International Journal of Engineering

Journal Homepage: www.ije.ir

Framework of Electric Vehicle Fault Diagnosis System Based on Diagnostic Communication

X. Yang*

Automobile & Rail Transportation School, Tianjin Sino-German University of Applied Sciences, Tianjin, China

PAPER INFO

Paper history:
Received 02 Novembr 2023
Received in revised 06 December 2023
Accepted 24 December 2023

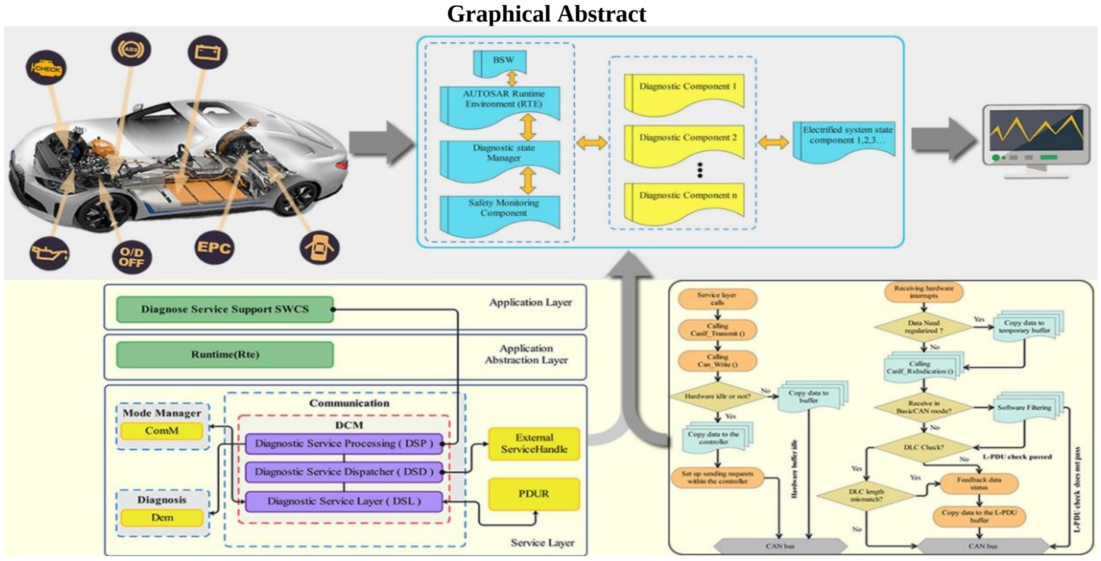
Keywords:
Electric Vehicle
AUTOSAR
Fault Diagnosis
Fusion

ABSTRACT

With the escalating integration of Electronic Control Units (ECUs) in contemporary vehicles, the intricacy of vehicle networks is incessantly advancing. Diagnostic communication, as a pivotal facet within these networks, grapples with protracted development cycles and heightened intricacies. In a bid to augment software reusability and portability, this study meticulously scrutinized pertinent research and proffered an electric vehicle fault diagnosis system predicated on the Controller Area Network (CAN) bus, leveraging the diagnostic communication architecture advocated by the AUTOSAR standard. The integration of AUTOSAR seeks to pioneer an innovative software development paradigm for automotive fault diagnosis systems, thereby remedying extant limitations. The communication and diagnostic module of this study were instantiated using AUTOSAR, thereby obviating the necessity for developers to immerse themselves in hardware intricacies and communication implementations. This allows developers to focalize their efforts on crafting software features for fault diagnosis. Empirical results illustrate that the single-core CPU utilization rate of the proposed method in this article stands at 40.68%, with a fault detection time of 0.0217. The success rate of fault detection is 98.70%, indicating an increase of 12.97% and 8.98% when compared to the CAN bus and structural analysis methods, respectively. Testing indicators are significantly mitigated, yielding more precise fault detection outcomes. The exploration of this avant-garde software development methodology in automotive electronic products markedly amplifies the efficiency of automotive troubleshooting system software, underscoring its potential for academic contribution and application in real-world scenarios.

