

VECTOR >

MICROSAR Adaptive

Product Catalogue

Contents

1. Overview	4
1.1 The MICROSAR Adaptive Stack	4
2. OS - Operating System	6
2.1 Operating System Application Layer (OsAl)	6
3. COM - Communication Management	7
3.1 Service-Oriented Communication (Com)	7
3.2 Communication Tracing (Tracing)	7
3.3 Protected End-to-End Communication (E2e)	7
3.4 External Communication (ExtCom)	8
3.5 Flexible Data Distributor (FDD)	8
3.6 Signal to Service Translation (S2s)	8
3.7 Secure Onboard Communication (SecOC)	8
3.8 Scalable Service-Oriented Middleware over IP (Somelp)	8
4. SEC - Security and Cryptography	9
4.1 Cryptography (Crypto)	9
4.2 Identity and Access Manager (IaM)	10
4.3 Intrusion Detection System Manager (IdsM)	10
5. MPC - Multi-Purpose Components	11
5.1 Http Transport Protocol (HttpClient/HttpServer)	11
5.2 Data Compression LZMA (DataCompression)	11
6. PER - Persistency	12
6.1 Persistency (Per)	12
7. MC - Measurement & Calibration	13
7.1 Logging (Basic)	13
7.2 Xcp	13
8. DIAG - Diagnostics	14
8.1 Diagnostic Manager for UDS (Diag)	14
8.2 DoIP Client	15
8.3 DoIP Server	15
8.4 In-Vehicle Diagnostic Access (IVDA)	15
8.5 Sovd Library	15

8.6	Sovd-2-Uds Adapter.....	15
8.7	UDS Gateway	15
9.	TSYN - Time Synchronization	16
9.1	Time Synchronization (TSyn)	16
10.	PHM - Platform Health Management	17
10.1	Platform Health Manager (PhM).....	17
11.	EXEC – Execution Management.....	18
11.1	Execution Management (Exec)	18
12.	SM - State Management Framework.....	19
12.1	State Management Framework (SMF)	19
13.	UCM - Update & Configuration Management	20
13.1	Update and Configuration Management (UcM)	20
14.	NM – Network Management	21
14.1	UDP Network Management Protocol Specification (UdpNm).....	21

1. Overview

1.1 The MICROSAR Adaptive Stack

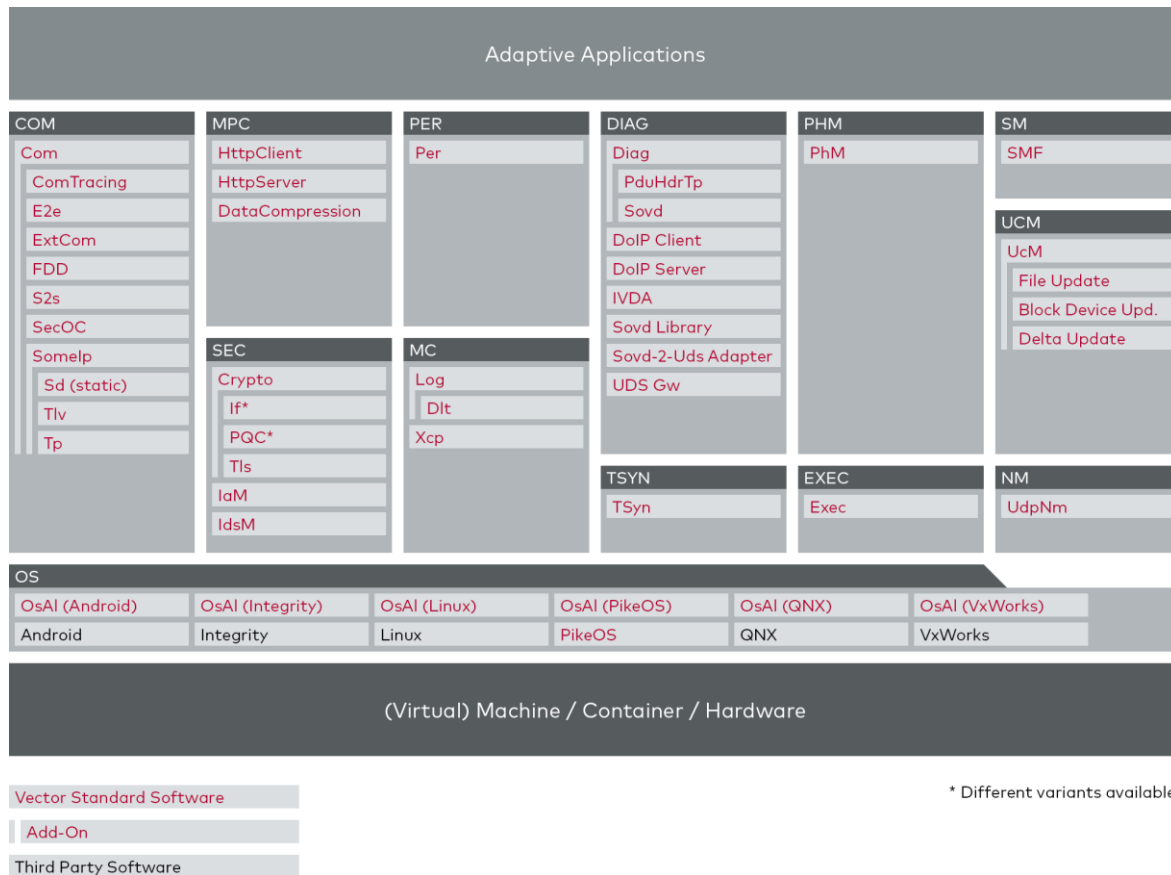


FIGURE 1: MICROSAR Adaptive BSW stack

MICROSAR Adaptive is an advanced and versatile software base layer solution designed to meet the dynamic requirements of modern automotive systems. Ideal for high-performance ECUs such as in-vehicle application servers, ADAS ECUs and infotainment systems, it provides the flexibility and dynamic behavior required for today's automotive applications.

MICROSAR Adaptive features object-oriented programming and service-oriented communication, enabling developers to build modular, reusable and maintainable software components.

Benefit from a proven solution that enables efficient development, global collaboration, safety, and robust security to drive the future of automotive engineering.

- > In series use for many years
- > Development for safety-related applications up to ASIL D
- > High-performance communication via Zero Copy
- > Always up to date by continuous provision of software updates
- > Support of various POSIX-based operating systems
- > Quick start-up by included example projects
- > Seamless interoperability with MICROSAR Classic ECUs
- > Over-The-Air (OTA) software updates
- > Smooth integration into DevOps environments
- > Communication integrity between all applications, whether they are in the same or different ECUs
- > All necessary automotive functions:
 - > Diagnostic stack including fault memory
 - > Logging and Debugging

