KPIT

# AUTOSAR Handbook
# KPIT Technologies Ltd.

AUTOSAR

[ THIS PAGE INTENTIONALLY LEFT BLANK ]

# AUTOSAR Solutions

A Premium Member of AUTOSAR consortium since 2005, KPIT provides products and services for the various layers of AUTOSAR stack for OEMs, Tier1s, and Semiconductor ODMs (Original Design Manufacturers). Actively involved in the automotive standardization across the globe, we help customers at every stage of their AUTOSAR development through end-to-end AUTOSAR Tool chain in association with leading industry Tool Suppliers.

# Contents



# Part 1

# Introduction to AUTOSAR

## Automotive Industry coping up with increasing complexity

The in-vehicle systems are becoming more and more complex day in and day out. Today's hardware and component-driven development process is becoming more and more requirement and functional-driven. Future engineering does not aim at optimizing single components but optimizing on system level which requires an open architecture as well as scalable and exchangeable software modules.

AUTOSAR (AUTomotive Open System ARchitecture), a worldwide consortium of OEMs, suppliers and other companies, founded in 2003, have been working on the development and introduction of an open and standardized software architecture for the automotive industry.

To reduce development effort and improve quality are important reasons for introducing a uniform procedure independent of system platform. Hardware and software are decoupled from one another to assure such results.

The AUTOSAR concept is based on modular components with defined interfaces.

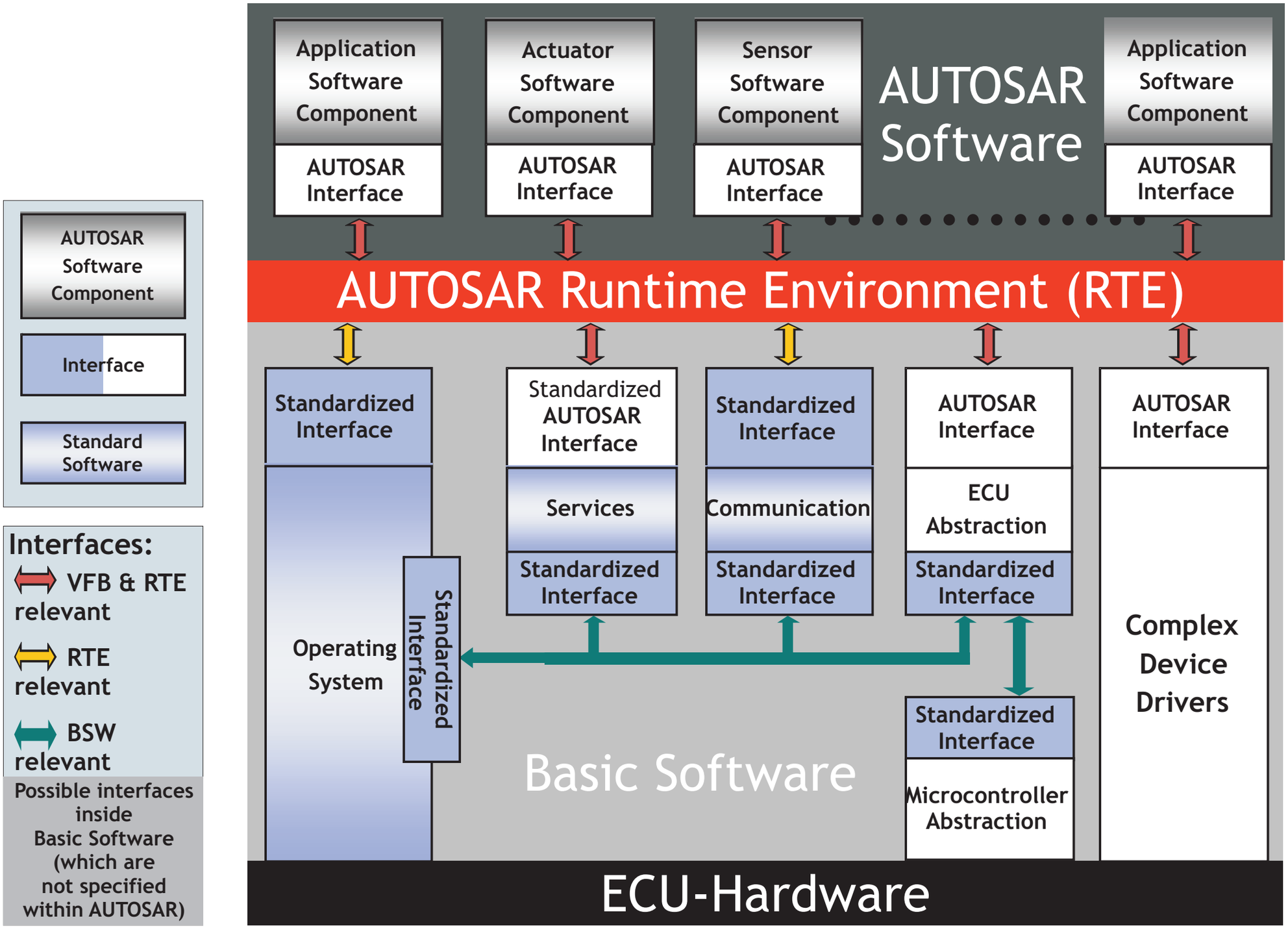



**Figure 1: Simplified Component View**

# AUTOSAR Layer Model

In AUTOSAR, the ECU software is abstracted and sub-classified as software (BSW) layer, runtime environment (RTE) and application layer.

The **Microcontroller Abstraction Layer** contains internal drivers, which are software modules with direct access to the micro controller and internal peripherals.

The **ECU Abstraction Layer** offers uniform access to all features of an ECU like communication, memory or I/O, no matter if these features are part of the microcontroller or realized by peripheral components. The drivers for such external peripheral components reside in this layer.

The **Service Layer** provides various types of background services such as vehicle network communication and management services, diagnostic services, memory management, ECU state management, mode management and Logical and temporal program flow monitoring. The operating system is also part of this layer.

The **RTE** integrates the application layer with the BSW. It implements the data exchange and controls the integration between the application software component (SWCs) and BSW.

The **Application Layer** contains the SWCs, which realize the application functionality of the ECU.

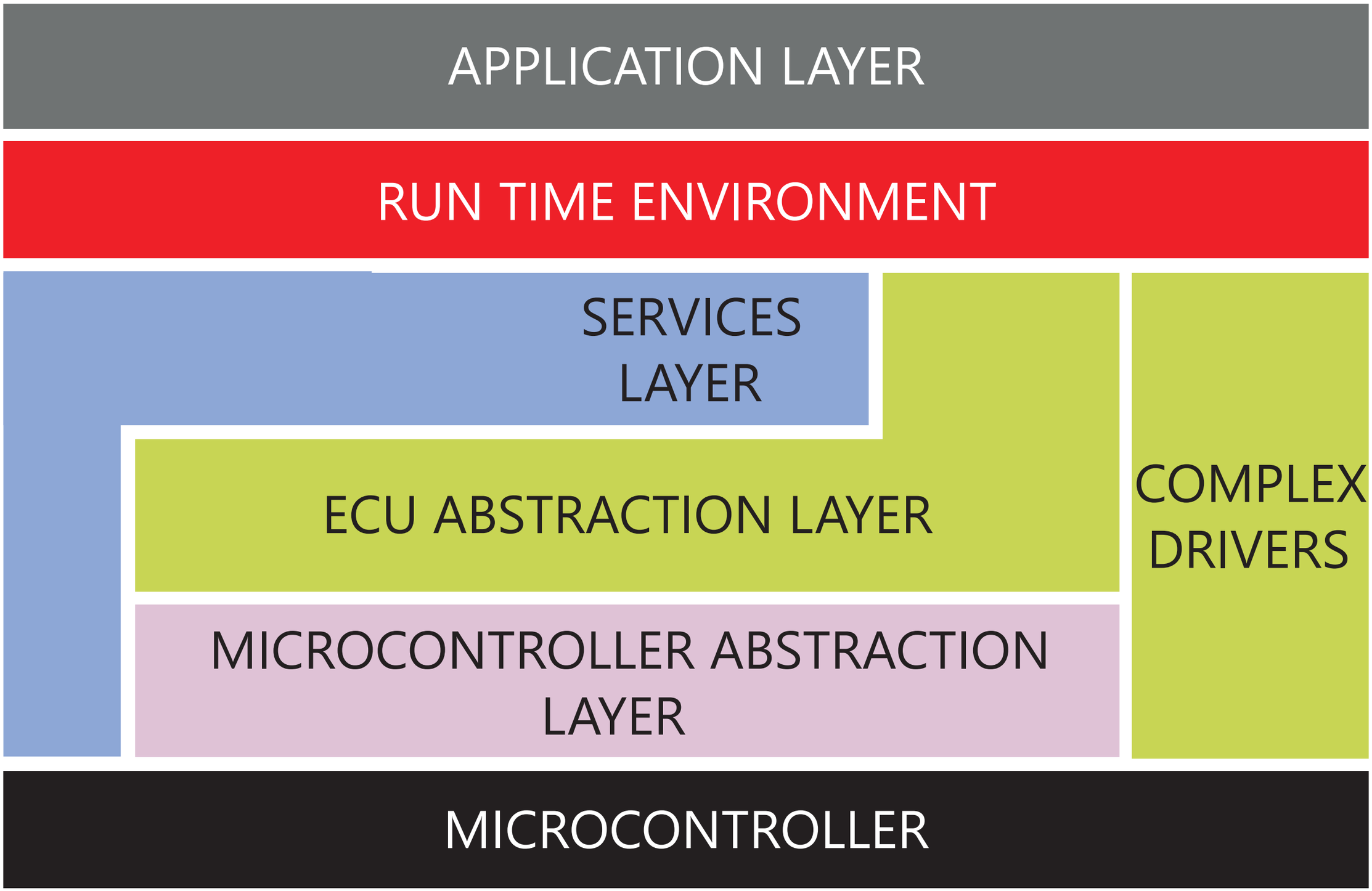



**Figure 2: AUTOSAR Layered Architecture**

# Design and Communication of software Components

A fundamental design concept of AUTOSAR is the separation between Application and Infrastructure. An application in AUTOSAR consists of interconnected "AUTOSAR Software Components". The interfaces of each SWC are formally defined. Communication between SWCs takes place chiefly over two kinds of ports, **Client/ Server ports** where server is a provider of a service and the client is a user of a service and **Sender/ Receiver ports** where a sender distributes information to one or several receivers in synchronous as well as asynchronous environment. The implementation architecture of SWC is formally defined in terms of so-called runnable entities. They correspond to procedures and are executed on a specific event such as a periodic activation or reception of new input value. During system design phase the SWCs can be integrated with their environment (e.g. hardware, driver, OS, etc) based on **Virtual Functional Bus (VFB)**. The virtual functional bus is the abstraction of the AUTOSAR Software Components interconnections of the entire vehicle.

Once the system of SWCs is deployed to the concrete vehicle network architecture, the RTE and BSW of involved ECUs realize the communication between the SWC either as ECU-local communication or as network based communication.