doi: *10.5829/ije.2024.37.06c.16*

Graphical Abstract


CHECK
ABS
EPC
O/D OFF
BSW
AUTOSAR Runtime Environment (RTE)
Diagnostic state Manager
Safety Monitoring Component
Diagnostic Component 1
Diagnostic Component 2
Diagnostic Component n
Electrified system state component 1,2,3…
Diagnose Service Support SWCS
Application Layer
Runtime(Rte)
Application Abstraction Layer
Communication
Mode Manager
ComM
DCM
Diagnostic Service Processing (DSP)
Diagnostic Service Dispatcher (DSD)
Diagnostic Service Layer (DSL)
Diagnosis
Dem
External ServiceHandle
PDUR
Service Layer
Service layer calls
Calling CanIf_Transmit ()
Calling Can_Write ()
Hardware idle or not?
Copy data to buffer
Copy data to the controller
Set up sending requests within the controller
Hardware buffer idle
CAN bus
Receiving hardware interrupts
Data Need regularized ?
Copy data to temporary buffer
Calling CanIf_RxIndication ()
Receive in BasicCAN mode?
Software Filtering
DLC Check?
L-PDU check passed
DLC length mismatch?
Feedback data status
Copy data to the L-PDU buffer
L-PDU check does not pass
Yes
No

*Corresponding Author Email: yangxiaogang@tsguas.edu.cn (X. Yang)

1. INTRODUCTION

Approximately 70% of the time spent in the maintenance process of traditional cars is devoted to fault identification, with the remaining 30% allocated to troubleshooting. In comparison to conventional vehicles, the electrical system of pure electric vehicles is more intricate and electronic in nature. It encompasses several subsystems, such as the vehicle, drive motor, battery management, high-voltage electrical safety, instrument panel control, auxiliary power system, air conditioning, power steering, and electronic brakes. Each subsystem carries out its functions through its dedicated Electronic Control Unit (ECU), and communication between ECUs is facilitated through the Controller Area Network (CAN) bus network, ensuring the coordinated operation of the entire vehicle (1-3).

Gholami and Sanjari (4) designed a real-time fault diagnosis system for pure electric vehicles. This system can promptly identify potential faults and implement appropriate strategies to ensure the safe and reliable operation of vehicles. Bhosale and Mastud (5) developed a fault diagnosis system for pure electric vehicles based on the CAN bus. They completed the design of a fault diagnosis instrument for pure electric vehicles and tested the fault diagnosis system with CANoe software. Ahmadigorji and Mehrasa (6) used the structural analysis method to establish a fault diagnosis system for the power system of pure electric vehicles. Jian et al. (7) established diagnosis rules and constructed the corresponding fault tree based on the study of fault diagnosis technology for pure electric vehicles. Subsequently, a set of expert fault diagnosis system models was designed using WPF software language to enhance fault diagnosis efficiency and compensate for the lack of technical expertise among after-sales personnel. Ochando et al. (8) developed a pure electric vehicle status monitoring and fault diagnosis system based on the onboard CAN network. They utilized LabVIEW and Kvaser USBcan communication card to achieve real-time monitoring of the status and fault information of pure electric vehicles during operation. Faults were analyzed based on monitored status and specific fault phenomena when encountered, leading to effective solutions. Wang (9) studied the fault diagnosis of the distributed control system of electric vehicles based on the CAN bus. This involved discussions on fault diagnosis modes, fault monitoring and diagnosis methods, and the coding method of fault codes (DTC) and fault information for electric vehicles, representing a valuable exploration into the in-depth study of electric vehicle fault diagnosis based on the CAN bus.

As the integration of Electronic Control Units (ECUs) with modern vehicles continues to rise, the complexity of the entire vehicle network is increasing. As a critical function within the onboard network, the development cycle and difficulty of diagnostic communication are escalating. To enhance software reuse and portability, this paper develops an electric vehicle fault diagnosis system based on the CAN bus, incorporating the diagnostic communication architecture recommended by the AUTOSAR standard through an analysis of relevant research.

2. INTRODUCTION TO RELATED THEORIES

2. 1. AUTOSAR Architecture Haur (10) endeavored to implement crucial functions within an automotive electronic software system, with the objective of standardizing functional interfaces. This standardization facilitates the seamless integration and effective reuse of software modules, thereby enhancing the efficiency of software updates and development processes. To achieve this, the software architecture is structured into three layers: the application layer, the run-time environment layer, and the base software layer. This hierarchical and modular approach aligns with contemporary software development and design philosophy (11), as illustrated in Figure 1.

2. 1. 1. Application Layer The application architecture of AUTOSAR comprises interconnected software components (SWCs) linked through a virtual function bus. Each SWC incorporates one or more ports, and these SWCs establish connections through these ports. Within the SWCs, running entities (REs) represent the smallest code fragments, eventually mapped to specific operating system (OS) tasks and scheduled by the OS to execute corresponding functions (12).