> Global Time Synchronization

For more information about MICROSAR Adaptive visit vector.com/microsar-adaptive

2. OS - Operating System

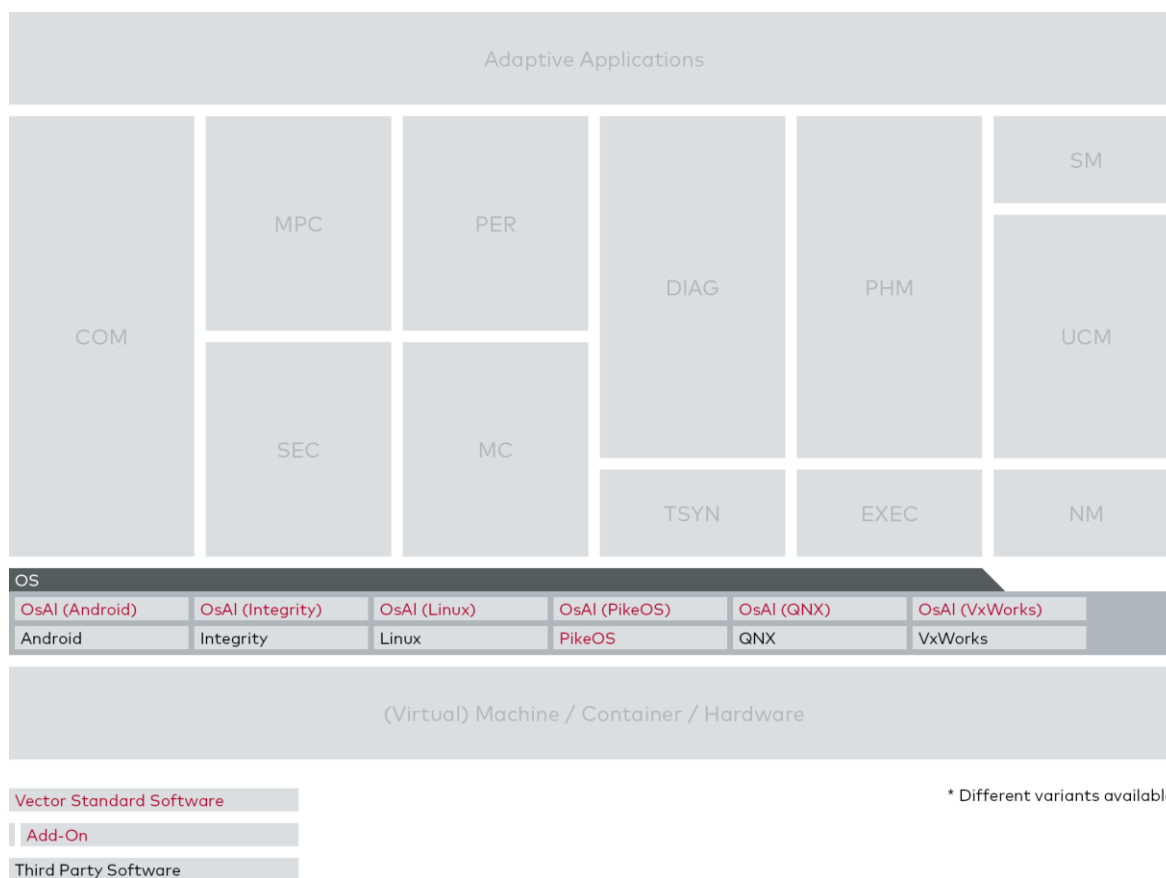


FIGURE 2: OS cluster

MICROSAR Adaptive Operating System contains Application Layer (OsAI) which provide a common API for the portability of software on higher architectural layers. MICROSAR Adaptive does not contain any hardware-dependent modules or drivers, as the hardware dependency is mainly covered by the respective Operating System. Operating Systems have differences in their API and available feature set. All modules above the OS layer are using OsAI for OS interaction and are therefore OS independent.

- > Process Management and Event Management
- > Signal Handling and Clocks and Timers
- > Network Communication and Inter Process Communication
- > Shared Memory
- > System Resource Information and System Log

2.1 Operating System Application Layer (OsAI)

Application Layers are available for Android, Integrity, Linux, PikeOS, QNX and VxWorks.

3. COM - Communication Management

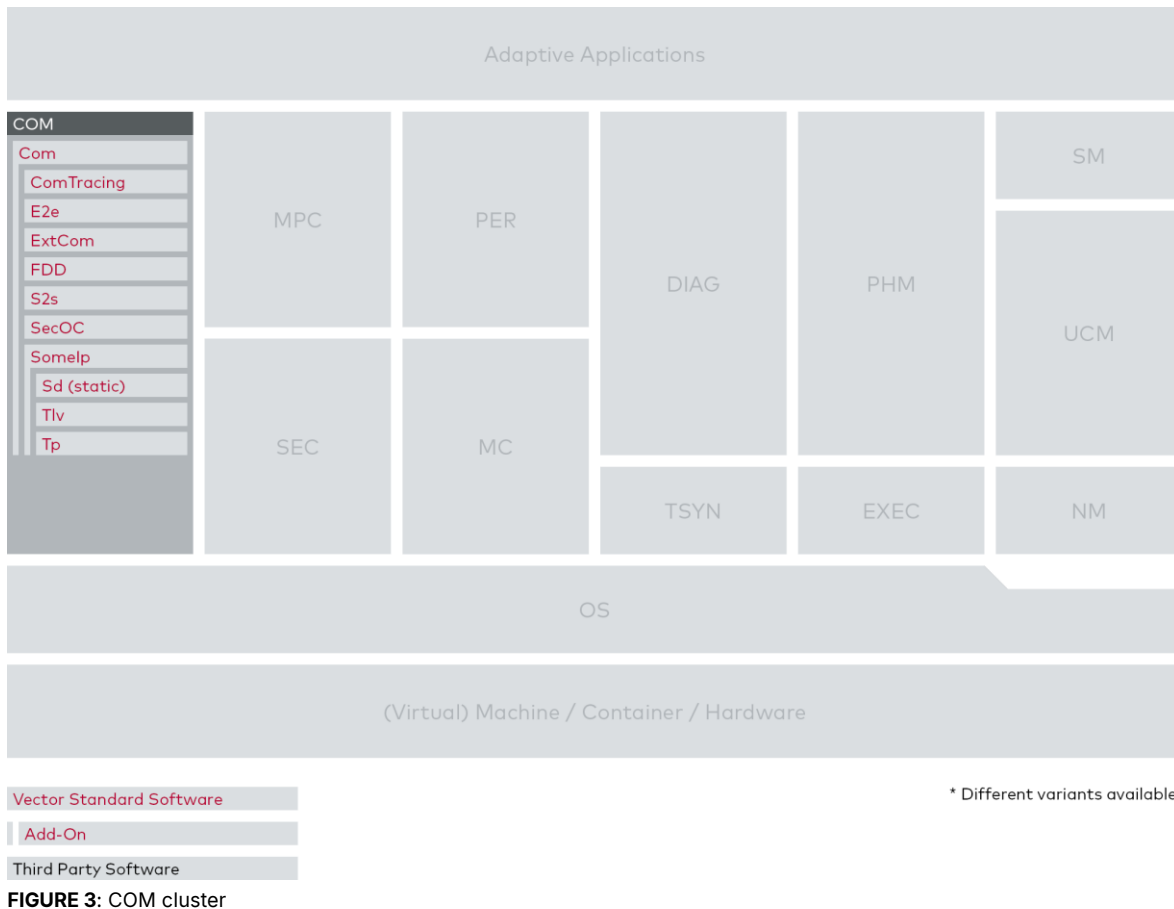


FIGURE 3: COM cluster

MICROSAR Adaptive Communication Management provides service-oriented communication (SOC) between applications or components of the MICROSAR Adaptive platform via an abstract C++ API that uses communication protocols such as SOME/IP (Scalable Service-Oriented Middleware over IP), IPC (Inter-Process Communication) or ZC (Zero-Copy).

3.1 Service-Oriented Communication (Com)

The Com module provides middleware for service-oriented communication, featuring two IPC mechanisms for intra-ECU connectivity. These are a shared memory-based channel for direct process interaction and an enhanced Zero-Copy binding. When preconditions are satisfied, the Zero-Copy technique supports multicast without incurring overhead and eliminates data copying along the communication path.

3.2 Communication Tracing (Tracing)

Ara::com facilitates accessing IPC between Adaptive Applications, providing both IPC-Binding and Zero-Copy Binding for communication measurement. This enables tracing application interactions. The Tracing component exposes all service-oriented communication trace points, allowing users to access relevant communication events and their payloads.

3.3 Protected End-to-End Communication (E2e)

The E2e module ensures the integrity of the data exchange between a sender and a receiver (sender protects data with E2E, receiver checks with E2E whether the data is properly protected). E2E profiles are available for this purpose.

3.4 External Communication (ExtCom)

The ExtCom module sets up communication between a MICROSAR Adaptive software stack and other software stacks on different operating systems using IPC channels outside the standard framework. It includes methods like shared memory channels, SPI, and third-party communication libraries.

A gateway facilitates seamless integration by connecting to the SOME/IP daemon, acting like a remote SOME/IP node. Additionally, messages can be directly processed within an application using a special version of the SOME/IP and S2S binding.

3.5 Flexible Data Distributor (FDD)

The FDD module functions as a central gateway application, managing the receipt, repackaging, and distribution of service and signal-based communications to various applications. It uses passthrough connectors to split PDUs and creates new services to combine signals from multiple sources.

3.6 Signal to Service Translation (S2s)

The S2s add-on enables the translation of signal-based (Classic ECUs) to service-oriented communication (Adaptive ECUs) and vice versa.

3.7 Secure Onboard Communication (SecOC)

Secure Onboard Communication (SecOC) protects the authenticity and freshness of transmitted SOME/IP and signal-serialized PDUs using cryptographic message authentication codes. It employs a Freshness Value Management module to guard against replay attacks by ensuring freshness values are used in communication.