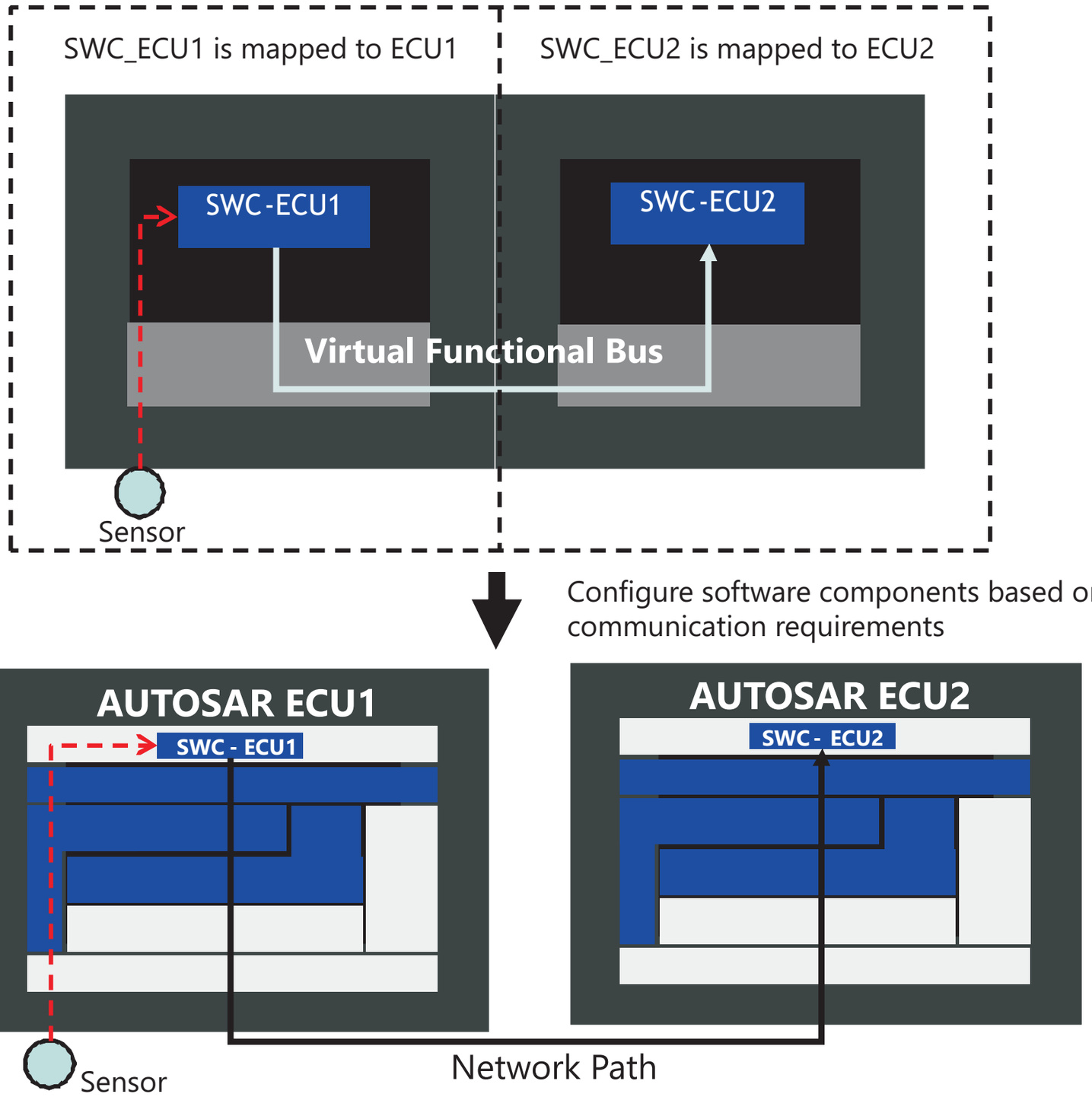



**Figure 3: AUTOSAR System Design**

# AUTOSAR Method

The AUTOSAR Methods (in AUTOSAR specifications also called “Methodology”) describes the workflow that could be followed – from the system configuration up to the final generation of an executable for a concrete ECU.

The activities are supported by dedicated AUTOSAR tools.
For exchanging work products between such tools AUTOSAR defined one comprehensive XML file format.

The detailed description of AUTOSAR work flow please refer Part 2 Description of AUTOSAR work prod & activates of this handbook.

# AUTOSAR Interfaces

AUTOSAR Interfaces are used in defining the ports of software-components and/or BSW modules. Through these ports, software-components and/or BSW modules can communicate with each other (Send or receive information or invoke services). AUTOSAR makes it possible to implement this communication between Software-Components and/or BSW modules either locally or via a network. (Refer figure 1, page 3 of handbook)

- The **AUTOSAR Interface** is a generic interface which is derived from the ports of a SWC. AUTOSAR Interfaces are provided by the RTE and serve as interface between SWCs or between SWCs or between a SWC and the ECU firmware (IO HW and Complex Drivers). Via these interfaces, a SWC an e.g. read an input value or write an output value.
- The **Standardized AUTOSAR Interface** is a particular AUTOSAR Interface, which is already predefined by the AUTOSAR standard. Such interfaces are used by the SWCs to access AUTOSAR Services, which are provided by BSW modules of the Service Layer like the ECU manager or the diagnostic event manager.
- The **Standardized Interface** is an interface, which is predefined by the AUTOSAR standard as API in C language. It is used between BSW module within an ECU, between RTE and Operating System (OS), or between RTE and the Communication Layer.

# AUTOSAR Basic Software Module

AUTOSAR has defined a set of BSW modules. They are responsible for different tasks:

- Operating System
- Access to non volatile memory
- Communication via CAN, LIN, FlexRay and Ethernet
- Handling the diagnostics
- Access to I/O ports
- System services like ECU state management

In addition, so-called Complex Device Drivers can be integrated into an AUTOSAR ECU. They are used to access the features of the ECU, which are not covered by the standard BSW of AUTOSAR.

The detailed description of the BSW module parameters is included in a module specific XML file – the BSW Module Description (compare to Figure 4 and the table in part 2 of this handbook).

A list of all BSW modules and a short description can be found in the part 3 of this handbook.

# Part 2

# Description of AUTOSAR work products & activities

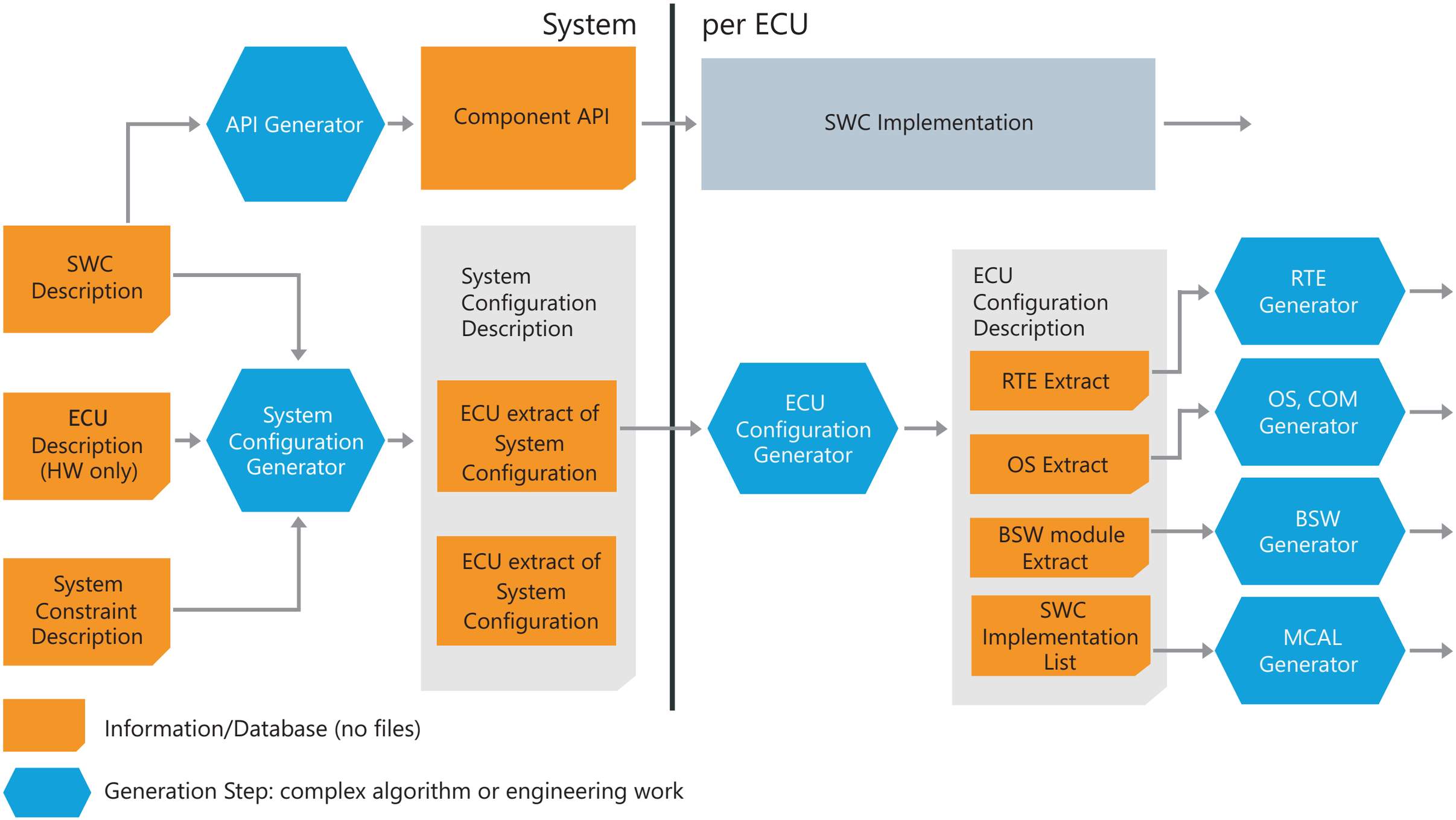



**Figure 4: Overview of the AUTOSAR method**

| *Information/ activities* | *Description* |
|---|---|
| System Constraint | These are constraints, which must be considered during system configuration. An example for such System Constraints is a given (partial) communication matrix from the last vehicle series, which must not be changed when designing the new vehicle series. |
| System Configuration Generator | This activity creates a complete System Description with the necessary SWC Description and the associated system Communication Matrix. Basis for this activity are the System Constraints. In addition, this activity requires design decisions to be made at system level by considering the available ECU & network resources. This includes the definition of network topology, definition of SWC, mapping of SWC to ECUs and specification of the network communication. |
| System Configuration Description | This includes all system information and the information that must be agreed between different ECUs including the network topology and the allocation of the components to the ECUs. A System Description is always completed by the necessary SWC Descriptions and an associated System Communication Matrix. |
| SWC Description | This specifies information about an SWC including its ports and runnable entities. |

| *Information/ activities* | *Description* |
|---|---|
| ECU Extract of System Configuration | This work product contains information from the System Configuration Description needed for a specific ECU. It includes the description of the SWCs on this ECU as well as the subset of the communication matrix relevant for the ECU. |
| ECU configuration Generator | This creates an ECU Configuration Description. Basis is the ECU Extract and the vendor specific BSW Module Description. In addition, this activity requires design decisions to be made at ECU level.<br><br>This includes setting values for the configurable parameters of all BSW modules and the RTE, like mapping runnable entities to operating system tasks, defining the memory layout or configuring the operating system. |
| ECU Configuration Description | This describes 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 |
| BSW Generator | This activity generates the configurable part of the BSW module of an ECU. Basis for this generation process is the ECU configuration Description of the ECU. This activity requires no design decisions. |

| ***Information/ activities*** | ***Description*** |
|---|---|
| BSW extract of ECU configuration | This describes of all configuration parameters of a particular BSW module, including vendor specific parameters. This description always reflects a concrete implementation of the BSW module. Therefore, it is provided by the BSW module supplier and is not changed during the ECU configuration process. |
| RTE Generator | This activity generates the RTE of an ECU. This activity requires no design decisions. |

[ THIS PAGE INTENTIONALLY LEFT BLANK ]