To facilitate system integration, AUTOSAR introduces the Virtual Function Bus (VFB) (13). The VFB enables the design of application software without direct dependence on the underlying hardware and communication mechanisms. SWCs communicate through ports, interacting with hardware resources via the VFB. This design choice renders the application layer software implementation independent of the specific hardware, thereby significantly enhancing the portability of the application software.

2. 1. 2. RTE Layer At the core of the AUTOSAR architecture lies the Run-time Environment (RTE), serving as a tangible realization of the Virtual Function Bus (VFB). The RTE plays a pivotal role by mapping Running Entities (REs) within all Software Components (SWCs) on the local Electronic Control Unit (ECU) to tasks in the operating system (OS). It is responsible for establishing communication among these REs. In cases where REs are mapped to different ECUs, the RTE takes on the responsibility of facilitating communication between them.

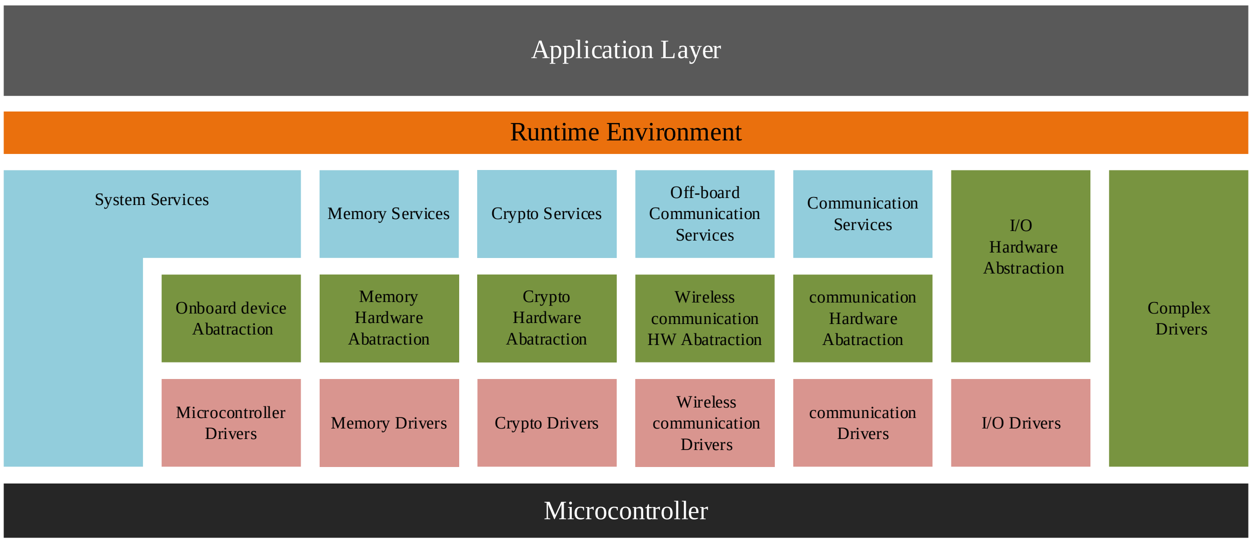


Figure 1. AUTOSAR architecture

Furthermore, the RTE is instrumental in implementing the segregation between application and base software. It provides communication services between SWCs in the application layer and acts as a conduit for communication within single ECU systems or across multi-ECU systems (14). The RTE defines interfaces for data communication between application layer SWCs and the underlying software modules. This includes standardizing interfaces for input/output (I/O), storage, and other fundamental accesses, thus ensuring the application's independence from underlying hardware characteristics.

2. 1. 3. Base Software Layer The base software layer serves as a crucial foundation, delivering essential services to the application layer's software components. These services encompass a spectrum of functionalities, including underlying hardware drivers, bus and network communication, real-time task scheduling, vehicle troubleshooting, and other foundational services. Comprising approximately 80 base software modules, this layer is organized into the microcontroller abstraction layer, ECU abstraction layer, service layer, and complex driver layer, following a bottom-up hierarchy.

The microcontroller abstraction layer, ECU abstraction layer, service layer, and complex driver layer collectively enable applications to access microcontroller hardware resources directly. This access is facilitated through the complex driver layer, allowing the implementation of intricate sensor and controller operations, such as fuel injection, ignition control, and other specific and complex functions. The complex driver layer is particularly valuable for implementing hardware resources not supported by AUTOSAR or not standardized, while ensuring compliance with real-time requirements for specific operations (15).