3.8 Scalable Service-Oriented Middleware over IP (Somelp)

The SOME/IP protocol is a lightweight IP-based communication middleware that enables data exchange between ECUs within a vehicle and can be used together with the service discovery protocol to achieve dynamic behavior typical of service-oriented communication.

Static Service Discovery (Sd static)

Sd static allows to preconfigure the location of the communication nodes (service discovery not necessary).

Tag Length Value (Tlv)

- > Tlv brings meta data into the serialization, enhances forward/backward compatibility, and enables optionality of data elements.

SOME/IP Transport Protocol (Tp)

- > Tp allows sending UDP messages that don't fit in a single ethernet frame, even above the limit of IP fragmentation of 64kb.

4. SEC - Security and Cryptography

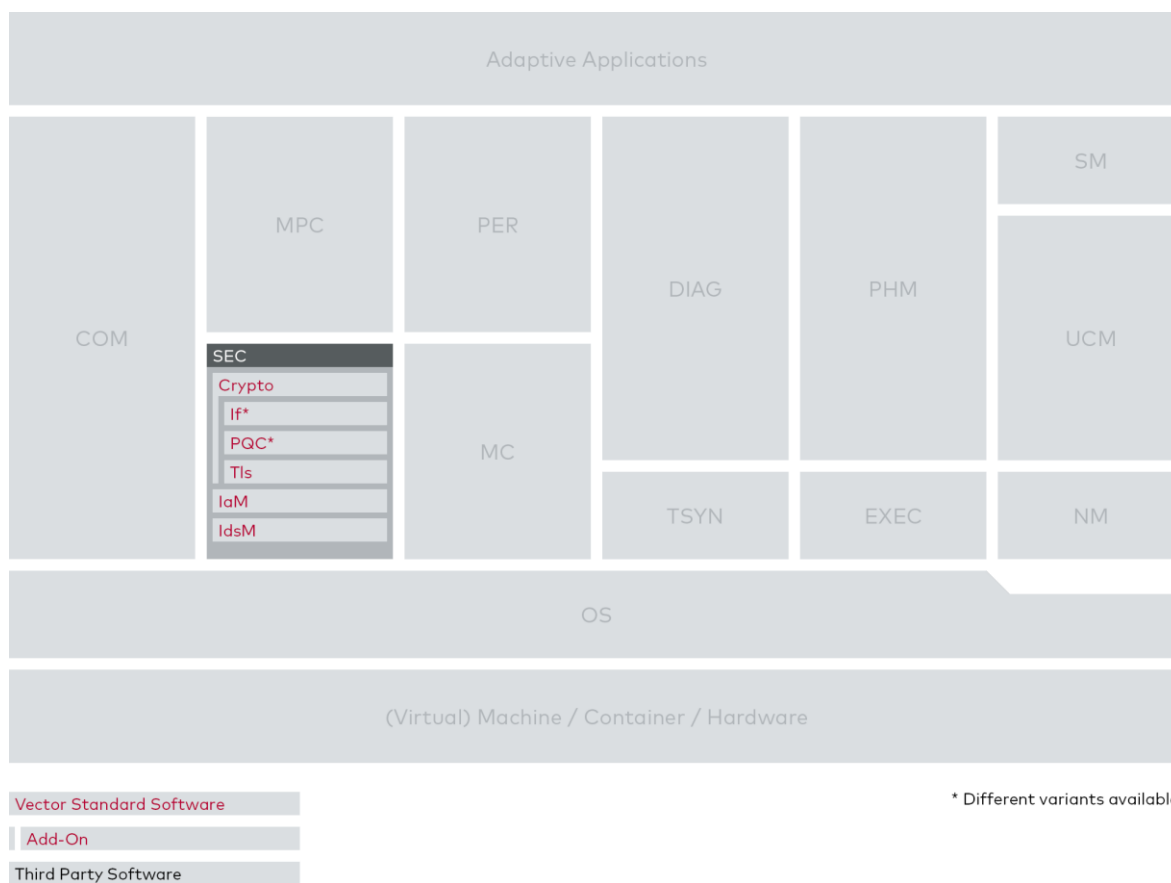


FIGURE 4: SEC cluster

MICROSAR Adaptive Security and Cryptography provides security functionality that can be used to support security use-case and achieve compliance to cybersecurity regulations. The functionality includes cryptographic operations, certificate management, access control, and intrusion detection.

4.1 Cryptography (Crypto)

Crypto provides a software implementation of common cryptographic algorithms, key storage functionality in file system, and functions to handle X.509 certificates. All functionality is exposed via the `ara::crypto` API.

If

- > The Crypto HSM If provides access to a platform specific hardware security module (HSM) through the `ara::crypto CryptoProvider` API.

Asymmetric PQC Algorithms

- > With Crypto PQC, a software implementation of Post-Quantum algorithms is provided that is usable via `ara::crypto`.

Transport Layer Security for SOME/IP (Tls)

- > Transport Layer Security (TLS) is a secure communication protocol used to authenticate communication partners using certificates or pre-shared keys and protect the authenticity and confidentiality of exchanged data.

4.2 Identity and Access Manager (IaM)

IaM provides access control on services provided and consumed via ara::com. This enables the enforcement of the principle of least privilege on service oriented communication.

4.3 Intrusion Detection System Manager (IdsM)

IdsM receives security events (SEv) from components of the basic software or application and filters them based on its configuration. The now qualified security events (QSEv) can be forwarded to the Intrusion Detection System Reporter (IdsR – available in MICROSAR Connect), from where they are in turn forwarded to a Security Operations Center (SOC) at the vehicle manufacturer.

5. MPC - Multi-Purpose Components

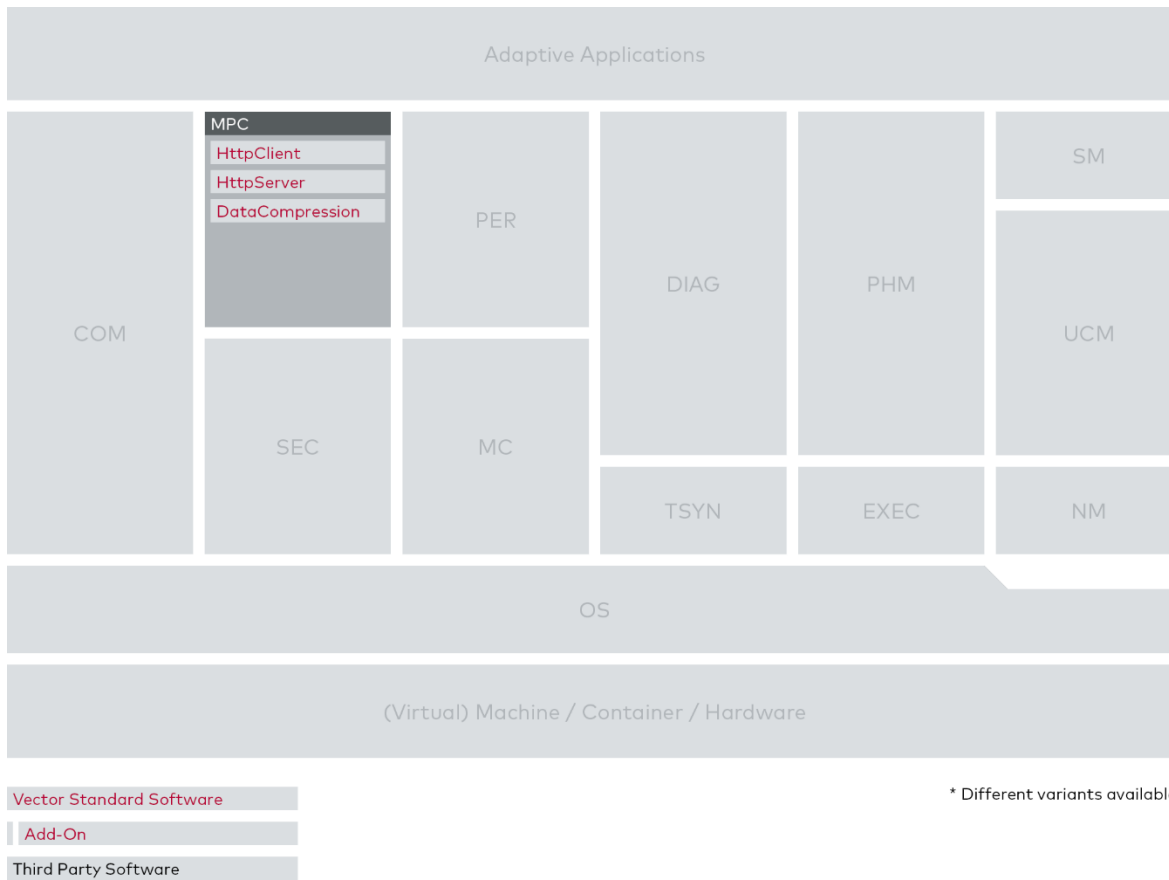


FIGURE 5: MPC cluster

MICROSAR Adaptive Multi-Purpose Components provides modules for base functionality such as data transfer and decompression.