# Part 3

# Description of AUTOSAR Basic Software Module (BSW)

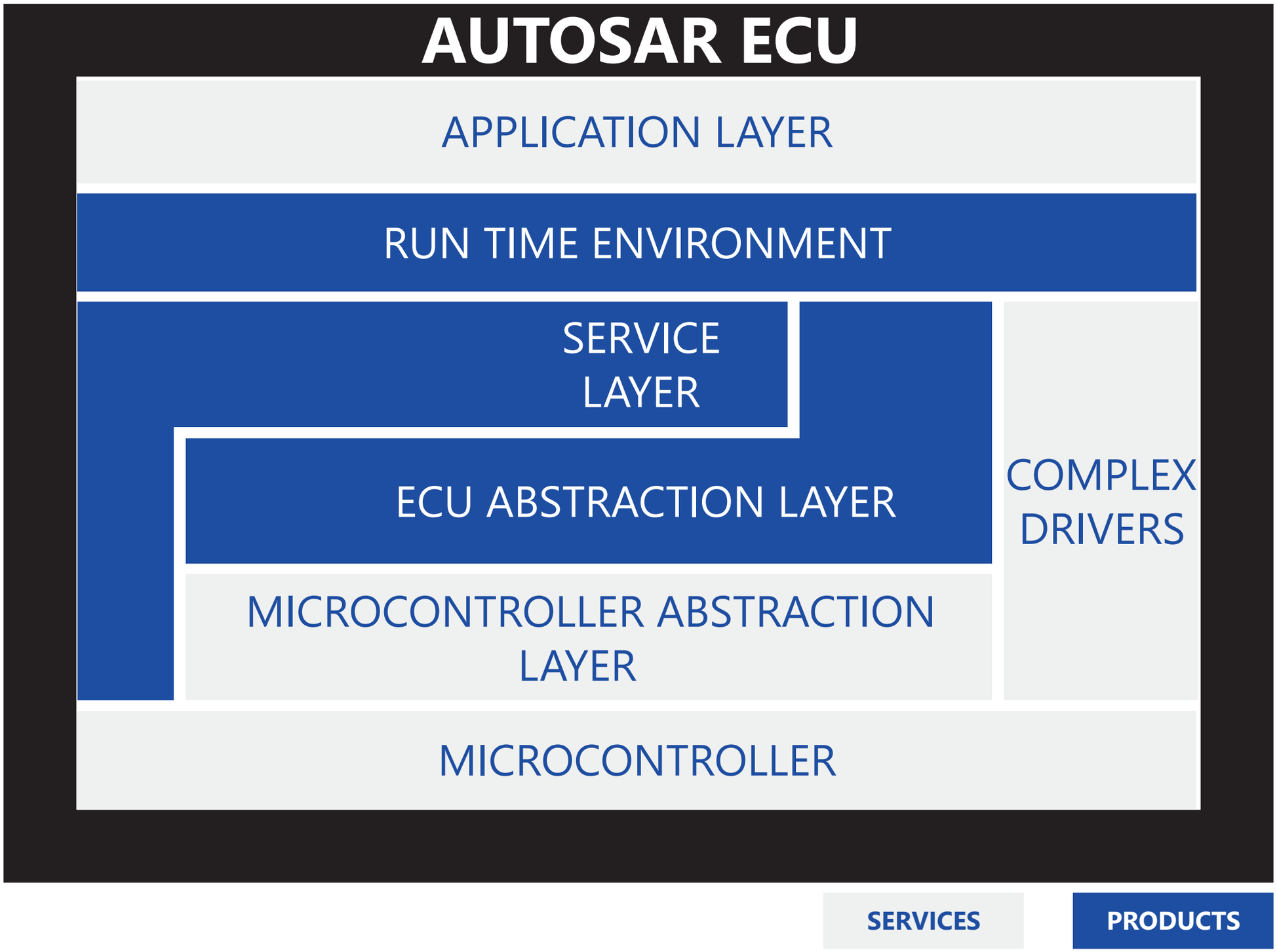



**Figure 5: AUTOSAR Stack and KPIT Offerings**

APPLICATION LAYER

RUN TIME ENVIRONMENT

ECU STATE MGR
COM MANAGER
FUNCTION INHIBITION MGR
DIAGNOSTIC EVENT MGR
WATCHDOG MGR
DEVELOPMENT ERROR TRACER
SYNC TIME BASE MGR
AUTOSAR OS
DIAGNOSTIC LOG AND TRACE
BASIC SOFTWARE MODULE MGR

NVRAM MANAGER

IPDU MULTIPLEXER
AUTOSAR COM
DCM
PDU ROUTER
CAN/ FR TRANSPORT
CAN/ LIN/FR STATE MANAGER
GEN. NM INTERFACE
CAN/ LIN/ FR NM

I/O SIGNAL INTERFACE
DRIVER FOR EXT. ADC ASIC
DRIVER FOR EXT. I/O ASIC

COMPLEX DRIVERS

WATCHDOG INTERFACE

MEMORY ABS INTERFACE
EEPROM ABSTRACTION
FLASH EEPROM EMULATION
EXTERNAL EEPROM DRIVER
EXT FLASH DRIVER

CAN/LIN/FLEXRAY/ETHERNET INTERFACE
CAN/FR/ETH TRANS-CEIVER DRIVER
DRIVER FOR EXTERNAL CAN ASIC

MCU DRIVERS
MEMORY DRIVERS
COM DRIVERS
I/O DRIVERS

MICROCONTROLLER

**Figure 6: BSW Modules (Not all modules are shown)**

| Module Name | Description |
|---|---|
| FlexRay Interface | The FlexRay interface provides identical access mechanisms for the ECUs FlexRay channels, independent of their implementation (microcontroller internal or external). It extracts the number of FlexRay drivers and manages the synchronization to global FlexRay time. |
| FlexRay Network Management | This module is responsible for the FlexRay network management. It coordinates the transition between normal operation and bus-sleep mode of the network. |
| FlexRay State Manager | The FlexRay State Manager controls and monitors the wakeup and startup of the node in the FlexRay cluster. |
| FlexRay Transport Layer | The FlexRay transport protocol segments long data packets in the transmit direction, collects data in the receive direction and controls the data flow. Errors such as message loss, message duplication or sequencing errors are detected. |
| FlexRay Transceiver Driver | The driver for an external FlexRay transceiver is responsible for Network diagnostics and switching a transceiver on and off. |

| Module Name | Description |
|---|---|
| FlexRay Interface | The FlexRay interface provides identical access mechanisms for the ECUs FlexRay channels, independent of their implementation (microcontroller internal or external). It extracts the number of FlexRay drivers and manages the synchronization to global FlexRay time. |
| FlexRay Network Management | This module is responsible for the FlexRay network management. It coordinates the transition between normal operation and bus-sleep mode of the network. |
| FlexRay State Manager | The FlexRay State Manager controls and monitors the wakeup and startup of the node in the FlexRay cluster. |
| FlexRay Transport Layer | The FlexRay transport protocol segments long data packets in the transmit direction, collects data in the receive direction and controls the data flow. Errors such as message loss, message duplication or sequencing errors are detected. |
| FlexRay Transceiver Driver | The driver for an external FlexRay transceiver is responsible for Network diagnostics and switching a transceiver on and off. |

| Module Name | Description |
|---|---|
| LIN Interface | The LIN Interface provides a hardware independent interface for access to LIN frames. In addition, it manages Schedule Table processing and the implementation of the LIN Transport Layer and LIN Network Management. |
| LIN Network Management | This module is responsible for the LIN network management. It coordinates the transition between normal operation and bus-sleep mode of the network. |
| LIN state Manager | The LIN State Manager switches the Scheduler Tables as well as the PDU groups in COM and servers the LIN Interface in term of sleep and wake-up. In addition it manages the activation of the LIN Transceiver driver. |
| LIN Transport Layer | The LIN transport protocol segment data in the transmit direction, collects data in the receive direction and controls the data flow. Errors such as message loss, message duplication or sequencing errors are detected. The LINTP is part of the LINIF. |
| LIN Transceiver Driver | The LINTRCV driver for an external LIN transceiver is responsible for monitoring and controlling the wake-up and sleep functions. |

| Module Name | Description |
|---|---|
| Ethernet Interface | This module provides to upper layers a hardware independent interface to the Ethernet Communication System comprising multiple different Ethernet controllers and transceivers. This interface is uniform for all Ethernet controllers and transceivers. Thus, the upper layers (Internet Protocol, Address Resolution Protocol) may access the underlying bus system in a uniform manner. |
| Ethernet Transceiver Driver | The module provides to the upper layer (Ethernet Interface) a hardware independent interface comprising multiple equal transceivers. This interface is uniform for all transceivers. Thus, the upper layer (Ethernet Interface) may access the underlying bus system in a uniform manner. The configuration of the Ethernet Transceiver Driver however is bus specific, since it takes into account the specific features of the communication transceiver. |
| Ethernet State Manager | The Ethernet State Manager shall provide an abstract interface to the AUTOSAR Communication Manager to startup or shutdown the communication on an Ethernet cluster. It does not directly access the Ethernet hardware |

| Module Name | Description |
|---|---|
| Generic Network Management Interface | The NM module provides a general, network independent interface for access to bus-dependent network management modules (CANNM and FRNM). In addition, the module manages the synchronous, cross-network shutdown of the communication system in conjunction with the other ECUs. |
| Communication | The communication layer provides a signal-based data interface for the application, and sends messages according to the defined send types. Additional interfaces are provided in the form of messaging mechanisms for successful sending and receiving of data as well as their timeouts. For multi-channel ECUs, the module COM also provides an option to route signals between communication buses (signal gateway) |
| Diagnostic Communication Manager | This module implements diagnostic communication as per ISO14229-1 (UDS). Diagnostic requests are partially converted directly in DCM (administration of diagnostic sessions, reading error codes, EcuReset,...), and partially sent to software components via port interfaces (reading, writing, and controlling of data identifiers, execution of routines, ...). |

| Module Name | Description |
|---|---|
| IPDU Multiplexer | This module deals with multiple uses of fixed PDUs with different data contents. |
| EEPROM Abstraction | The EA module provides a hardware independent interface for access to EEPROM driver (EEP). Data blocks can be read, written, or deleted. In addition, the EA module distributes write request across different areas of the EEPROM so that all EEPROM cells are subjected to equal load, and their lifespan is increased. |
| Flash EEPROM Emulation | The FEE module provides a hardware independent interface for access to flash data using a flash driver (FLS). Data blocks can be read, written, or deleted. In addition, the FEE module distributes write requests across different areas of the flash so that all flash cells are to equal load, and their lifespan is increased. |
| Memory Abstraction Interface | This module allows the NVRAM manager to access several memory abstraction modules (FEE or EA modules). This module abstract from the number of underlying FEE or EA modules and provide upper layers with a virtual segmentation on a uniform linear address space. |

| Module Name | Description |
|---|---|
| External Driver | On request, We offer the implementation of drivers for externally connected components as an extension to AUTOSAR 3.0. These are already available for the control of certain EEPROM (EEPEXT), flash components (FLSEXT), Watchdog (WDGEXT), ... for example |
| Watchdog Interface | This module provides uniform access to services of the watchdog driver (WDG), such as mode switching and triggering. |
| Watchdog Manager | The Watchdog Manager monitors the reliability and functional assurance of the application in an ECU. This includes monitoring the correct execution of the SWCs and BSW modules, and the triggering of Watchdog at the required time intervals. It reacts to possible faulty behavior on numerous escalation levels. Where resumption of normal operation is impossible, the Watchdog hardware performs a reset of the microcontroller. |
| I/O Hardware Abstraction | The I/O Hardware Abstraction represents the connection between the RTE and the I/O channels of the ECU. It encapsulates access to the I/O drivers such as ADC, DIO or PWM, thereby making available the ECU's I/O signals. |

# Part 4

# AUTOSAR System Services

**This module is the operating system of an AUTOSAR ECU. It is actually an extended OSEK operating System. The extensions are organized so-called scalability classes (SC1-SC4). They cover the following features.**

- **SC1 :** deterministic RTOS baseline (tasks, events, counters, alarms, messages)
- **SC2 :** timing based task determinism (low-latency, precise timing for periodic tasks)
- **SC3 :** protected memory (MMU/MPU) for tasks avoids memory collisions for safety
- **SC4 :** timing and memory protected tasks, utilizes the full capabilities of the silicon for secure & protected Automotive grade RTOS

| | SC1 | SC2 | SC3 | SC4 | Hardware Requirements |
|---|---|---|---|---|---|
| OSEK OS (All Conformance Classes) | ✓ | ✓ | ✓ | ✓ | |
| Counter Interface | ✓ | ✓ | ✓ | ✓ | |
| Schedule Tables | ✓ | ✓ | ✓ | ✓ | |
| Stack Monitoring | ✓ | ✓ | ✓ | ✓ | |
| ProtectionHook | | ✓ | ✓ | ✓ | |
| Timing Protection | | ✓ | | ✓ | Timers with high priority interrupt |
| Global Time/Synchronization Support | | ✓ | | ✓ | Global Time Source |
| Memory Protection | | | ✓ | ✓ | MPU |
| OS-Application | | | ✓ | ✓ | |
| Service Protection | | | ✓ | ✓ | |
| CallTrustedFuction | | | ✓ | ✓ | (non-) privilege Modes |

