Session 19

AUTOSAR interfaces, Application layer features in AUTOSARAn

The AUTOSAR Application Layer is the top layer in the AUTOSAR architecture and is where the core vehicle functionalities reside. It's composed of Software Components (SWCs) that implement specific tasks like engine control, transmission control, or infotainment features. These SWCs communicate with each other and with lower layers (like the Basic Software) through standardized interfaces and the Runtime Environment (RTE).

Key Features of the AUTOSAR Application Layer:

- **Software Components (SWCs):**

  These are the building blocks of the Application Layer, encapsulating specific functionalities.

- **Standardized Interfaces:**

  AUTOSAR defines standardized interfaces for communication between SWCs and with other layers, ensuring modularity, portability, and reusability.

- **Hardware Abstraction:**

  SWCs interact with hardware indirectly through the RTE and lower layers, promoting hardware independence.

- **Modular Design:**

  The layer's modular structure, with SWCs representing distinct functions, allows for easier development, maintenance, and scalability.

- **Integration of Vehicle Functions:**
  The Application Layer brings together various vehicle functions, like engine control, chassis control, and infotainment, into a cohesive system.

How it Works:

1. **1. Application Logic:**

   SWCs contain the specific code for implementing the desired vehicle functionalities.

2. **2. Communication:**

SWCs communicate with each other and with the RTE through well-defined ports and interfaces.

3. **3. RTE Abstraction:**

   The RTE acts as a middleware, handling communication details and ensuring seamless interaction between SWCs, regardless of their location (same or different ECUs).

4. **4. BSW Interaction:**
   The RTE also interfaces with the Basic Software (BSW) layer, which provides services like communication, memory management, and hardware access.

In essence, the Application Layer provides a standardized, modular, and hardware-independent environment for developing and integrating the diverse functionalities of a modern vehicle.

Sample Pseudocode

1. Interface Definitions (simplified)

```
c
CopyEdit
// Sender-Receiver port: SmokeSensorValue (input)
uint16_t SmokeSensorValue; // Input data

// Client-Server port: BuzzerControl
void Buzzer_On(void);
void Buzzer_Off(void);
```

---

2. SWC Runnable Example

```
c
CopyEdit
#include <stdint.h>

#define SMOKE_THRESHOLD 300

// Input data (mapped from RTE read port)
extern uint16_t Rte_Read_SmokeSensorValue(void);

// Client-Server operation (mapped from RTE client
port)
```

```
extern void Rte_Call_BuzzerControl_On(void);
extern void Rte_Call_BuzzerControl_Off(void);

// Runnable entity executed periodically
void FireAlarmRunnable(void) {
    uint16_t smokeLevel =
Rte_Read_SmokeSensorValue();

    if (smokeLevel > SMOKE_THRESHOLD) {
        Rte_Call_BuzzerControl_On();
    } else {
        Rte_Call_BuzzerControl_Off();
    }
}
```