5.1 Http Transport Protocol (HttpClient/HttpServer)

The Hypertext Transfer Protocol transmits browser requests from a client to a server.

5.2 Data Compression LZMA (DataCompression)

DataCompression allows download of compressed data based on a LZMA compression algorithm with focus on the core functionality for LZMA2 decompression (LZMA2 decoder, CRC64 check, default compression level 6).

6. PER - Persistency

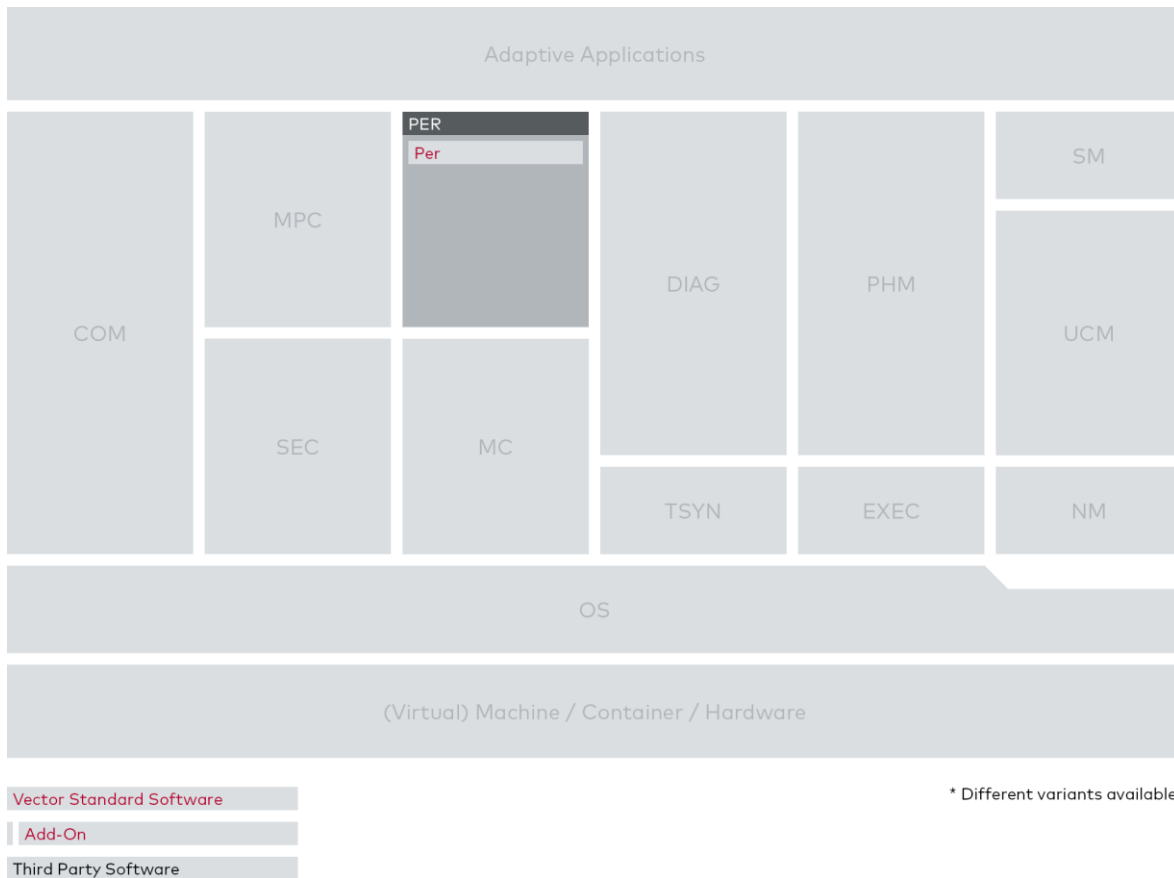


FIGURE 6: PER cluster

MICROSAR Adaptive Persistency is a library-based implementation and offers mechanisms to Adaptive applications to store information persistently in the non-volatile memory. The API specification includes key-value storage as well as plain file access.

- > KV database handling: Opening a KV database, discard pending changes.
- > KV pairs handling: Listing and removing all keys of an existing KV database, read/write or update all its KV pairs.
- > FileStorage handling: Opening a FileStorage, creating a new FileStorage, listing and removing all file names of an existing FileStorage, reading/writing or updating all files of the FileStorage
- > File handling: Opening new or existing files for write-only, read-only, or readwrite, accessing file contents using the access modes text (get single lines separated by a user defined special separator character, e.g. CR/LF) and binary (opens the file as binary, without any conversions of special characters), truncating a file (when the file is opened for writing), locating the file pointer at a desired seek position within the file (at the beginning, the end or in between, relatively or absolutely), adding newline to text-files, deleting a file from the FileStorage it is assigned to)

6.1 Persistency (Per)

Module to perform persistency functions.

7. MC - Measurement & Calibration

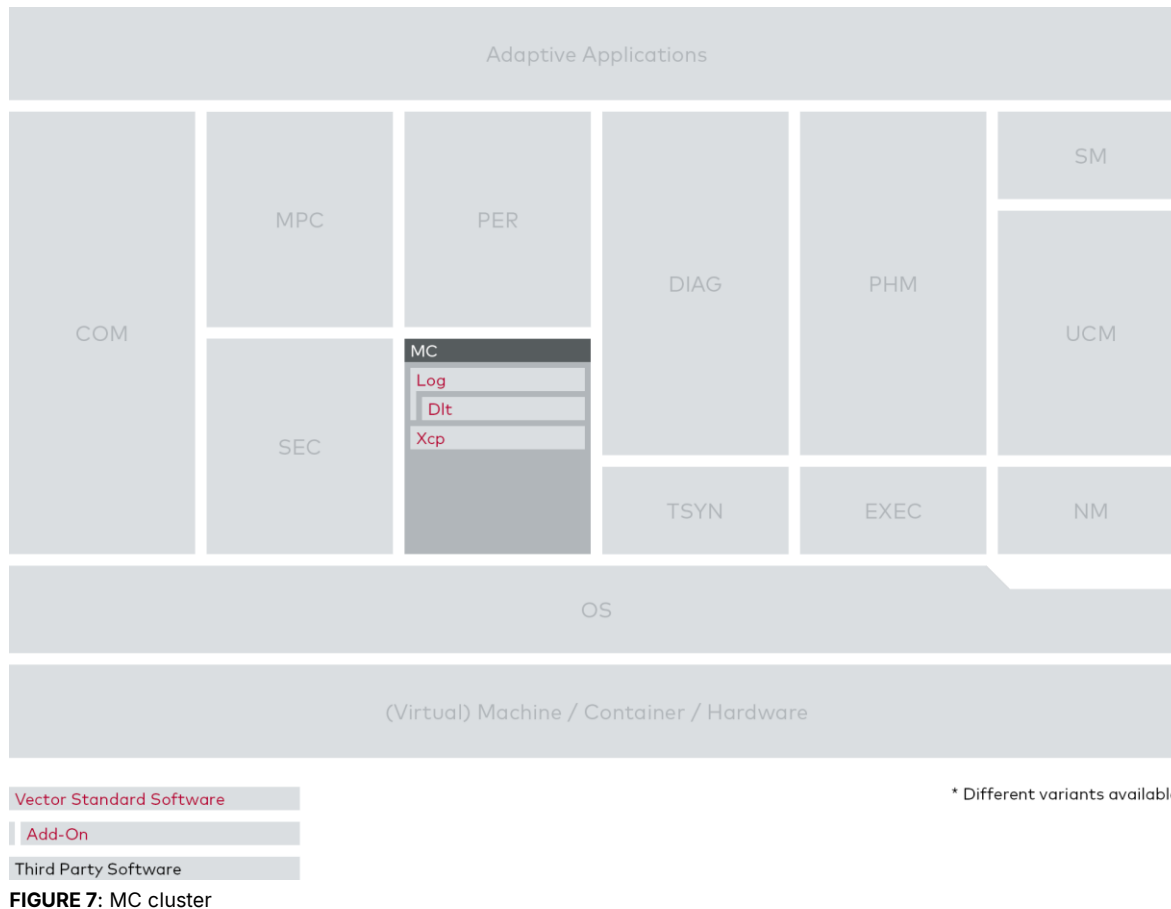


FIGURE 7: MC cluster

MICROSAR Adaptive Measurement & Calibration provides interfaces for applications to forward logging information to the communication bus, console, or file system. It is used to configure application-wide default log settings and process log messages. These are written to configured log sinks: console, file, DLT (Diagnostic Log and Trace module), or syslog. To forward log messages via the DLT, the Adaptive Log and Trace (ALT) must be present. To forward messages via Syslog, any Syslog daemon must be available.

