

Curriculum: AUTOSAR Basics

PUBLIC

1 Architecture Development for AUTOSAR Systems

1.1 Terms and Principles

Software architecture, functional software architecture, technical software architecture, software development, distributed systems. AUTOSAR Methodology in context of iterative incremental software architecture development.

1.2 Learning Goals

LG 1-1 History and constraints of AUTOSAR

Participants know:

- the historical background of AUTOSAR
- Historical milestones
- The differentiation between AUTOSAR Classic and Adaptive
- the organizational constraints:
- Typically each ECU is developed by a different ECU manufacturer
- It is the task of the OEM to design the vehicle communication and provide each ECU manufacturer with the communication aspects required for that ECU
- the technological constraints:
- Inherent distributed systems design (multiple ECU's per vehicle)
- Support for Inter- and Intra-ECU communication

LG 1-2 Systematic approach to AUTOSAR software development with AUTOSAR Development Methodology

Participants understand how basic design approaches of software architectures relate to AUTOSAR Top-down design approach.

Participants understand the AUTOSAR system term:

- The term AUTOSAR system is defined.
- AUTOSAR systems are distributed by nature.
- The AUTOSAR system is separated from the terms used in ISO 26262 and ASPICE.

Participants understand the meaning of methodology and the AUTOSAR methodology in particular. They understand the motivation for this methodology. They have an overview of the steps and work products.

Participants understand what parts of the methodology happen on AUTOSAR system level and which parts are executed for each ECU (creation of ECU extract). They understand the impact of components (AUTOSAR SWCs and CDDs) being mapped to different ECUs on the communication (bus communication can be introduced). Participants understand how the ECU image is created from the different work products of the methodology.

Participants understand that depending on the project circumstances different architectural decisions are made at a different point of time in the development process.

For example:

- Decomposition of the function block into technical blocks (e.g., AUTOSAR SWCs) and at another point of time the concrete deployment is decided.

LG 1-3 Modeling and analysis of functional software Architectures

Participants understand the motivation to introduce the functional architecture.

Participants can apply a systematic approach to modeling and analysis of functional architectures:

- Model the software context with external interfaces, physical entities and the logical events that occur between the software and its environment.
- Hierarchically decompose the software from a purely functional perspective, identifying function blocks and their interactions via events or information flows. In addition, the logical operations are described which define the functions that are used to fulfill the responsibilities of the logical blocks.
- Define event chains based on the functional architecture. An event chain traces the end-to-end behavior from an external triggering event to the desired AUTOSAR system reaction through the involved function blocks, events, and information flows.

- Introduction of the VFB (Virtual Function Bus) as a communication concept / middleware for modeling the logical interaction of components in distributed systems in a technology-neutral way.

Participants understand that a functional architecture enables reasoning about software quality characteristics according to the ISO 25010 at an abstract level, e.g., regarding time behavior, dependability, or flexibility.

For example:

- Decide on the communication paradigm (e.g., time triggered vs. event triggered).
- Decide on the logical partitions which means to separate the function blocks related to safety and QM.
- Flexibility is supported through software configuration mechanism and AUTOSAR SWCs can be seen as a plugin to the AUTOSAR framework.

LG 1-4 Modeling and analysis of technical software architecture

Participants understand the motivation for AUTOSAR Layered Software architecture:

- Abstraction of details
- Separation of concerns (Signal Interface, Safety, Security)
- Support for various communication technologies (Can, Lin, Flex Ray, Ethernet)

Participants understanding the AUTOSAR Layered Software Architecture:

- Support for various communication technologies (Can, Lin, Flex Ray, Ethernet)
- Driver and Interface define the communication technology implementation.
- Driver handle communication frames.
- Interface translates communication technologies specific format to harmonized PDUs.
- PduR as the central routing module.
- Com module as the signal interface.
- RTE as the Application Software Interface.
- Transformer as Serialization technology.

Participants understand the systematic approach to modeling and analysis of technical software architecture:

- Analyze the impact factors, like quality requirements, organizational constraints, technological constraints, as defined in the CPSA Foundation Level curriculum. For the technical software architecture of an AUTOSAR system, impact factors are often related to dependability and safety requirements, performance and real-time requirements, and flexibility requirements to support different product variants and product lines. Examples for further considerations are development schedules, organizational attitudes, and software development process.

- Based on the impact factors, identify architectural issues, and develop solution strategies for topics such as error handling, resource efficiency, real-time architectural design, and variability.
- Evaluate alternative technical software architectures and select a technical software architecture for further refinement.
- Hierarchically decompose the software into technical building blocks (e.g., AUTOSAR SWCs, CDDs, ECU Abstraction ...), and establish a mapping to the elements of the functional architecture, if applicable.
- Map event chains to the technical architecture, if applicable: For example, allocate a logical operation to an AUTOSAR SWC runnable to determine a measurement to a physical sensor, or allocate an information flow to an AUTOSAR sender-receiver-interface and map it to an AUTOSAR system signal for bus communication (cf. LG 2-3).
- Evaluate the defined technical software architecture regarding the impact factors.
- AUTOSAR propagates well defined interfaces (C/S, S/R IRV's, data access etc.).