**Figure 7: Scalability Class Details**

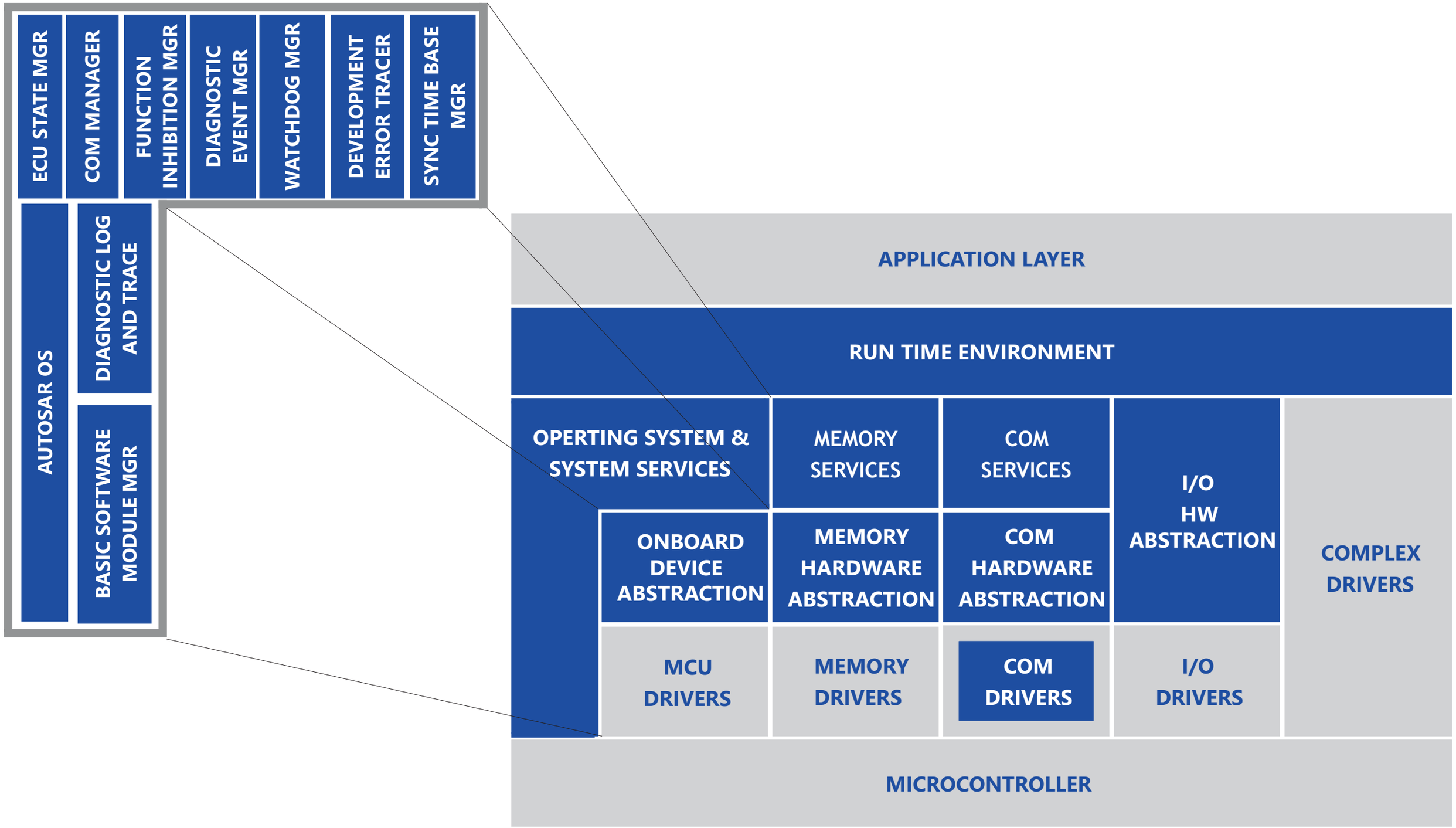



**Figure 8: System Services**

| *Module Name* | *Description* |
|---|---|
| ECU State Manager | The ECU State Manager performs the initialization/de-initialization of all basic software modules, including the RTE and the operating system (OS). The module controls the operating state of an ECU (Sleep, Startup, Wakeup, Shutdown, and Run) based on system events. |
| Communication Manager | This module controls the state of all communication channels connected to the ECU and provides a bus-independent interface to the SWCs (and thereby their application) for requesting external communication. |
| Function Inhibition Manager | This module controls (enable/disable) functionalities of SW components based on the conditions such as faults, signal quality, ECU and vehicle states, diagnostic tester commands, etc. |
| Diagnostic Event Manager | This module implements error memory as per manufacturer-specific documentation. A standardized interface for diagnostic monitors allow for uniform, cross-manufacturer development of software components. The DEMs module is responsible for administrating the Diagnostic TroubleCode statuses, the error environment data, and for storing the data in NVRAM. |

| Module Name | Description |
|---|---|
| Development Error Tracer | This module supports error searches during software development and provides an interface for error reporting. This interface is called from the individual BSW modules in an error situation. |
| BSW Scheduler | The SCHM module calls the cyclic function for the individual BSW modules and makes available the functions that the BSW modules need to call at the beginning and end of critical sections. This Module is part of RTE (Runtime Environment in R4.0) |
| CRC Routines | The Cyclic Redundancy Check module provides a service function for computing CRC checksum. |
| Runtime Environment | The RTE is responsible for the execution of the software components and realize the data exchange between the software components and the basic software. |

[ THIS PAGE INTENTIONALLY LEFT BLANK ]

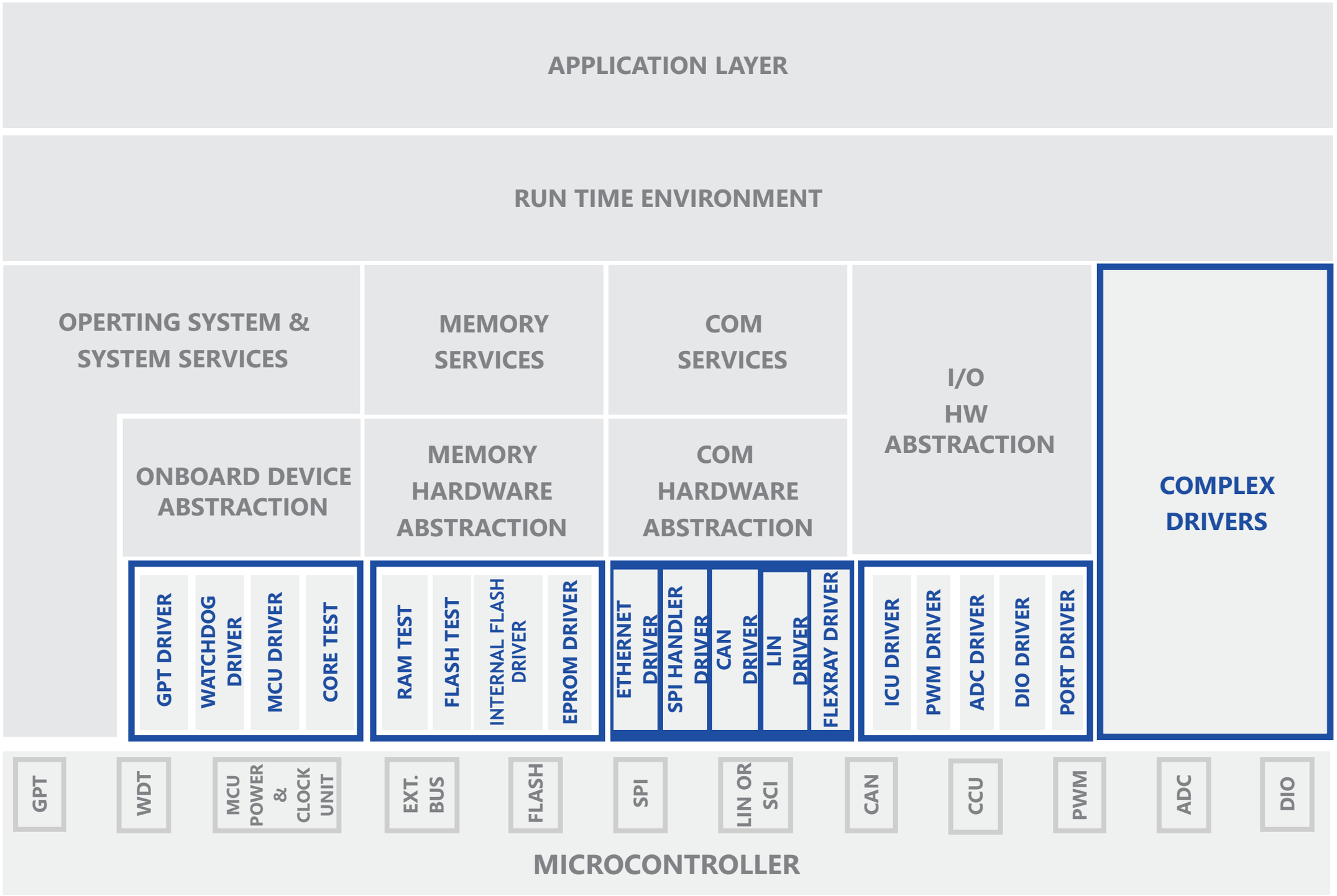



**Figure 9: K-SAR Editor MCAL Modules**

| Module Name | Description |
|---|---|
| Port Driver | This module provides service for initialize the entire port structure of the microcontroller. |
| DIO Driver | The Digital Input Output Driver provides read and write services to the DIO channels (pins), DIO port and DIO channel groups. |
| ADC Driver | The ADC Driver is responsible for controlling the analog to digital converter and for accessing the results of a conversion. In detail, it initializes the converter, provides services for starting or ending a conversion, and for selecting the trigger source and for selecting the trigger source and trigger condition. |
| PWM Driver | The Pulse Width Modulation Driver provides services for initialization and controlling PWM channels of the microcontroller |
| ICU Driver | The ICU driver provides services for edge detection, measuring periodic signals, assigning edge timestamp and controlling wake-up interrupts. |

| Module Name | Description |
|---|---|
| CAN Driver | The CAN driver provides services for initializing the CAN controller, for sending and receiving messages, and for switching the controller states (sleep, stop etc.). |
| FlexRay Driver | The FlexRay Driver is used to abstract hardware related differences between different FlexRay communication controllers. All the necessary properties of the communication controller per the FlexRay Protocol Specification are encapsulated in this module and can be reached via its uniform interface. |
| LIN Driver | The LIN Driver provides services for initiating frame transmission (Header, Response, Sleep Mode and Wake-Up) as well as receiving responses, checking the momentary state and validating wake-up events. |
| SPI Handler / Driver | The SPI driver provides an option for exchanging data across the SPI interface. It is primarily used for external connection of EEPROM and Watchdog, ... |
| Ethernet Driver | The Ethernet driver provides an option for data exchanges over Ethernet interface. With the Ethernet interfacing available in, it is possible to develop very powerful gateways with a MOST connection and thereby utilize the advantages of the AUTOSAR architecture. |

| Module Name | Description |
|---|---|
| EEPROM Driver | The EEPROM driver enables hardware independent, uniform access to EEPROM storage. It makes available services for reading, writing, and data comparison, as well as for deleting blocks. |
| Internal Flash Driver | The flash driver provides a hardware independent and uniform access to flash memory. It offers services for reading, writing and comparing of data and the erasure of blocks (sector). |
| RAM Test | This module tests microcontroller internal RAM cells. A complete test is performed during startup and shutdown of the ECU, or is trigged by a diagnostic command. A cyclical test (block for block or cell for cell) is performed during normal operation. |

| Module Name | Description |
|---|---|
| Watchdog Driver | This module provides services to control and trigger watchdog hardware. The trigger routine is called by the watchdog manager. |
| GPT Driver | The General Purpose Timer Driver provides an interface for access to the microcontroller's internal timers. It can be used to control events that occur periodically or once-off. |
| MCU Driver | The Micro Controller Unit Driver provides the following services:<br>■ Software-triggered microcontroller reset.<br>■ Selection of the microcontroller power mode (STOP, SLEEP, HALT, etc.)<br>■ Configuration of wake-up behavior.<br>■ Management of the internal PLL clock unit.<br>Initialization of RAM areas with pre-defined values. |
| Core Test | The Core Test Driver provides services for configuring, starting, polling, terminating and notifying the application about Core Test results. It also provides services for returning test results in a predefined way. Furthermore it provides several tests to verify dedicated core functionality like e.g. general purpose registers or Arithmetical and Logical Unit (ALU). |

| Module Name | Description |
| --- | --- |
| Complex Drivers | The Complex Drivers contain drivers which are not standardized in AUTOSAR and which utilize specific properties of a microcontroller or ECU (e.g. complex peripheral devices). They include functionalities for sensor evaluation and controller monitoring with direct access to the microcontroller. |

[ THIS PAGE INTENTIONALLY LEFT BLANK ]

# Part 6

# Multicore Support

# Introduction:

As the demand for computing power is rapidly increasing in the automotive domain, OEMs and Tier-one suppliers are gradually introducing multicore ECUs in their electronic architectures. Additionally, these multicore ECUs offer new features such as higher levels of parallelism which ease the respect of the safety requirements such as the ISO26262 and the implementation of other more complex automotive use-cases. Main use cases for multicore ECUs can be;

1. Decreasing complexity of architecture
2. Dealing with resource demanding applications
3. Improving the safety
4. Dedicated use of cores

Keeping all this in mind, AUTOSAR version 4.0 has introduced support for multi-core embedded real-time operating systems. New concepts such as locatable entities (LEs), multi-core startup/shutdown, Inter-OS-Application Communicator (IOC), and Spinlock have been introduced in the AUTOSAR multi-core OS architecture specification to extend the single-core OS specifications.