- > Can be used during the development phase to log in and store debugging information externally on a device.
- > The specification is based on the Scope of Operation for Diagnostic Log and Trace (DLT) with extensions.

7.1 Logging (Basic)

Logging (Basic) enables logging of status information from the application and basic software, and by default allows logging in to the operating system console. This functionality includes logging to a log file that is stored locally in the ECU.

Diagnostic Log and Trace (Dlt Framework)

- > The Dlt Framework option provides the possibility to communicate status information using the AUTOSAR DLT protocol. It enables external access to log information without the need to access the console of the OS.

7.2 Xcp

Xcp is a protocol for communication between a master (PC tool) and a slave (Adaptive Application). It is standardized by ASAM and is mainly used for measuring, calibrating and testing Adaptive Applications. XCP supports the bus system Ethernet (EthXcp).

8. DIAG - Diagnostics

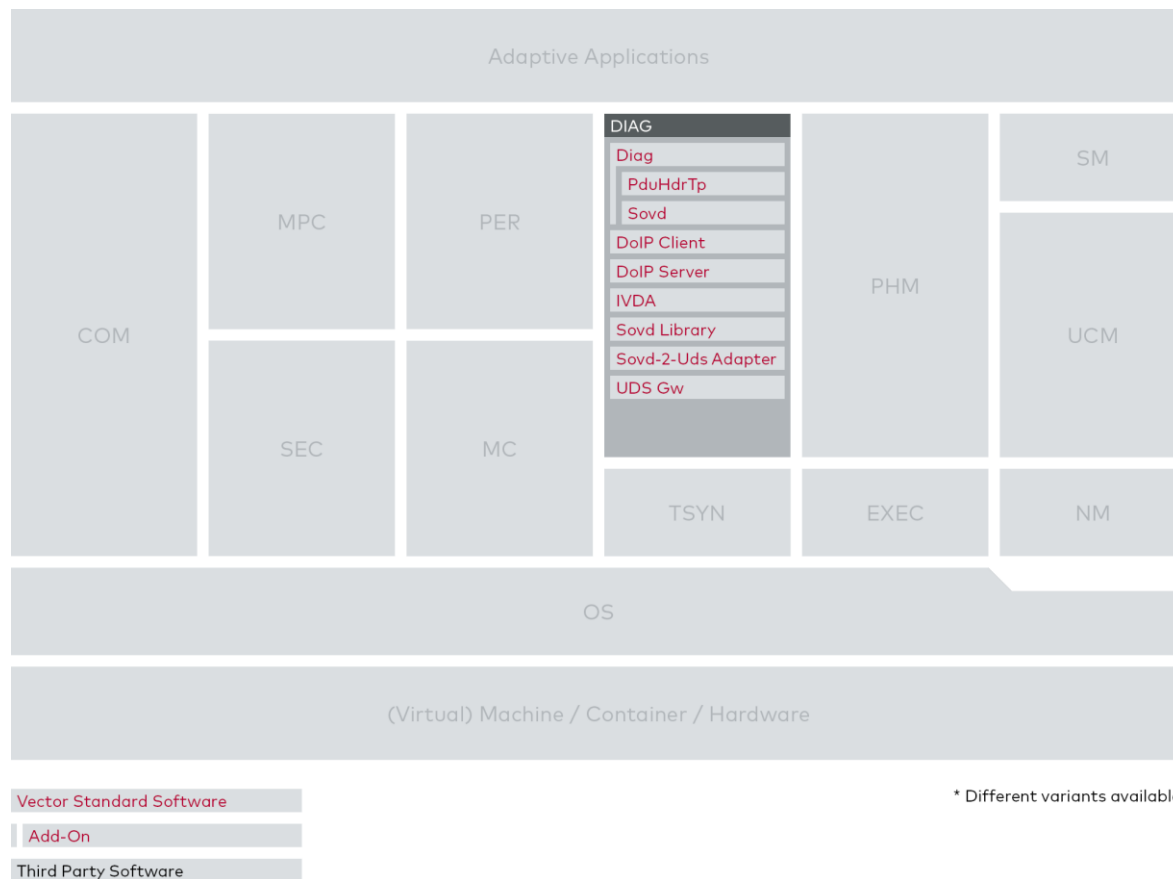


FIGURE 8: DIAG cluster

MICROSAR Adaptive Diagnostics features Diagnostic Service handling and Diagnostic Fault Memory handling. The solution is designed to support the following Diagnostic protocols:

- > UDS (ISO 14229)
- > SOVD (ISO 17978)

8.1 Diagnostic Manager for UDS (Diag)

The Diag module realizes an UDS server implementation according to ISO 14229. The solution features a distributed architecture, composed of the central Diagnostic Manager (DM) daemon and multiple `ara::diag` libraries. The `ara::diag` libraries represent the interface for user applications to implement against. Overall the solution supports:

- > Diagnostic Service handling
- > Diagnostic Fault Memory handling
- > Support for multiple and custom transport protocols

Available transport protocols:

PDU/HDR TP Binding (PduHdrTp)

- > PDU/HDR TP Binding is a transport protocol based on SoAd PDUs. It is a TCP/IP based transport protocol that encapsulate UDS payload in the SoAd PDUs.

Service-Oriented Vehicle Diagnostics (Sovd)

- > SOVD (ISO 17978) is a modern HTTP/REST based Diagnostic protocol. The DM implements a native SOVD server and helps realizing SOVD methods as specified in ISO 17978. This functionality requires Http Server.

8.2 DoIP Client

DoIP implements the DoIP transport protocol as specified in ISO 13400. DoIP is a transport protocol for diagnostic communication over Ethernet. The DoIP Client realizes a client implementation for DoIP (ISO 13400). To realize an In-Vehicle UDS Client it is recommended to additionally use the In-Vehicle Diagnostic Access (IVDA), which handles the UDS Session layer and provides a interface to user applications.

8.3 DoIP Server

The DoIP Server realizes a server implementation for DoIP.

8.4 In-Vehicle Diagnostic Access (IVDA)

The In-Vehicle Diagnostic Access (IVDA) provides Diagnostic access to function as an In-Vehicle UDS Client. This includes the UDS connection and session handling on the DoIP Transport Protocol.

It provides an interface to the user application to be used as an on-board Diag Tester, to requests Diagnostic services within the Vehicle.

8.5 Sovd Library

The Sovd Library is the standalone solution for Service-Oriented Vehicle Diagnostics (SOVD). This functionality requires Http Server.