Participants know examples of how technical software architectures can be modeled, for example with AUTOSAR authoring tools (AATs).

Modelling Elements are, for example:

- Software component design
- Software component types
- Application SW Component Type
- Complex Device Driver SW Component Type
- Sensor Actuator SW Component
- Parameter SW Component Type
- NV Block SW Component Type
- Service Proxy SW Component Type
- Service SW Component Type
- ECU Abstraction SW Component Type
- Software component instantiation
- Composition
- Interface Types
- Sender/Receiver interfaces
 - Sender/Receiver communication default as push principle
- C/S interfaces
 - Client/Server communication default as pull principle
 - Synchronous and asynchronous concept
- Mode Switch interfaces
 - Mode Declaration Group
 - Trigger communication as means to transfer actions without data

- Mode communication as means to transfer mode information
- Parameter interfaces
- NV Data Interface
- Trigger Interface
- AUTOSAR data types and abstractions
- Primitive Types and complex types
- Compu Methods
- Data Constraints
- Constants
- Unit and Physical Dimension
- Data Type Mapping Set
- Internal Behavior
- Runnables
- Data Access
- Server Call Points (Synchronous and Asynchronous)
- Events triggering Runnables
 - Events:
 - Timing Event
 - Data Received Event
 - Operation Invoked Event
 - Init Event
 - ...
- Event to task mapping (concept)
- Exclusive Areas
- Interrunnable Variables
- Parameters
- Data Type Mapping Refs
- Per Instance Memory
- Service Needs
- Port instantiation
- Required-Port
- Provide-Port
- Top Level Composition

Participants understand that the design of a technical software architecture is an iterative process that requires feedback from software development and hardware development, AUTOSAR system designer.

In addition, participants understand the close interplay between AUTOSAR ASW and AUTOSAR BSW development.

Participants understand that AUTOSAR elements are part of the technical architecture.

Participants know typical decisions that need to be made in the design of the technical software architecture:

- How a function block is implemented (decomposition), for example in one or multiple AUTOSAR SWCs (AUTOSAR Software-level).
- Deciding the assignment of SWCs to control units (deployment) (AUTOSAR System-level).
- Decide on the communication paradigm (e.g., time triggered vs. event triggered). (AUTOSAR System-level)

Participants understand the challenges of distributed systems regarding latency, throughput, lack of a global clock, and the impact on dependability.

2 Basic Software Concepts of AUTOSAR

2.1 Terms and Principles

Tasks, Events, Alarms, Interrupts, IOCs, Resources, Event, Preemptive, Context, Scheduling, Priority, Virtual Functional Bus (VFB), Signal Based Communication, Service Oriented Communication, Sender-Receiver Communication, Client/Server Communication, Trigger Communication, Mode Communication.

2.2 Learning Goals

LG 2-1 Basic Concept of Operating System

The participant understands the fundamental concepts of the operating system (OS) and their practical applications.

This includes distinguishing between basic and extended tasks and their states, as well as the various timer types like HRT and PIT. The significance and use of alarms and interrupts, including the differences between CAT1 and CAT2 interrupts, are also covered. Additionally, the management and function of OS resources, as well as the concepts of IOC and OS applications, are thoroughly explained. Finally, the chapter addresses system scalability in different classes and protection mechanisms such as memory and timing protection.

- System Scalability
- Scalability Class 1
- Scalability Class 2
- Scalability Class 3

- Scalability Class 4
- Protection
- Timing Protection
- Memory Protection
- Tasks
 - What a basic task is and the different states
 - What an extended task is and the different states
 - Differences between Basic and Extended Task
 - What a Context is
- Scheduling Policy and Priority
- Timer/Alarms
 - The two types of timers.
 - What an HRT (High Resolution Timer) is
 - What a PIT (Periodic Interrupt Timer) is
 - The behavior of an alarm
 - The use of alarms.
- Interrupt
 - What an interrupt is
 - The difference between CAT1 and CAT2 interrupt
 - Impact of the Interrupt category on the ECU software
- OS Resources
 - What OS Resources are for
 - How OS Resources works
 - Priority Ceiling Protocol
- OS Application
 - What an OS Application is
 - How they are used
 - Which classes of OS Application AUTOSAR has (Trusted and Non-Trusted)
- IOC
 - What IOC (Inter Os Application Communication) is for

- How IOC (Inter Os Application Communication) works
- Spinlocks

LG 2-2 Basic Concept of IO

The participant is able to identify and understand the relevant AUTOSAR driver modules that are responsible for handling input and output operations within an automotive software system.

Additionally, they can describe in detail the concept of hardware abstraction, which involves decoupling software from specific hardware implementations to enhance portability and reusability.

Furthermore, the participant is capable of explaining the functionality and roles of the PORT driver, which is responsible for configuring microcontroller pins, and the DIO (Digital Input/Output) driver, which manages digital signal processing and communication between software and hardware components.

- recognize AUTOSAR driver modules which are relevant for I/O.
- describe the concept of abstraction of HW components.
- describe the functionality of PORT and DIO Driver.
- Complex Device Driver (CDD)

LG 2-3 Basic Concept of Communication

The participant understands the fundamental concepts of the Virtual Function Bus (VFB) and the top-down design approaches for AUTOSAR systems based on software components.

