AUTOSAR Abstraction Layer

Project Training –Automotive Overview

26/07/2024
Areeb Hammad N
Anuratha N
Atraindra Gupta
Anji Babu

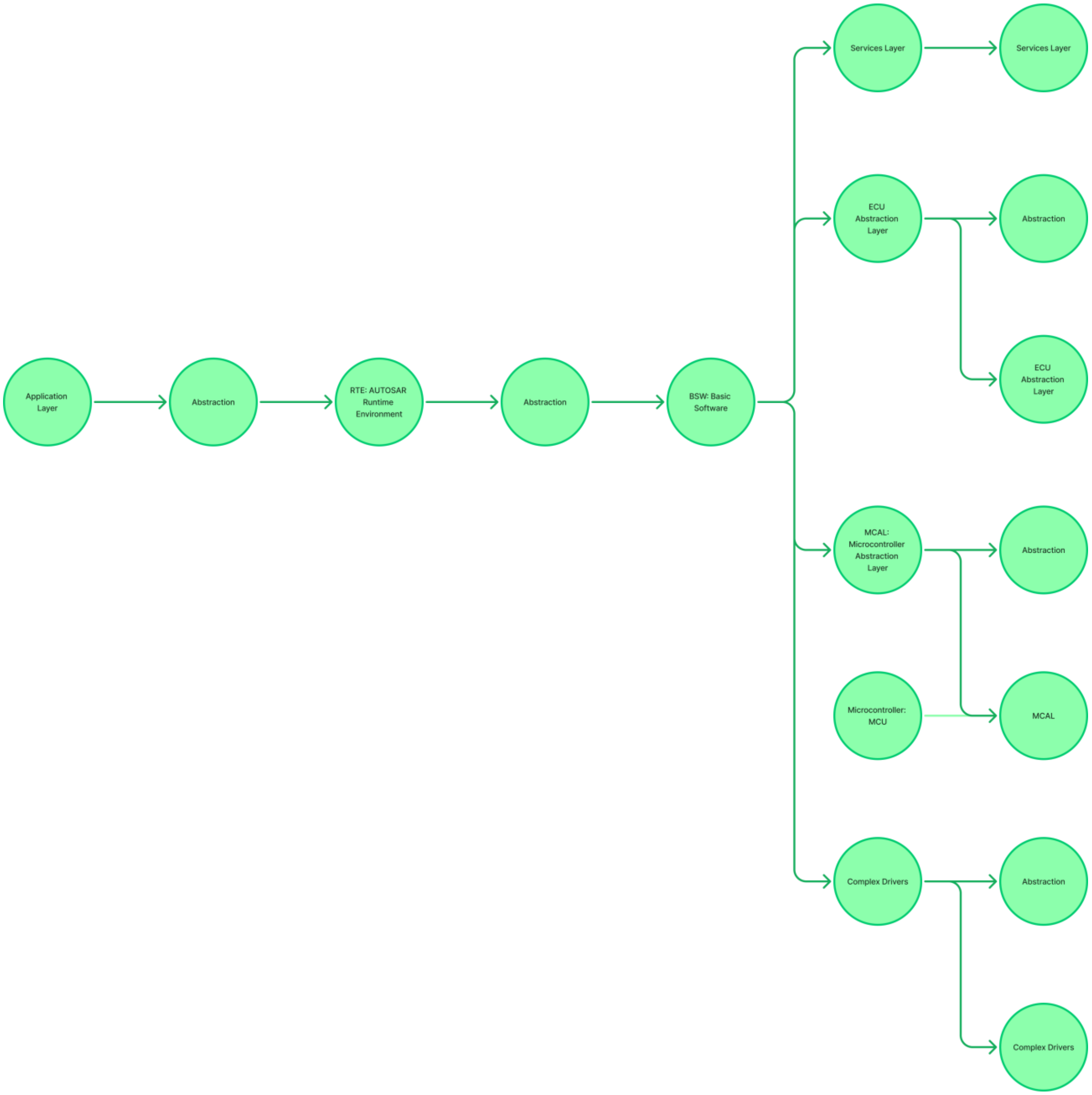
Version: 1.0

Created: 26/07/2024

Last Updated: 26/07/2024

Status: DRAFT (The status would change to finalized post the BA, PM and dev team review and sign off)

Task2: Why we need the abstraction layer for AUTOSAR integrating software and hardware components



Application Layer:

- **Description**: This is the topmost layer where the application software components reside. These components perform specific functions required by the vehicle's operations, such as controlling the engine, managing the infotainment system, etc.
- **Abstraction**: This layer is hardware-independent, meaning applications do not need to know the specifics of the underlying hardware.

AUTOSAR Runtime Environment (RTE):

- **Description**: Acts as a middleware that facilitates communication between the application layer and the lower layers. It enables the interaction of software components within the application layer as well as their communication with the Basic Software (BSW) modules.
- **Abstraction**: The RTE abstracts the communication mechanisms, so application components can exchange data without knowing the details of the data transmission.

Basic Software (BSW):

- **Description**: This layer is further divided into several sub-layers, each responsible for different aspects of hardware interaction and services.

a. *Services Layer:*

- **Description**: Provides general services such as system management, diagnostics, communication, and memory services that are independent of the hardware specifics.
- **Abstraction**: It is hardware-independent, meaning it can operate regardless of the specific microcontroller or ECU hardware used.

b. *ECU Abstraction Layer:*

- **Description**: Provides an interface to the hardware by abstracting the details of the ECU hardware. This includes handling peripherals like I/O ports, ADCs, and timers.
- **Abstraction**: It abstracts the ECU hardware specifics and offers a standardized interface to the higher layers.

c. ***Microcontroller Abstraction Layer (MCAL):***

- **Description**: This layer abstracts the microcontroller-specific features, providing a uniform interface to the higher layers for accessing microcontroller resources.
- **Abstraction**: It is dependent on the specific microcontroller used but hides these details from the upper layers, facilitating portability.

d***. Complex Drivers:***

- **Description**: These drivers handle complex hardware interactions that are not covered by the standard abstraction layers. They interact directly with the ECU hardware.
- **Abstraction**: While they are independent of the microcontroller specifics, they are dependent on the ECU hardware.

Microcontroller (MCU):

- **Description**: The physical hardware component, such as the microcontroller, that executes the software.
- **Abstraction**: The hardware specifics are abstracted by the Microcontroller Abstraction Layer (MCAL).

How Abstraction Works

- **Services Layer**: Provides hardware-independent services. This layer interacts with the application layer and RTE, ensuring that application software does not need to concern itself with hardware details.

- **ECU Abstraction Layer**: Sits between the microcontroller abstraction layer and the services layer. It abstracts the specifics of the ECU hardware, allowing the services layer to use a standardized interface to interact with different types of hardware.
- **Microcontroller Abstraction Layer (MCAL)**: Sits directly above the microcontroller. It abstracts the microcontroller specifics, offering a consistent interface to the ECU abstraction layer. This means that changes in the microcontroller type do not affect the higher layers.
- **Complex Drivers**: These are specialized drivers that provide functionality not covered by the standard abstraction layers. They interact directly with the ECU hardware but are designed to be independent of the microcontroller specifics, thus providing flexibility.