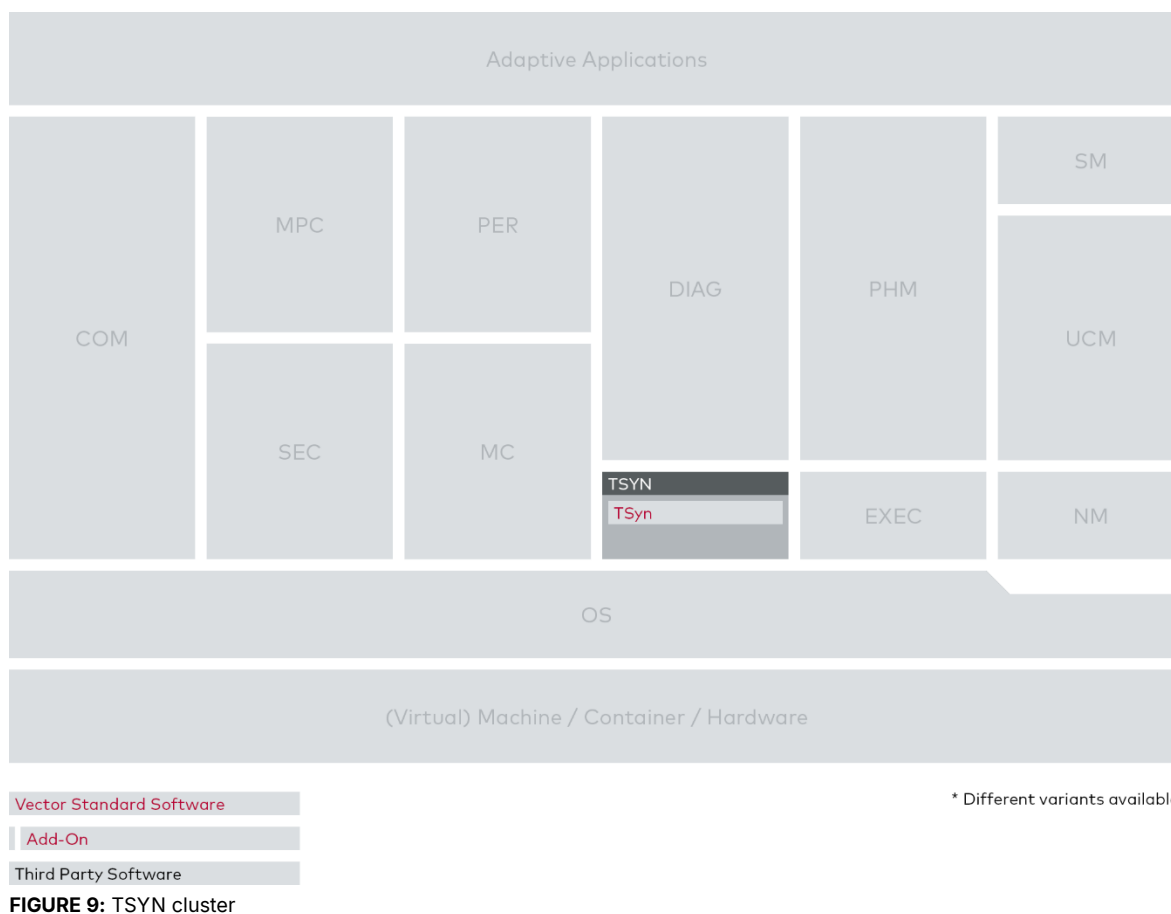
8.6 Sovd-2-Uds Adapter

The Sovd-2-Uds Adapter is a standalone daemon which translates SOVD requests to UDS requests and response vice-versa. It allows the easy integration of UDS based ECUs into an SOVD architecture. The translation is based on the Diagnostic descriptions (ODX) of the respective UDS based ECUs. This functionality requires Http Server, IVDA and DoIP Client.

8.7 UDS Gateway

The Uds Gateway supports routing of Unified Diagnostic Services (UDS) messages between different networks. It features a modular architecture, that allows easy integration of different protocol handlers. Third party handlers can be easily integrated and out of the box the Uds Gateway works with the Vector components DoIP Server, DoIP Client and Diagnostic Manager.

9. TSYN - Time Synchronization



MICROSAR Adaptive Time Synchronization provides synchronized time for the Adaptive applications within and between the ECUs. It is responsible for managing the time bases and for communication with off-board timesync masters and slaves.

- > Synchronized Slave Time Base (Synchronization to a reference clock over ethernet)
- > Pure Local Time Base (Synchronization of time within one ECU)
- > Rate correction for Slaves (Detection and elimination of rate deviations of local instances of Time Bases)
- > TimeLeap (Monitoring of time leaps)
- > Timeout (Monitoring of communication timeouts)
- > UserData (Retrieving for Synchronized Slave Time Base, setting and retrieving for Pure Local Time Bases)

9.1 Time Synchronization (TSyn)

Module to perform time synchronization functions.

10. PHM - Platform Health Management

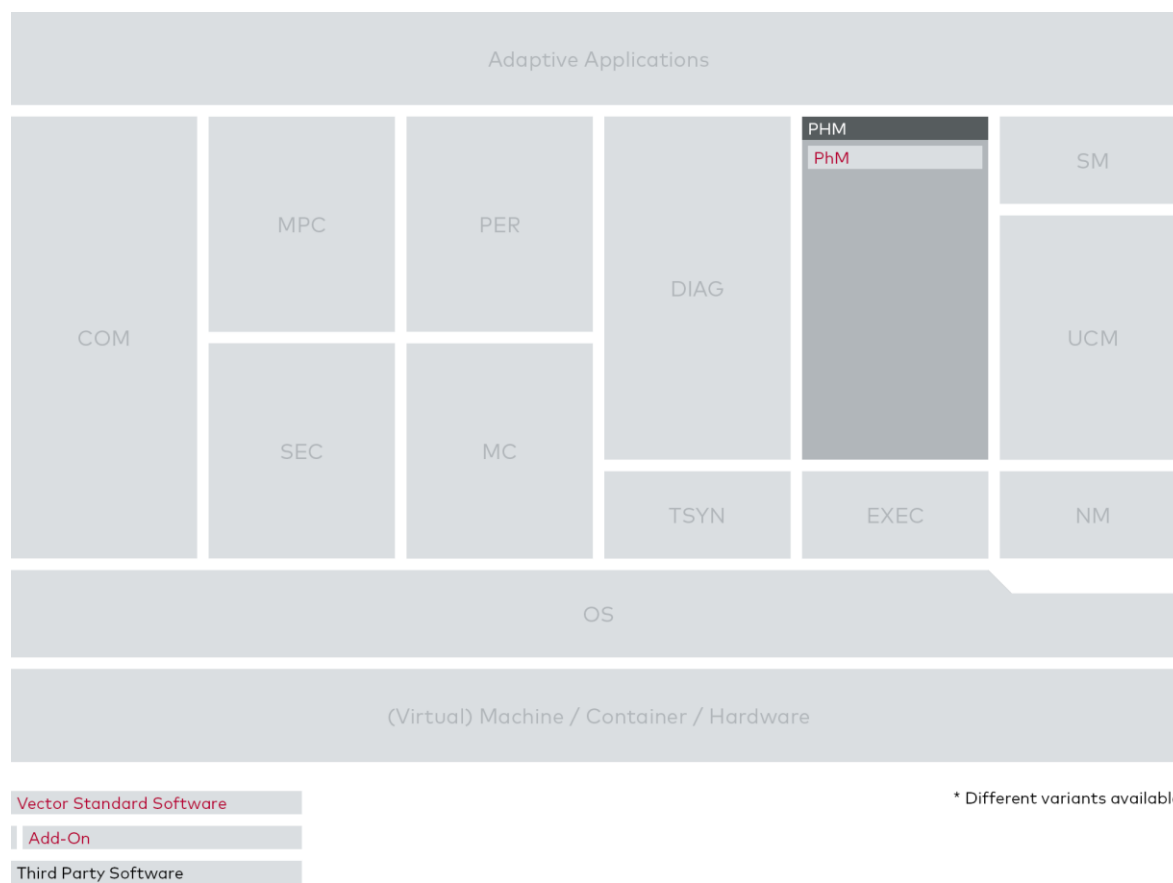


FIGURE 10: PHM cluster

MICROSAR Adaptive Platform Health Management supervises dedicated Adaptive applications and provides an interface to report Checkpoints of Supervised Entities and Health Statuses of Health Channels. In case of detected failures, the PHM notifies the State Management via Recovery Notifications. PHM monitoring is particularly relevant for functional safety.

- > Supervision of Alive, Deadline, Logical, Local, Global, and Health Channel
- > Recovery Notifications

10.1 Platform Health Manager (PhM)

Module to perform platform health management functions.