The Inter-OS-Application Communicator (IOC) which is part of AUTOSAR OS, provides communication services which can be accessed by clients which need to communicate across OS-Application boundaries on the same ECU. Every Core runs a kind of ECU state management. Each core will also have 'Core Test' module running in BSW.

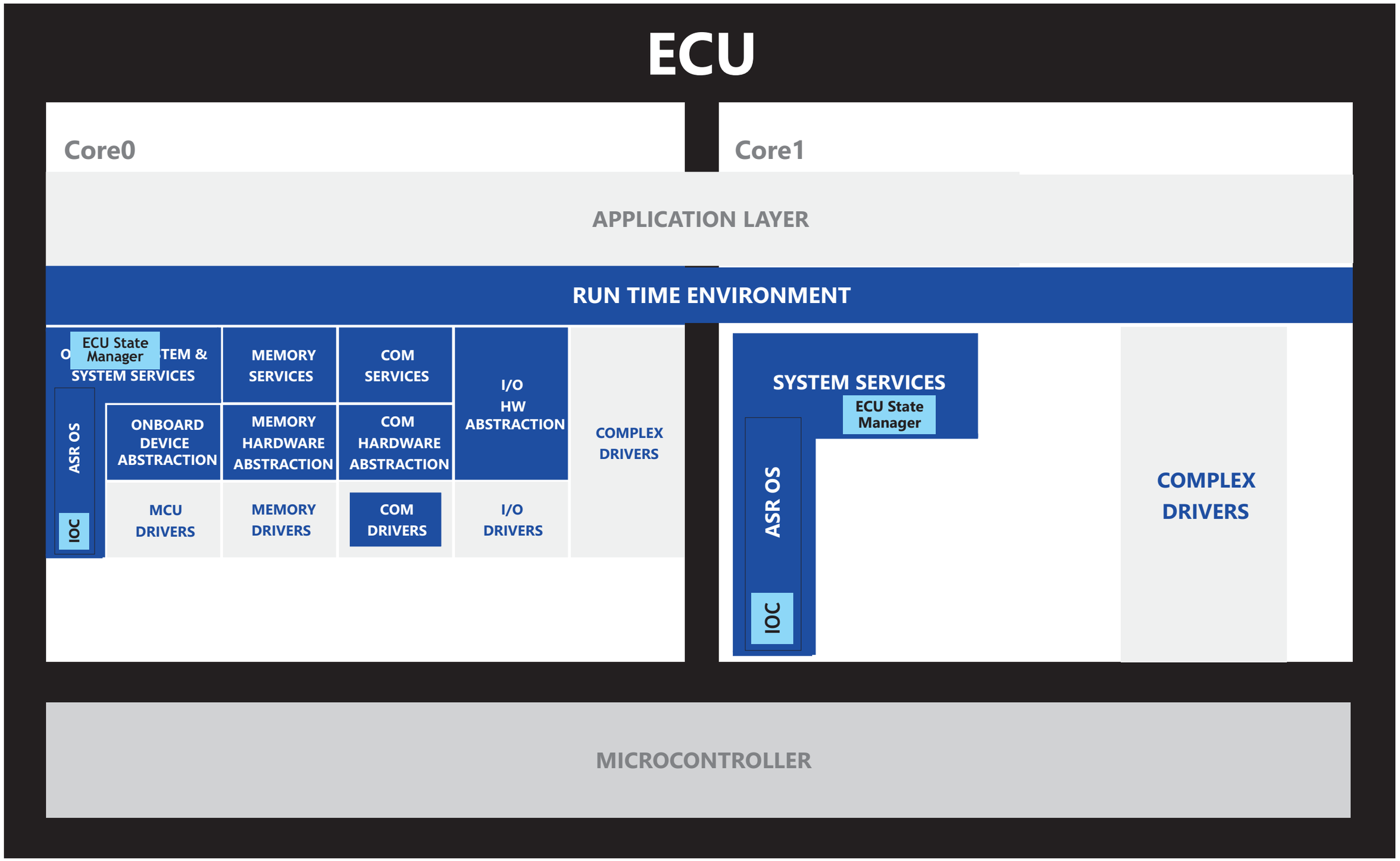



**Figure 10: An ECU with two core microcontroller**

[ THIS PAGE INTENTIONALLY LEFT BLANK ]

# Part 7

# Functional Safety

# Introduction:

Functional Safety is part of the overall safety of a system that depends on the correct execution of specific functions. The goal of functional safety is to perform the intended function correctly or the system will fail in a predictable safe manner. Functional safety standard ISO26262 which is derived from IEC-61508 mandates us to have automotive specific risk based approach for Electrical and Electronic (E/E) systems. It is applicable to passenger cars with max gross weight up to 3.5 tons.

Aspects such as complexity of the system design can be relevant for achievement of functional safety in automotive field. Software is one parameter that can influence complexity on system level. New techniques and concepts for software development can be used in order to minimize complexity and therefore can ease the achievement of functional safety.

As a software standardization initiative, AUTOSAR R4.0 considers aspects of functional safety relevant for today's automotive software development.

# AUTOSAR Architecture & Safety Implementation in R4.0:

- Memory Protection feature (MPU) in OS – “SC4”
- Multi core OS features
- E-Gas monitoring related features
- Program flow monitoring related features
- Timing related features includes;
  - Features related to the provision of synchronized time bases
    - Provision of a synchronized time-base within a cluster
    - Services for accessing to synchronized time-bases
    - Sync AUTOSAR OS with FlexRay Global Time in a well-defined way
  - Features related to synchronization of processing of asynchronous processing units
    - Services for synchronization of SW-Cs
  - Features to allow time deterministic implementation of applications
  - Features related to protection against timing violation
- Program flow monitoring related features
- Communication Stack Related Features such as Data sequence control and multiple communication links
- SWC End-to-End (E2E) communication protection
- Memory partitioning and user/supervisor-modes Related Features

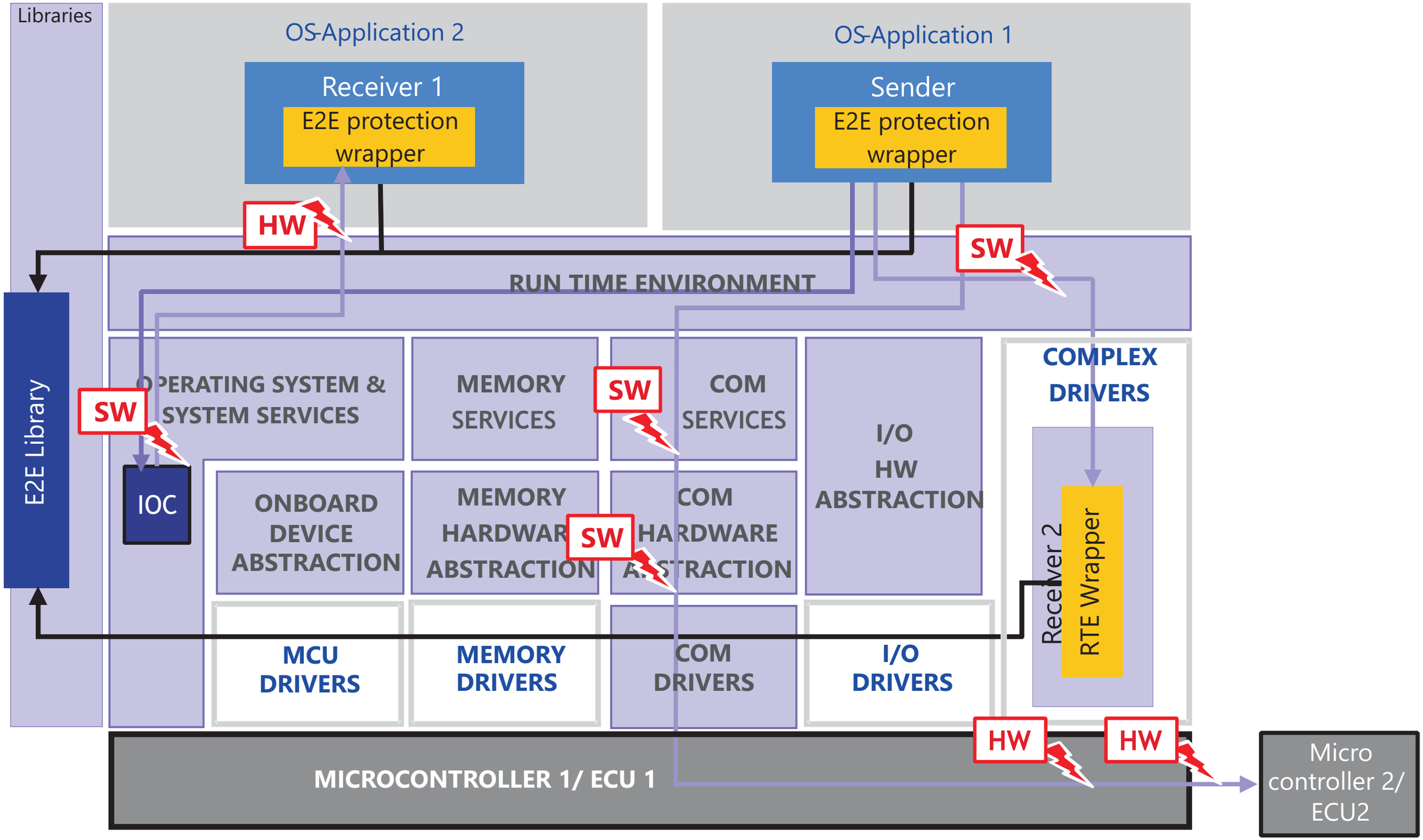



**Figure 11: Safety End to End (E2E) Communication Protection Module**

Figure 11 describes typical sources of interferences, causing errors detected by E2E protection:

**SW-related sources:**

- Error in mostly generated RTE,
- Error in partially generated and partially hand-coded COM
- Error in network stack
- Error in generated IOC or OS

**HW-related sources:**

- Microcontroller error during core/partition switch
- Failure of HW network
- Network EMI
- Microcontroller failure during context switch (partition) or on the communication between cores

E2E Lib extends signals with CRC- and Sequence Counter-information on the sender side and checks the information on the receiver side, ensuring efficient 'communication failure' detection between SWCs.