Various communication patterns, including signal-based and service-oriented communication, as well as communication modes and data flows within the AUTOSAR Layered Software Architecture, are explained.

The focus is placed on the abstraction of communication details, support for different network technologies, and the role of the various communication modules.

- AUTOSAR Communication State Management
- Need for Network Management: Put networks and ECUs to sleep
- Need for Network State Management: Requested communication state might differ from available communication state
- AUTOSAR Communication Data Path
- Data flow in the AUTOSAR Layered Software Architecture: PDUs
- Configuration of the PDU handling in the AUTOSAR modules
- Communication Modes

- Transmission modes of an I-PDU
- Transfer properties for signals
- Signal filters (Never, always ...)
- The selection of the Transmission Mode for one specific I-PDU works
- Signal Flow
- Signal Invalidation
- Transmission of an invalidated Signal
- Reception of an invalidated Signal
- Handling of I-PDUs
- How an I-PDU Group is defined
- How I-PDU Groups start/stop
- Signal Indication (Unpacking of I-PDUs)
- Deadline Monitoring
- The reception of Deadline Monitoring
- The transmission Deadline Monitoring
- Notification handling
- Notification handling
- Update-Bits
- How the Update-Bit works on Sender Side
- The participant understands how the Update-Bit works on Receiver Side
- Communication Stack
- Communication related modules
- Mechanisms of RTE sender ports (Data mapping)
- The purpose of the COM module
- The purpose of the PDUR module
- The purpose of the bus specific IF module
- The purpose of the bus driver module

LG 2-4 Basic Concept of Mode Management

The participant has a comprehensive understanding of the fundamental operational states of an ECU (Electronic Control Unit) and the specific roles and interactions of the Basic Software

Modules (BSW) such as BswM (Basic Software Mode Manager), EcuM (ECU State Manager), and ComM (Communication Manager).

This includes detailed knowledge of the startup and wakeup procedures, the transitions between different phases and the management of network states.

Furthermore, the participant is familiar with the Mode Manager concept, which orchestrates the control and coordination of functional modes, and the service mapping strategy that ensures seamless communication and functional integration between Basic Software (BSW) modules and Software Components (SWC) within the AUTOSAR architecture.

- EcuM
- The two types of EcuM configurations
- The purpose of the EcuM
- Fundamental states of an ECU
- BswM
- The purpose of the BswM
- The context of mode requests
- Mode Notification Ports
- ComM and NM
- The purpose of the ComM
- States of the ComM
- The purpose of the NM
- The different Network States
- How the ComM state-machine works
- The purpose of the BusSM
- The sequence of function calls during startup
- The different ECU Sleep Phases

LG 2-5 Basic Concept of Memory

Learners will gain a solid understanding of memory handling and the role of the Non-Volatile Memory Manager (NvM) in AUTOSAR systems.

They will learn about NvM's functionality, features, and its importance during system startup and shutdown.

They will understand the configuration parameters of memory blocks, the connection between Nv and Fee blocks, and the differences between RAM, ROM, and various Non-Volatile (NV)

blocks.

Learners will also explore memory abstraction and modules like EA and FEE, analyzing the advantages and limitations of EEPROM and Flash EEPROM technologies, as well as the concept of memory chunks.

Additionally, learners will recognize the service needs of memory-related components and explain the purpose of NvBlockSWC, gaining a clear understanding of the interdependencies within memory management.

- NvM
- The role of the NvM
- NvM specific functions and features
- The Service Port categories of the NvM
- The role of the NvM during shutdown and startup
- Configuration parameters for Memory blocks
- Relation between the Nv Block and the Fee Block
- RAM and ROM blocks
- Types of NV blocks
- Memory abstraction & handling
- Memory related modules
- The role of chunks
- The purpose of EA specific features
- The purpose of FEE specific features
- Pros and cons of EEPROM and Flash EEPROM
- The role of PIM
- The configuration of Service Needs
- The purpose of NvBlockSWC

LG 2-6 Basic Concept of Diagnostics

The participant understands the fundamental diagnostic modules and their functions, including UDS, OBD, and the related transport protocols.

The processes of diagnostic requests and responses, as well as the roles of DCM, DEM, FIM, diagnostic monitors, and mechanisms such as debouncing, diagnostic sessions, events, operation cycles, and DTCs, are explained.

- Diagnostics
- Reasons for employing a transport protocol in diagnostics
- Diagnostic modules

- The role of UDS in diagnostics
- The role of OBD in diagnostics
- The different Diagnostic Sessions
- The two types of diagnostic requests
- DCM
- The role of the Diagnostic Communication Manager (DCM)
- DEM
- The role of Diagnostic Event Manager (DEM)
- Debouncing algorithms
- The content of a DTC
- The purpose of Events
- The purpose of Debouncing
- Debouncing Algorithms
- The purpose of Operation Cycles
- The purpose of a Diagnostic Monitor
- FIM
- The functionality of Function Inhibition manager (FIM)