2. 2. AUTOSAR Diagnostic Functions The diagnostic-related modules within the AUTOSAR automotive electronics software architecture are depicted in Figure 2.

The Function Inhibition Manager (FIM) module plays a pivotal role in enabling or disabling functional entities within the software component based on event statuses reported by the Diagnostic Event Manager (DEM). The Diagnostic Communication Manager (DCM) and DEM serve as core modules responsible for implementing the diagnostic functions inherent in AUTOSAR. As of the current version, AUTOSAR version 3.1 diagnostics encompass a comprehensive suite of 9 On-Board Diagnostics (OBD) services.

2. 3. Multi-sensor Information Fusion

2. 3. 1. Multi-sensor Information Fusion Concept

Multi-sensor fusion, commonly known as data fusion, involves amalgamating pertinent information gathered by various environmental sensing sensors installed on an innovative electric vehicle. The synthesis of information detected by multiple sensors, when combined and complemented, addresses the limitations of individual sensors under external influences. This collaborative approach mitigates the risk of decision errors and enhances overall recognition capabilities (16–19).

The integration of information data and the location of fusion delineate three distinct levels from low to high abstraction: the data layer (sensor-level data fusion), feature layer (central-level data fusion), and decision layer (hybrid data fusion).

(1) Data Layer Fusion

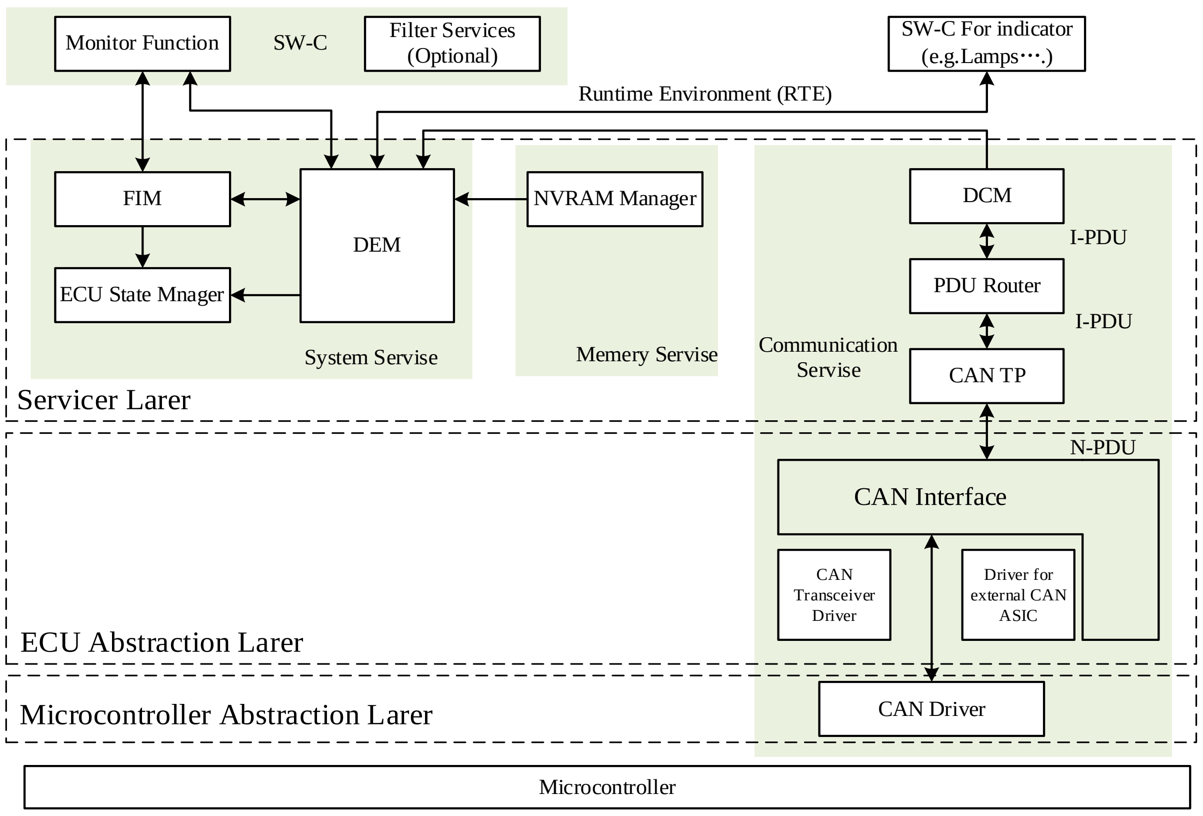


Figure 2. AUTOSAR Diagnostic functions

At the lowest fusion level, known as data layer fusion, raw data is directly transmitted to the fusion center without undergoing any preliminary processing or analysis. The process of data layer fusion is illustrated in Figure 3.