11. EXEC – Execution Management

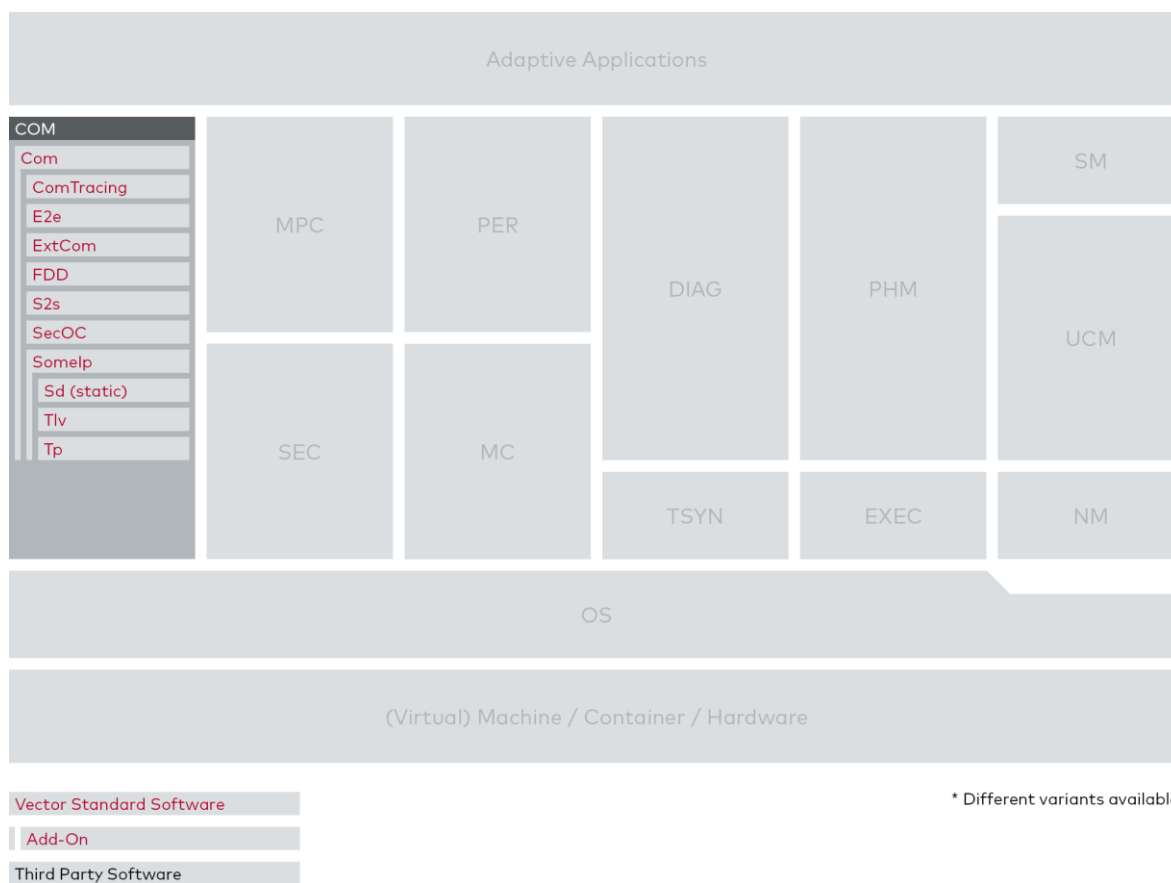


FIGURE 11: EXEC cluster

MICROSAR Adaptive Execution Management is primarily responsible for starting and stopping Adaptive applications. It is the first process to be started (init) and controls the other processes according to the requested state of the machine. To do this, it works with the operating system (OS) to perform application runtime scheduling.

- > Process Lifecycle Management and State Management
- > Application Recovery Actions
- > Resource Limitation

11.1 Execution Management (Exec)

Module to perform execution management functions.

12. SM - State Management Framework

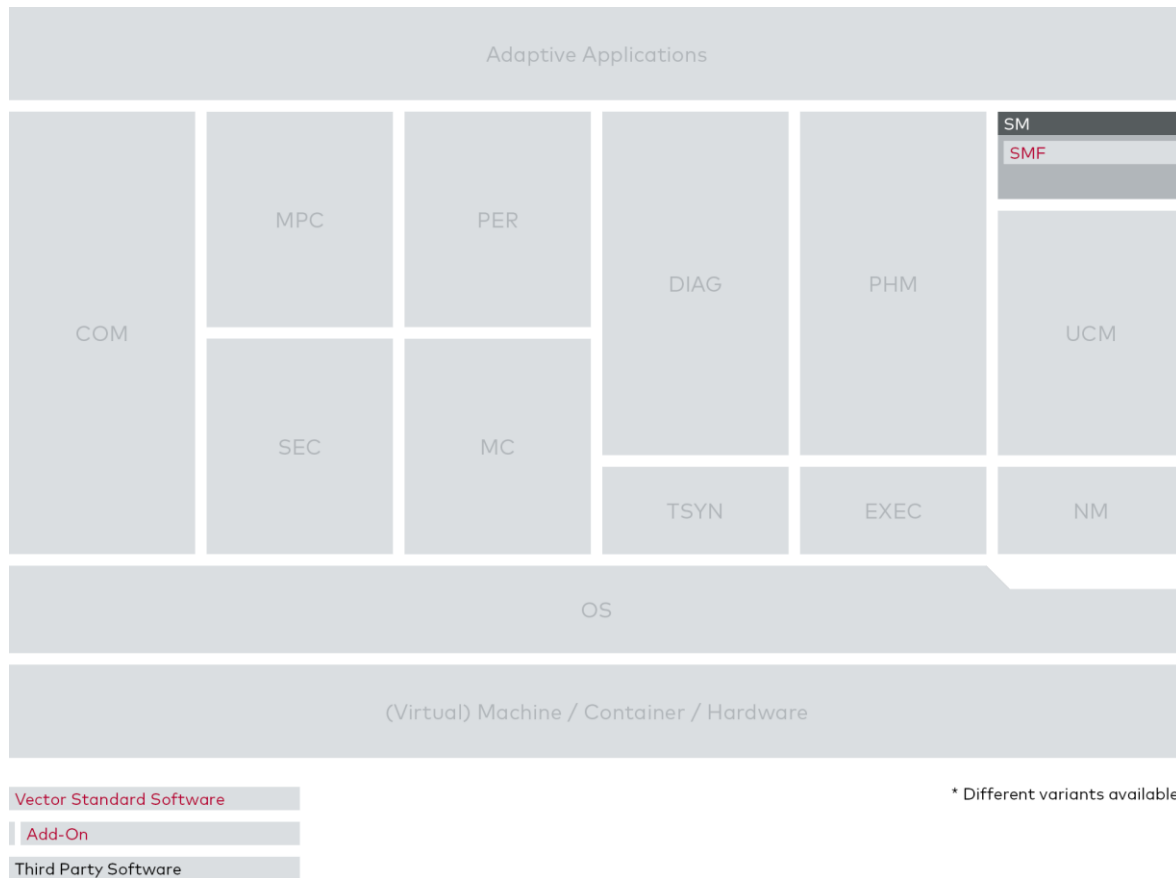


FIGURE 12: SM cluster

State Management oversees the operational lifecycle and state transitions of an Adaptive machine. It interacts with components like Execution Management, Network Management, Update Management, Diagnostics, Platform Health Management, and the Operating System to manage these tasks efficiently.

12.1 State Management Framework (SMF)

The State Management Framework (SMF) of MICROSAR Adaptive provides building blocks for creating project-specific state managers but lacks an off-the-shelf state manager implementation. This framework offers higher-level abstractions like events, guards, and actions for interaction with relevant Functional Clusters. Key offerings include:

- > **State Machine:** Create custom state machines with states, transitions, events, guards, and actions.
- > **Function Group Control:** Manage functional group states and respond to Execution Management changes for lifecycle purposes.
- > **Update Control:** Handle update session requests, verify updates, and prepare rollbacks.
- > **Network Control:** Manage logical network states and interact with Network Management changes.
- > **Runtime:** Schedule periodic tasks and handle necessary POSIX signals and I/O.

Additionally, templates are available to assist in developing a state manager with the framework. All components are provided as C++ libraries dependent on MICROSAR Adaptive-specific C++ APIs.