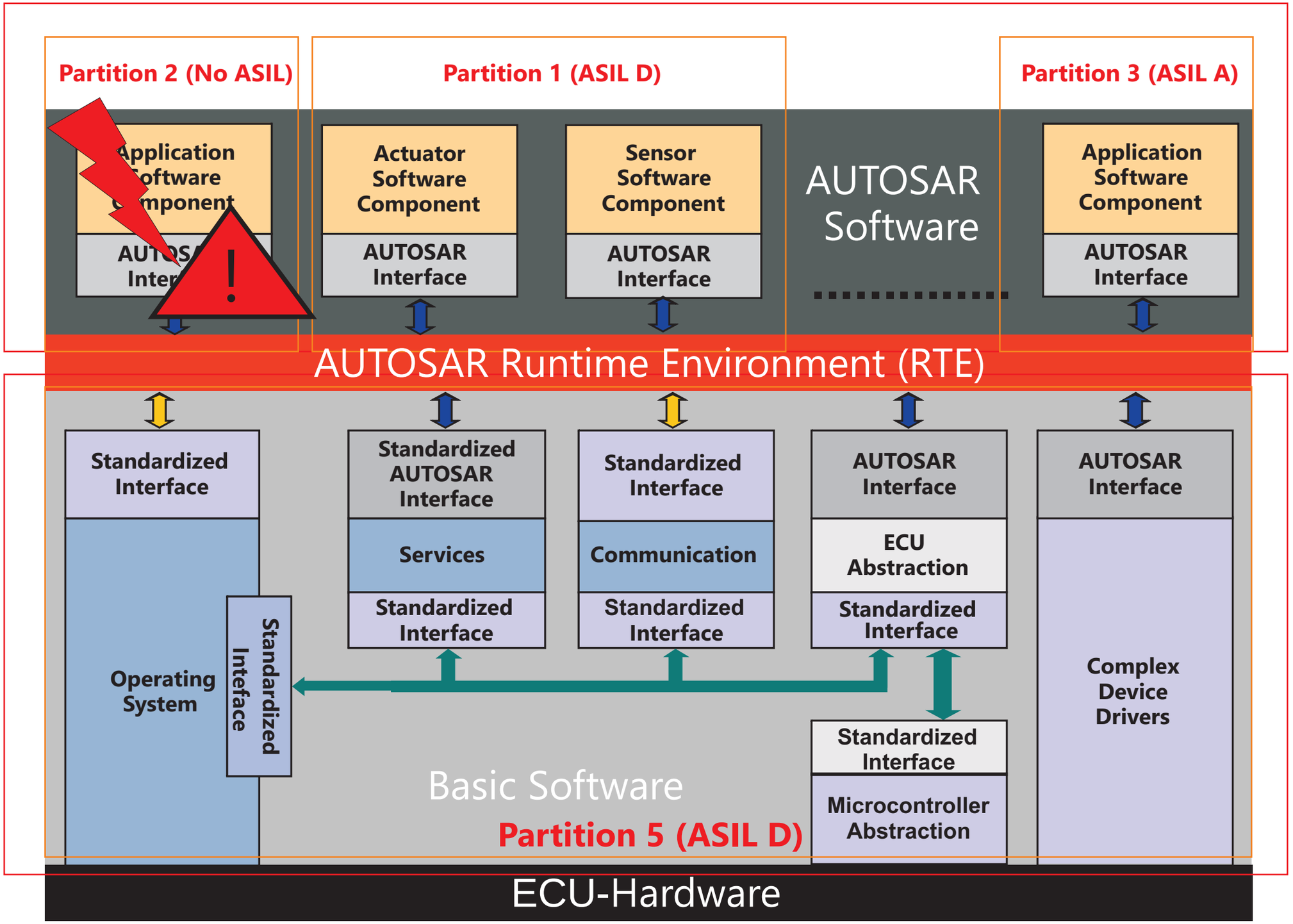



**Figure 12: Partitioning concept in functional safety**

Various software components (SW-Cs) are implemented with different Automotive Safety Integrity Levels (ASILs) in application layer. ASIL A, ASIL B, ASIL C and ASIL D are different safety integrity levels which are followed during software development depending on complexity and criticality of modules/ components ('A' being least critical, 'D' is most critical).

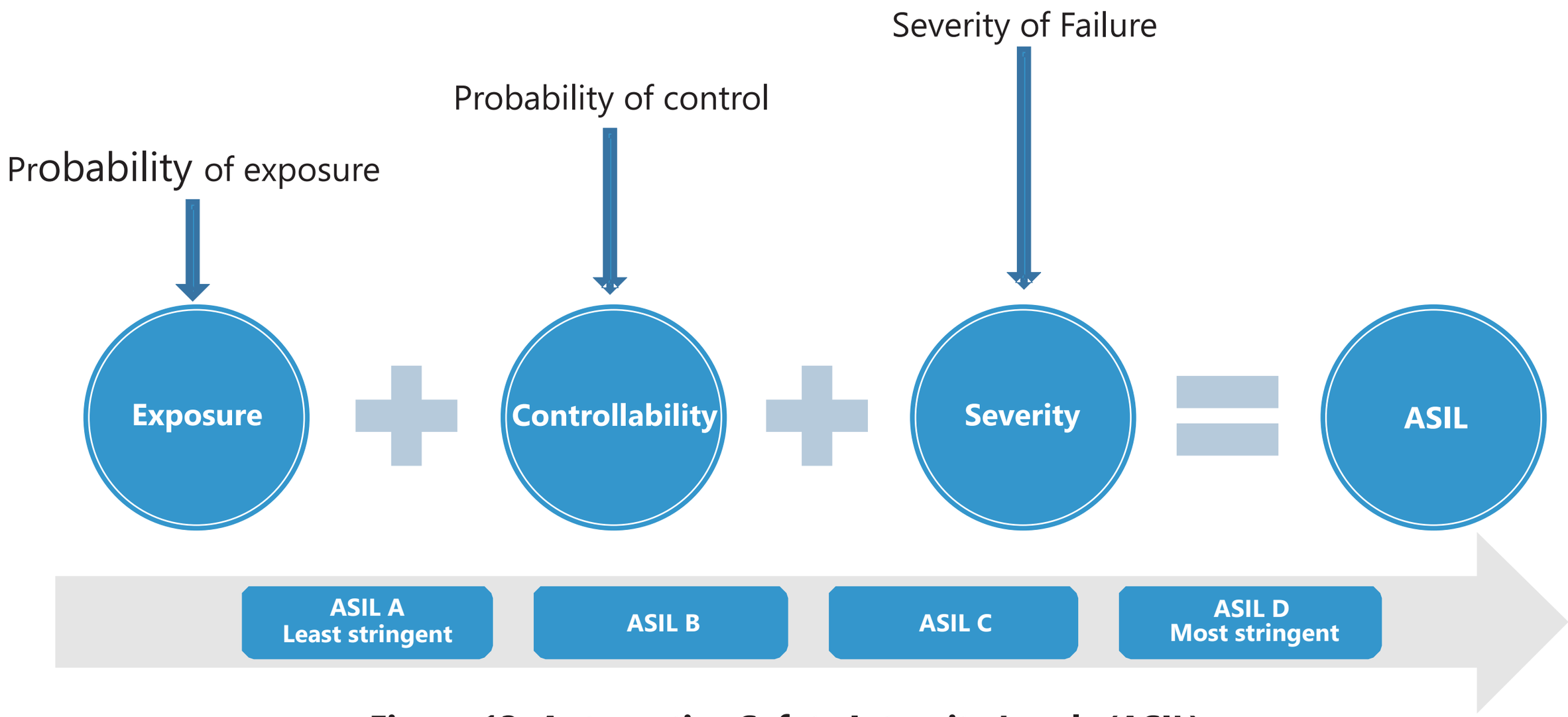



**Figure 13: Automotive Safety Integrity Levels (ASIL)**

With partitioning concept in AUTOSAR R4.0, SWCs can be placed into separate partitions of ECU. These partitions can be terminated, monitored and restarted independently. The sole purpose of these separate partitions is to achieve “Freedom from Interference”. With this SWCs with different ASILs (according to ISO26262) can be executed on same ECU.

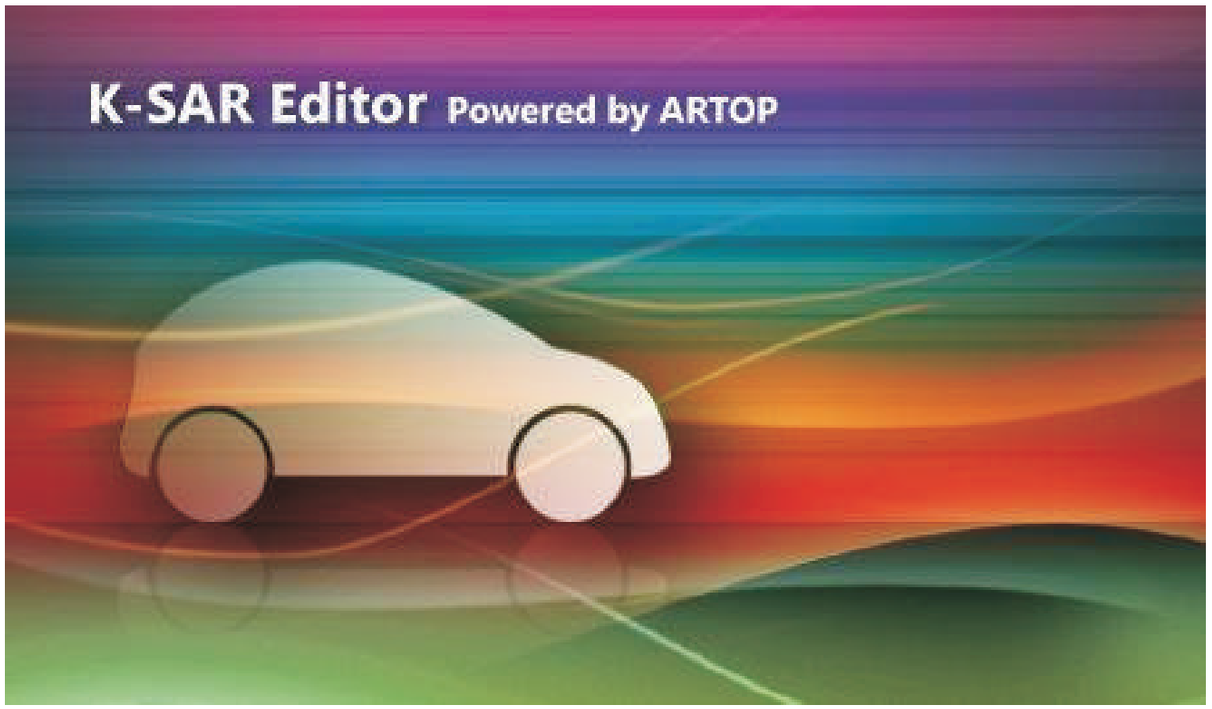
# Part 8

# K-SAR Editor Toolchain

## Introduction:

KPIT's K-SAR Editor toolchain dynamically generates GUI controls for AUTOSAR Modules specified in ECU Configuration Parameter Definition File and also generates ECU Configuration Description File.

K-SAR Editor toolchain supports Plug-in option for Generation Tools including third party Generation Tools. Using this feature, 'C' Header and Source files can be generated directly by invoking the Generation Tool from the K-SAR Editor.



## Inputs Used:

**ECU Configuration Parameter Definition File(s):** in XML format and contains definition for Modules, Containers and Parameters. The format of the XML file must compliant to AUTOSAR ECU specification standards.

**Custom Configuration CSV file:** as input at the time of loading the System Configuration Description file. This CSV file is in text format and contains the tier-1 specific notifications and ECU Configuration gets updated automatically based on the notifications.

## Outputs Generated:

The output of K-SAR Editor software is ECU Configuration Description File in XML format, which contains the configured values for Parameters, Containers and Modules. ECU Configuration Description File format must compliant to AUTOSAR ECU specification standards.

