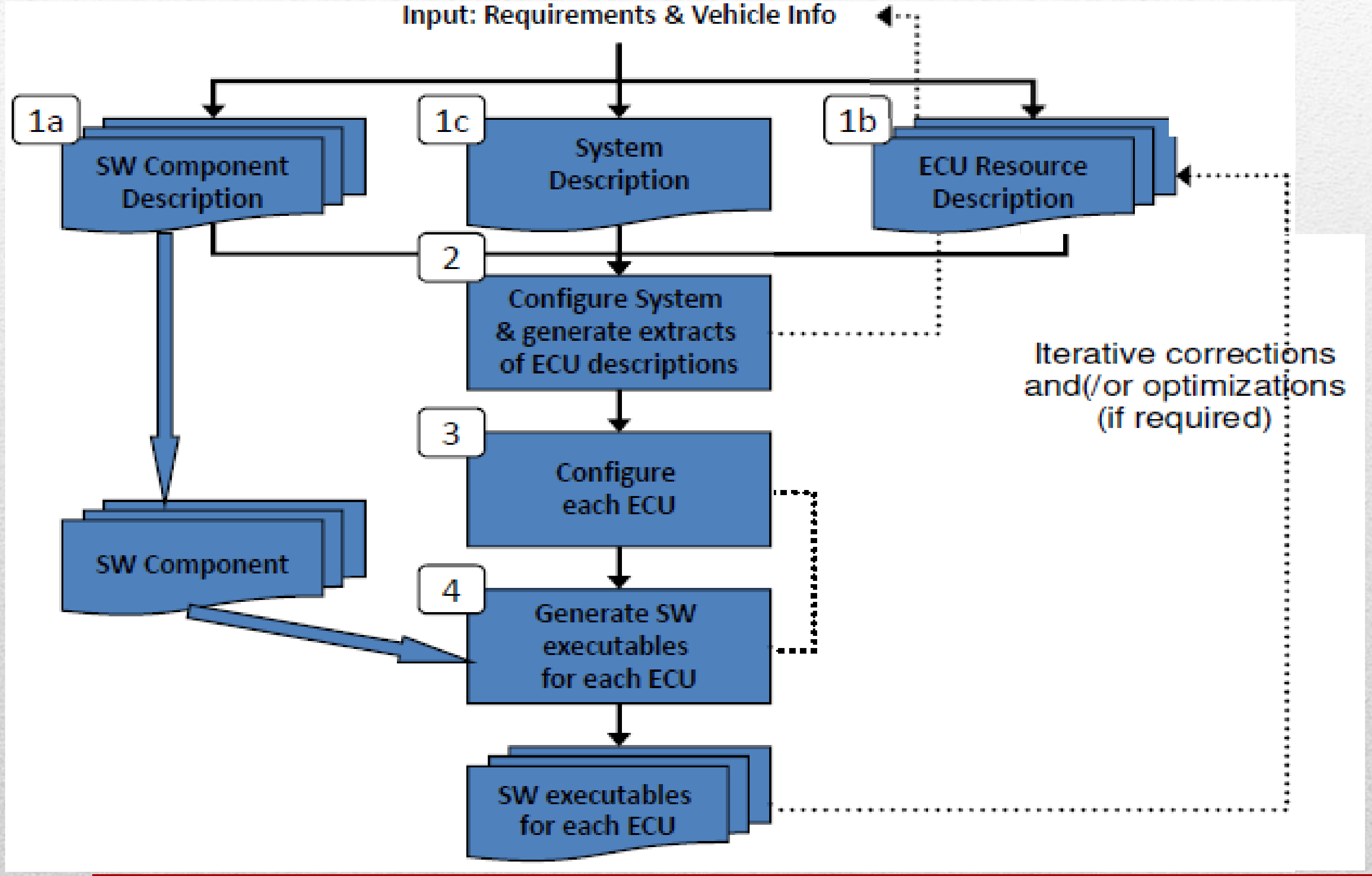


The first test

- After the first specification 31 software components were ordered from 15 vendors, these components were realized in 56 implementations
- The components were installed into two different systems. One 16 bit system and one 32 bit system
- The test led to 260 suggestions for changes in the specification
- Since then there has been few new suggestions for changes but the standard has developed and grown.

Main Concepts: Methodology

- AUTOSAR System - Design Process - Implementation Process



Main Concepts: Methodology

- AUTOSAR System - Design Process