13. UCM - Update & Configuration Management

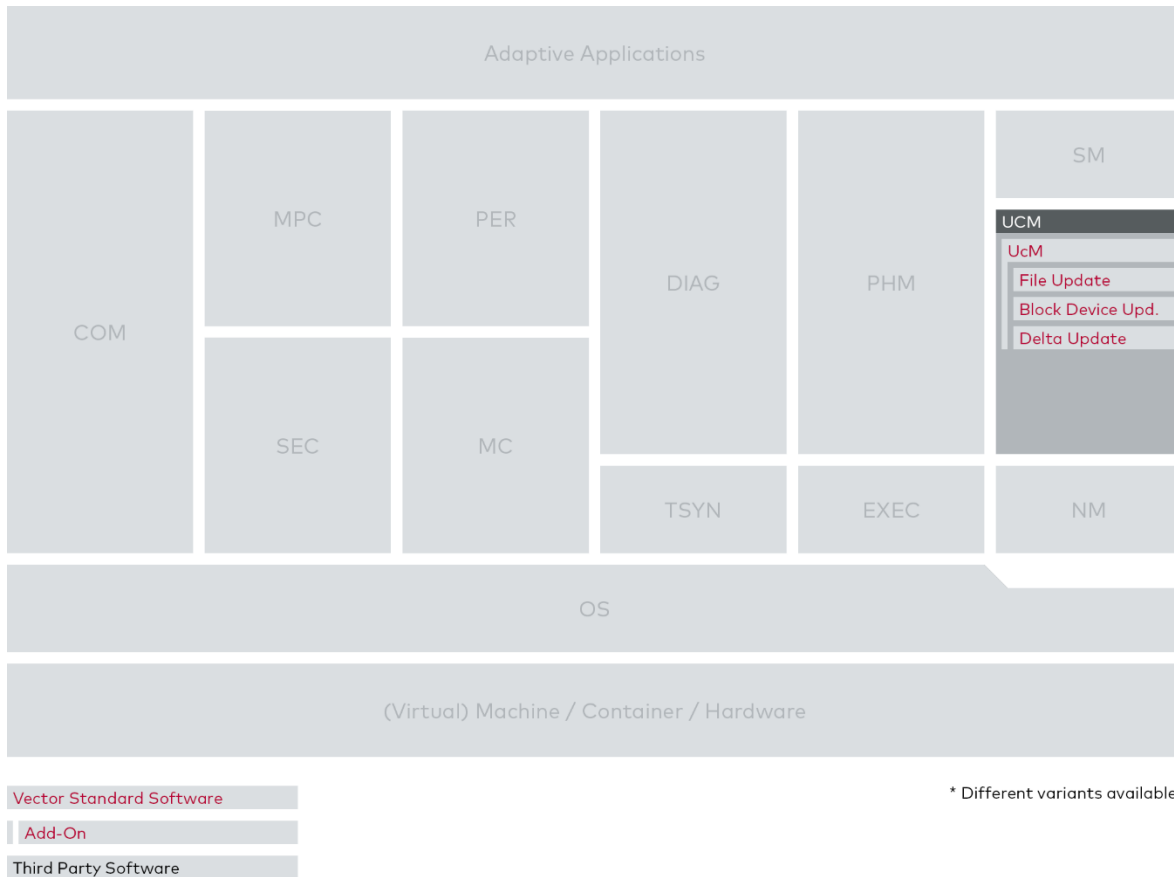


FIGURE 13: UCM cluster

MICROSAR Adaptive Update and Configuration Management is responsible for reporting of installed software clusters and apply changes to the installed software clusters by processing software packages. Software clusters are logical entities that correspond to, for example, Adaptive applications, the Adaptive Platform, or the OS itself.

- > Transfer and authentication of software packages
- > Processing of software packages to install, update or remove software clusters
- > Activation and rollback of changes to software clusters
- > Update state handling in coordination with state management
- > Transfer and update strategy interface to allow project-specific installation routines
- > Extension of the AUTOSAR package management interface to support suspend & resume of updates
- > Robustness to ensure update capability even in abnormal termination scenarios

13.1 Update and Configuration Management (UcM)

Module to perform update and configuration management functions.

Add-Ons

- > **File Update:** File-based installation of archives in the file system
- > **Block Device Update:** Installation of data to block devices. Assumes access to the block device via device file in the file system
- > **Delta Update:** Allows to apply differential updates, currently planned for Block Device Update Strategy, later also planned for File Update Strategy.

14. NM – Network Management

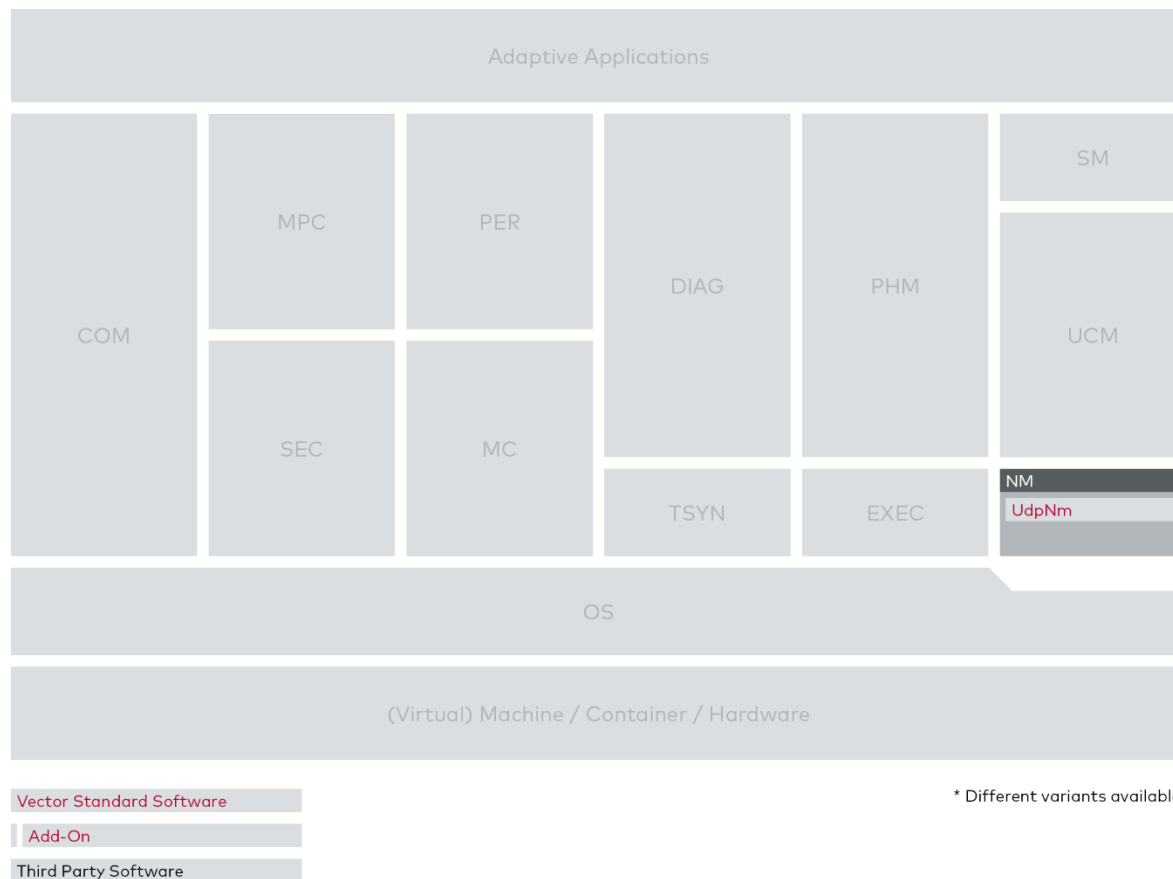


FIGURE 14: COM cluster

MICROSAR Adaptive Network Management implements the algorithm of the AUTOSAR NM protocol specification and decentrally synchronizes the communication states of all nodes in a network (based on UDP over Ethernet). A user-supplied adaptive application is recommended to interact with the NM, coordinate network requests and releases from other ECU applications, and request networks on the NM interface accordingly.

- > NM Algorithm, Network State Field, and Operational Modes
- > Transmission and Reception of NM PDUs
- > Partial Networking, State Report, and Car Wake-up
- > Communication Control and Active Wake-up Bit

14.1 UDP Network Management Protocol Specification (UdpNm)

Module to perform UDP network management functions.

MORE INFORMATION

Visit our website for:

- > News
- > Products
- > Demo software
- > Support
- > Training and workshops
- > Contact addresses

vector.com