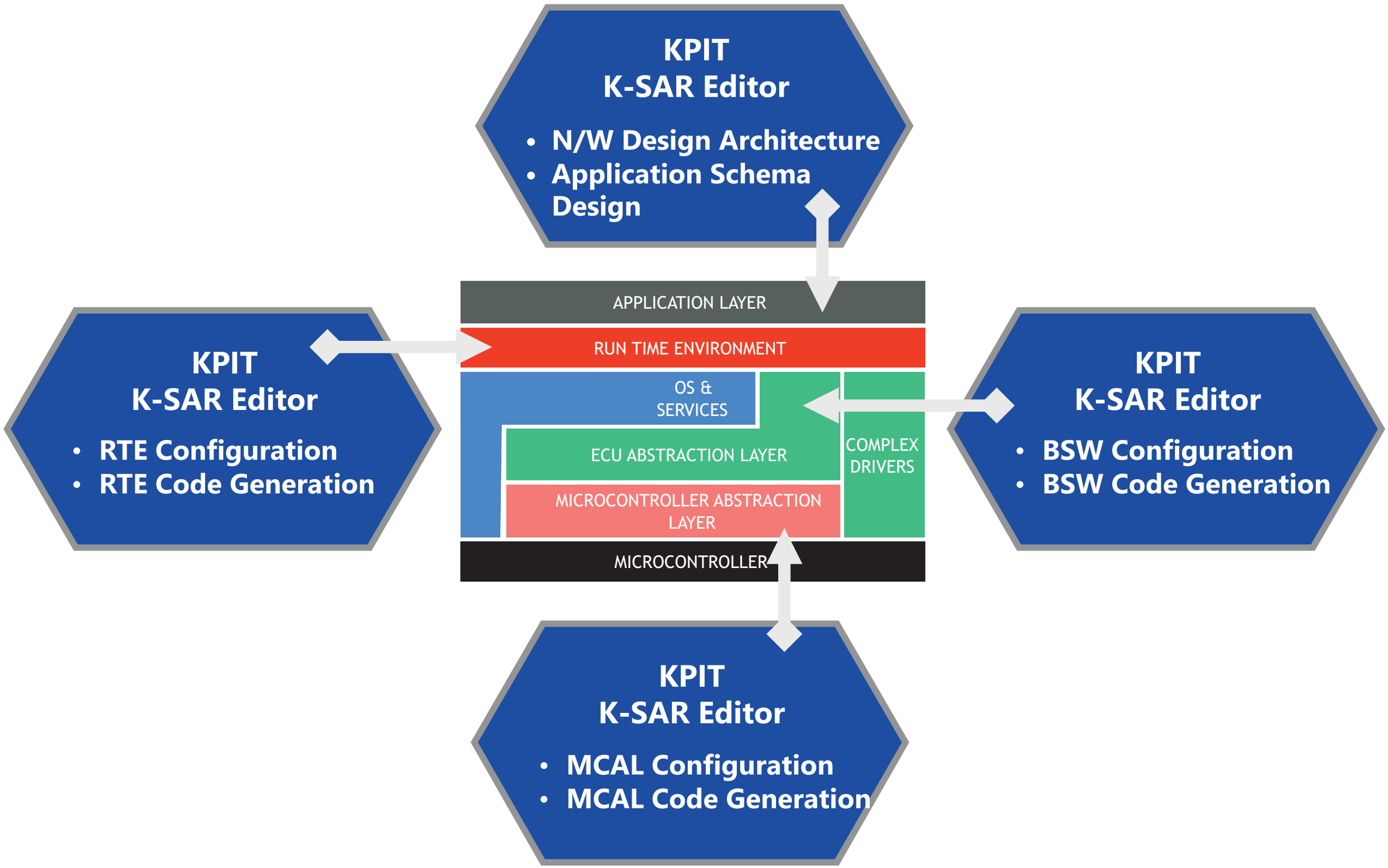



**Figure 14: KPIT's K-SAR Editor Toolchain for AUTOSAR Layered Model Development**

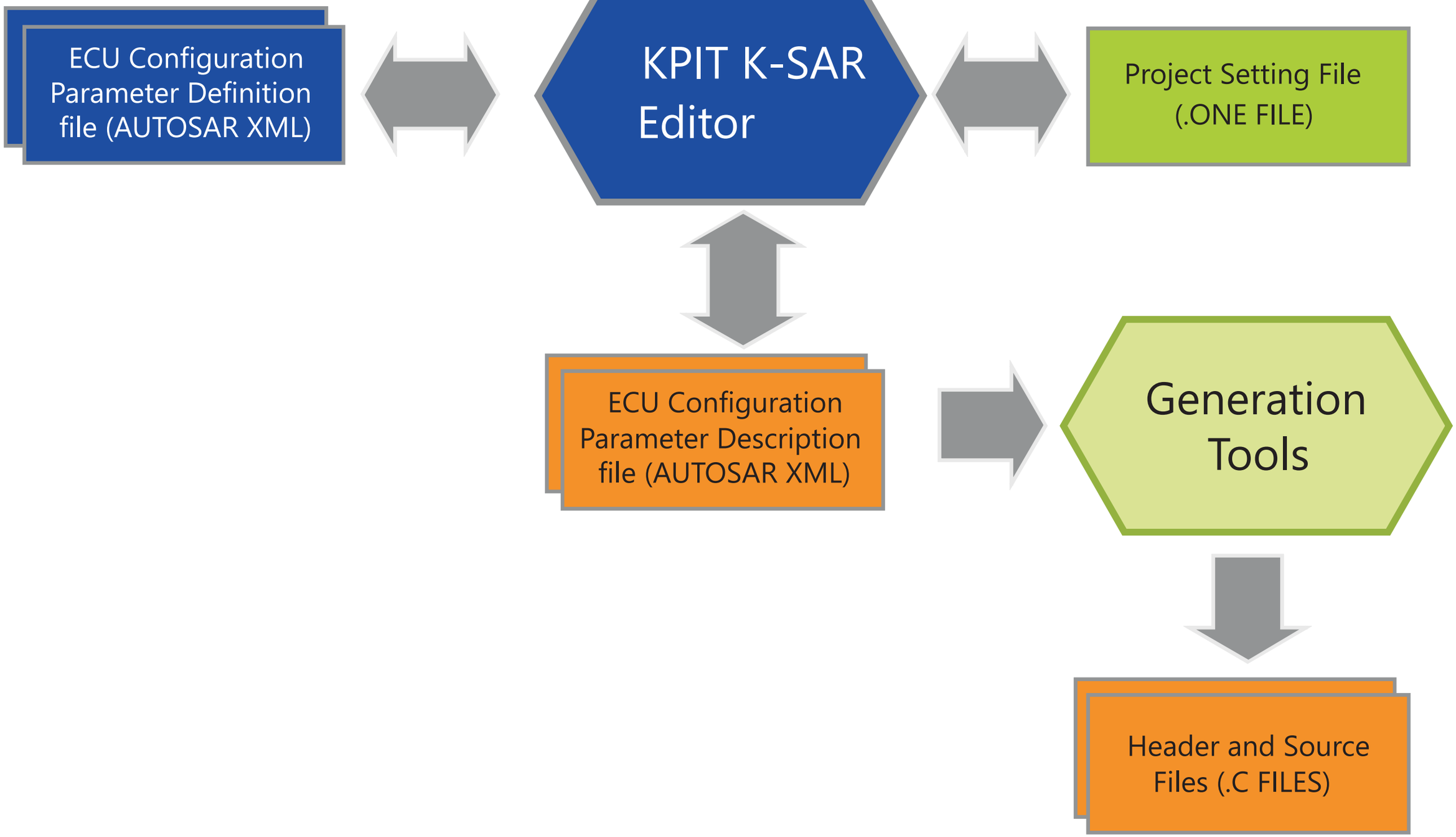



**Figure 15: K-SAR Editor Workflow – High-level**

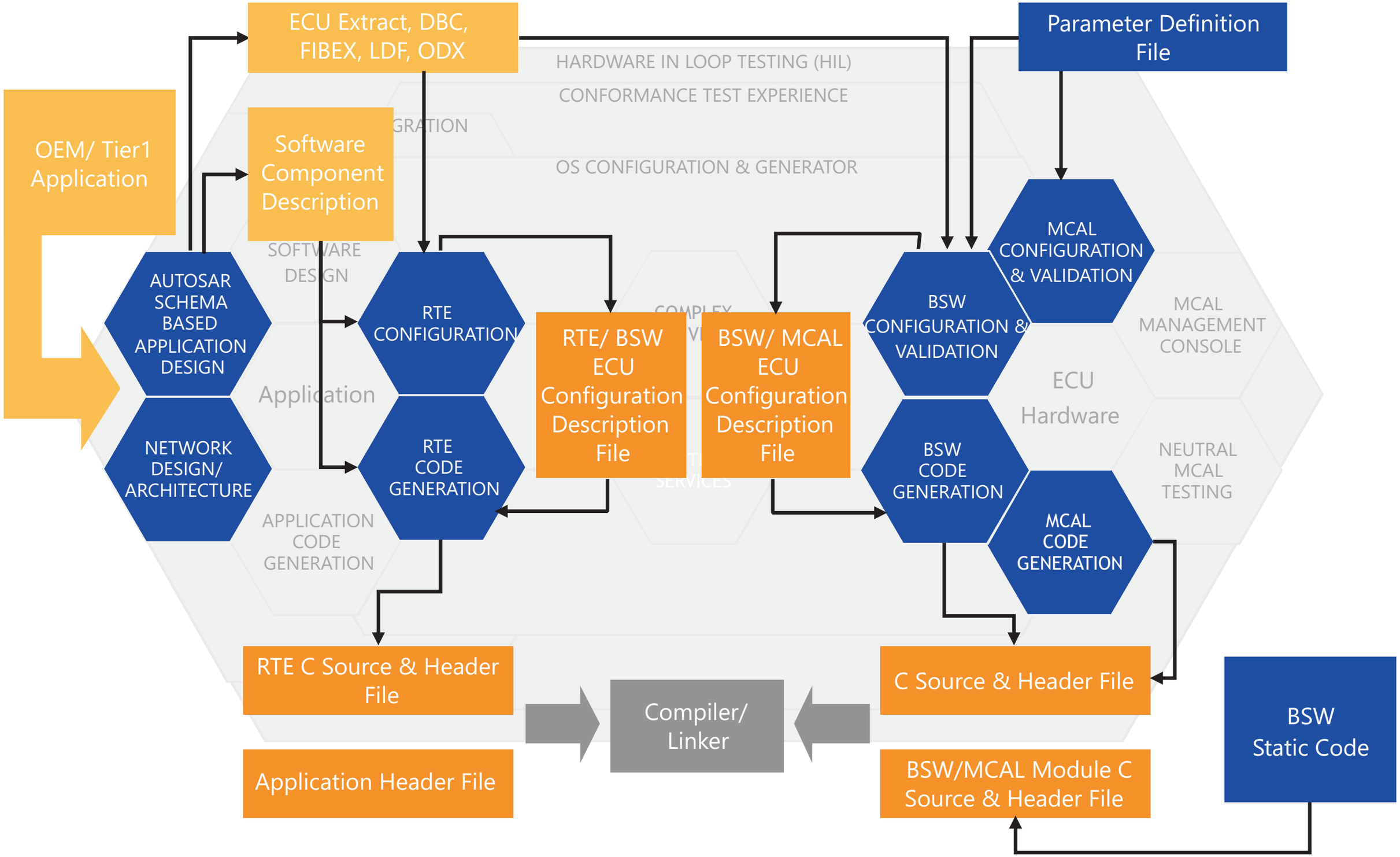
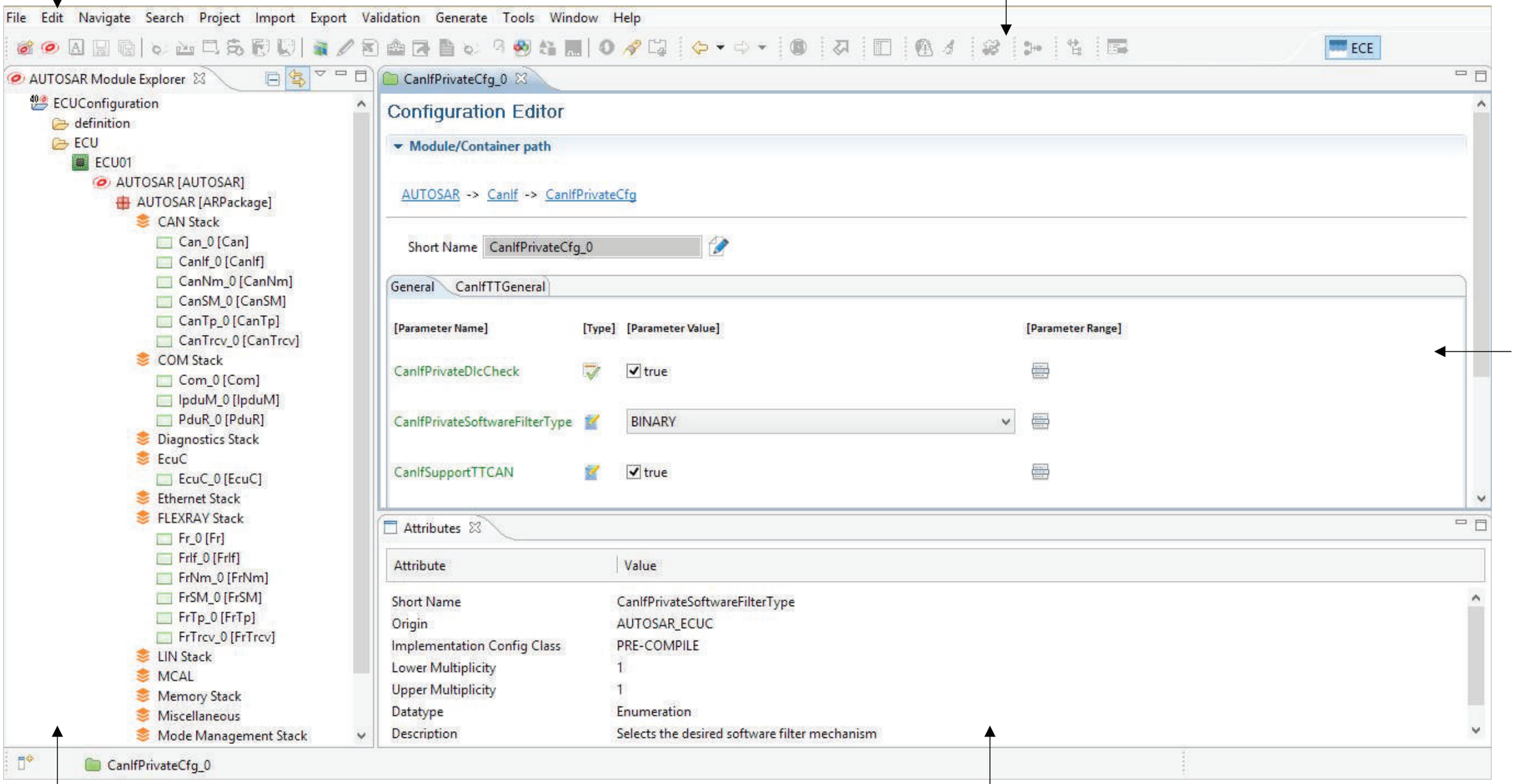