This fusion level, while minimizing original data loss, is characterized by extensive redundant data processing, resulting in compromised real-time robustness and interference resistance.

(2) Feature Layer Fusion

Feature layer fusion involves extracting target features, such as boundary, distance, velocity, size, orientation, and angle, through simple filtering. Subsequently, the collected data undergoes classification and analysis to eliminate invalid information before the actual data fusion process. The feature layer fusion process is depicted in Figure 4.

This fusion level necessitates preliminary data processing, involving the compression of raw data information to ensure effective real-time processing. However, this approach may introduce the potential loss of critical raw data, leading to biases in the fusion results (20).

(3) Decision-Level Integration

Decision-level fusion entails the amalgamation of local decisions made by sensors through a mid-level

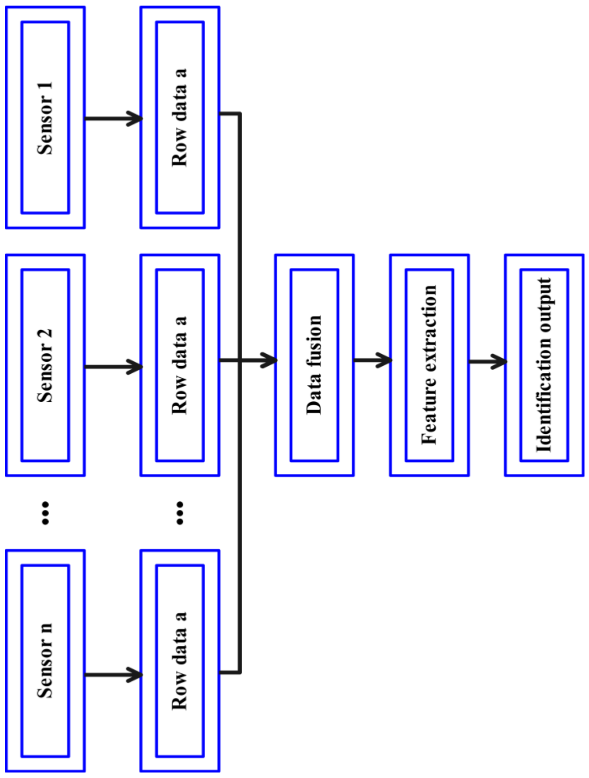


Figure 3. Data layer fusion process

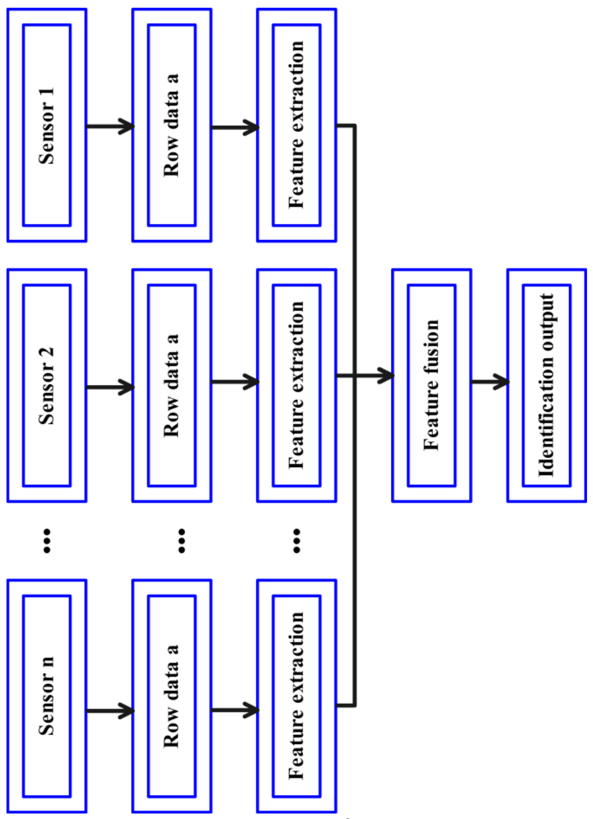


Figure 4. Feature layer fusion process

fusion processor. This occurs subsequent to the pre-processing of collected data for tasks such as classification, identification, and decision-making. The decision layer fusion process is illustrated in Figure 5. Since the fusion process is not directly involved in system decision-making, it guarantees flexibility in fusion and robust anti-interference capabilities. Even if certain sensor functions experience failure, it does not result in significant errors in the fusion results. However, the trade-off is that the pre-processing of data becomes more intricate, thereby increasing the processing difficulty.