**Figure 16: K-SAR Editor Workflow – Detailed**

Menu Bar

Toolbar




Main window

Module Explorer

Attribute window

# K-SAR Editor Features

- Simple to use like any other Windows based tool
- User-friendly GUI
- Support for MRU (Most Recently Used) feature
- Validation - The module's configuration can be checked for correctness and completeness through validation.
- For any inconsistencies/dependencies, the Editor displays the Error(s) / Information(s) / Message(s) in the 'Message Info' window
- Storing and Loading of User Configuration data
- Import of AUTOSAR ECU Extract, DBC, LDF and Fibex data
- HTML Report Generation - Editor allows the user to get the summary of the currently loaded project
- Microsoft compliant Help Support
- Easy installation and setup
- Less memory consumption
- No dependency on any RTE

## Terms Used:

**Project:**
A Project is used to store the configuration

**Module:**
Modules denote an ECU Configuration Parameters Software Module. Individual modules are assigned a unique name. Number of module instances depends on multiplicity of the module.

**Container:**
Containers are used to group parameters and references. The number of instances of a container depends on multiplicity of the container.

**Sub-Container:**
A Sub-Container is also used to group parameters and references. Sub-container is a part of container. Sub-containers are defined in containers. The number of instances of a container depends on multiplicity of the container.

**Multiplicity:**

Multiplicity is used to specify how often a specific configuration element (module, container, parameter or reference) may occur in an ECU Configuration Description file. Lower-Multiplicity and Upper-Multiplicity are two attributes to specify minimum and maximum occurrences. In any case Lower-Multiplicity should be less than or equal to Upper-Multiplicity. Lower-Multiplicity is mentioned as 1 means that the element is mandatory. Lower-Multiplicity mentioned as 0, means that the element is an option. Upper-Multiplicity mentioned as * means that the parameter can occur any number of times.

**Multiple Conguration Set:**

It is used to allow the description of several ECU Configuration Sets.

**Template:**

Templates are definitions of configurable elements (Module, Container or Sub-Container). This format is taken from the Definition File.

**Calculation Formula:**

A Calculation formula is used to provide information about how the values can be computed. It utilizes references to address foreign elements to gather the required information.

# Part 9

# About KPIT

# KPIT AUTOSAR Expertise

- AUTOSAR development at KPIT started in early 2005 when we became AUTOSAR Premium Member. We are actively contributing to the standardization movement of AUTOSAR and have been appointed as a general contractor for writing Conformance Specifications for the AUTOSAR standard. We are also the general contractor for the AUTOSAR conformance test project.
- We are an AUTOSAR software platform focus company and endorse the AUTOSAR philosophy of "Cooperate on Standards".
- Our focus area in AUTOSAR is to develop AUTOSAR BSW Modules & MCAL drivers as part of the AUTOSAR stack and provide services around these Modules. We have developed the first complete MCAL product.
- In our endeavor to provide standard compliant, high quality, and production ready components we cooperate with specialized AUTOSAR tools. The expectation from the cooperation is to make our components compatible with these specialized AUTOSAR compliant tools.
- We are continuously supporting Network platforms to premium brands in Europe for last 10 years and have supplied platforms for about 150 ECUs which are integrated in the vehicles-on-road. This support is currently being extended to AUTOSAR.
- We are also the Premium member of JasPar.

# KPIT Advantages

- Faster time to complete AUTOSAR Conformance by supplying components that are compliant to AUTOSAR standard.
- Reduced in Bill of Material (BOM) costs through the Automotive-grade BSW components.
- Improved performance of platforms through Hand-coded, Optimized AUTOSAR BSW Modules including MCAL.
- Faster and full-proof AUTOSAR migration of the legacy Applications by designing Application Migration methodologies.
- Reduced in development overhead and overall Time-to-Market (TTM) by using KPIT AUTOSAR Tool chain

| | |
|---|---|
| **Enabling New Technology Adoption** | ▪ Executed AUTOSAR migration for safety ECU<br>▪ Strong experience in migrating safety critical applications to upcoming, improved communication protocols such as FlexRay |
| **Optimization** | ▪ AUTOSAR Safety ECU created with the Tier1performs better than legacy<br>▪ Unique Business Model makes the migration process more cost effective and smoother you will<br>▪ Customer does not get Locked to specific tool, vendor product or methodology |
| **Renovative Improvement** | ▪ KPIT Application Migration Methodology designed for extensibility and thus helps you design long term AUTOSAR strategy rather than experimenting with a pilot alone<br>▪ Migration methodology helps you to reuse your legacy toolchain and modifies them to adopt newer requirements as much as possible |
| **Time Benefit** | ▪ Onsite + Offshore model of engagement improves speed of execution with better efficiencies<br>▪ Already built and tested conformance test suites enable fastest path to production readiness |
| **Enabling Compliance** | ▪ AUTOSAR Conformance Spec experience helps you build solution that passes conformance tests quickly<br>▪ Open Test & Automation Framework invented by KPIT helps you test and validated performance of the system at multiple levels in the V Cycle<br>▪ Our Methodologies help you meet desired system performance |

**Figure 18: KPIT AUTOSAR Differentiation Diagram**

# Software Services / Products Provided by KPIT

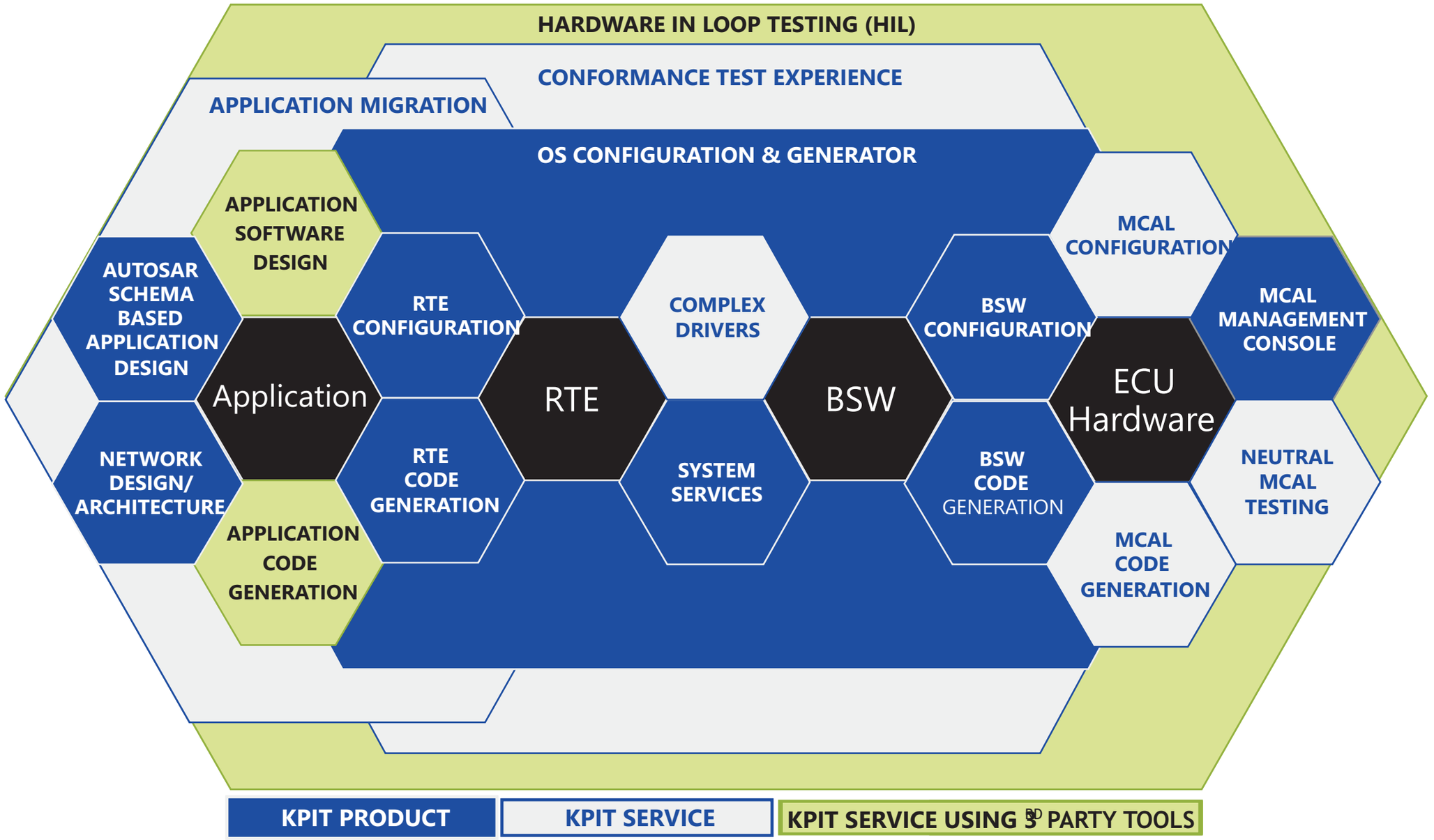



**Figure 19: Setting up AUTOSAR environment with KPIT**

[ THIS PAGE INTENTIONALLY LEFT BLANK ]

Legal Notice

We have made all efforts to offer current, correct and clearly-expressed information

All information in this handbook has been compiled meticulously; however, we cannot guarantee that the contents are completely accurate or free of errors. Neither the KPIT Technologies Ltd. nor the authors of this document accept any legal responsibility for its contents or any consequences, including direct or indirect liability, arising from its use.

KPIT Technologies Ltd. reserves the right to revise or change information contained in this document at any time without notice or justification to any person or entity.



V 1.0-0612

**Corporate Headquarters**
KPIT Technologies Ltd.
Plot No. 35 & 36,
Rajiv Gandhi Infotech Park
Phase 1, MIDC, Hinjewadi,
Pune 411 057
Phone: +91 20 6652 5000
Fax: +91 20 6652 5001

**China**
KPIT (Shanghai) Software
Technology Co. Limited
17A, Zhao Feng World Trade
Building, 369 Jiangsu Road,
Shanghai 200 050
Phone: +021 5631 5785
Fax: +021 5631 3925

**German Headquarters**
KPIT Technologies GmbH
Adams-Lehman-Str. 109,
80797 Munich
Phone: +49 89 3229 9660
Fax: +49 89 3229 9669 99

**Japan**
KPIT Technologies Ltd.
Muromachi CS Bldg. 5F
4-6-5, Nihonbashi - Muromachi
Chuo-KU, Tokyo 103 0022
Phone: +81 3 6913 8501
Fax: +81 3 5205 2434

**South Korea**
KPIT
B - 1001, USPACE2,
Sampyeong-dong, Bundang-gu,
Seongnam-si,
Gyeonggi-do 463-400, Korea.
Mobile: +82-10-3675-1401

**USA**
KPIT Infosystems Inc.
28970 Cabot Dr. Suite 100
Novi, MI 48377

* Premium Members