2. 3. 2. Methods for Multi-sensor Information Fusion The algorithm for information fusion in electric vehicles, leveraging multiple sensors, incorporates a range of techniques, including weighted average, Kalman filter, Bayesian estimation, D-S evidence theory, fuzzy logic inference, and artificial neural network (21-23). In this paper, the approach employed is the weighted average method.

Let the target data acquired by multiple sensors be denoted as $a_1, a_2 \dots, a_n$, with variances $\sigma_1^2, \quad \sigma_2^2 \dots, \sigma_n^2$, t and the corresponding weights of each sensor as $l_1, l_2 \dots, l_n$. After fusion, the resulting state data is:

$$\tilde{x} = l_1 a_1 + l_2 a_2 + \cdots + l_n a_n = \sum_{n}^{i=1} l_i a_i \tag{1}$$

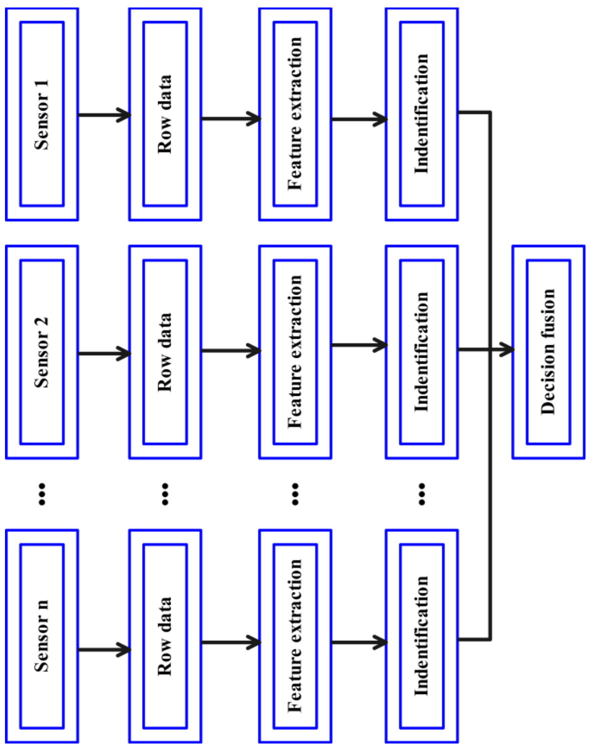


Figure 5. Decision-level integration process

The weighting conditions are defined as follows:

$$\sum_{n}^{i=1} l_i = 1 \tag{2}$$

If each sensor weight is equally distributed, with equal weights denoted as $l = \frac{1}{n}$, then the fused data can be expressed as:

$$\tilde{x} = \sum_{n}^{i=1} l_i a_i = \frac{1}{n} \sum_{n}^{i=1} a_i \tag{3}$$

The total variance after fusion is given by:

$$\sigma^2 = E[(a - \tilde{x})^2] = E\left[\sum_{n}^{i=1} l_i (a - a_i)\right]^2 \tag{4}$$

$$E[(a - a_i)(a - a_j)] = 0(i, j = 1,2,3, \dots, n, i \neq j) \tag{5}$$

The total variance of the weighted average fusion algorithm is calculated as follows:

$$\sigma^2 = \frac{\sum_{n}^{i=1} \sigma_i^2}{n^2} \tag{6}$$

3. DESIGN AND IMPLEMENTATION OF COMMUNICATION DIAGNOSIS MODULE

3. 1. Design for Information Fusion Information fusion technology integrates processed multi-sensor information to delineate specific characteristics of the external environment or the object under observation. In modern society, sensors play a fundamental role, serving as essential tools for monitoring the surrounding environment. They provide a tangible representation of the world to human perception and contribute significantly to technological progress. As depicted in Figure 6, the information fusion process in this paper unfolds in four distinct steps:

(1) Acquisition of Experimental Data: Data is collected in various scenarios, and the raw sensor data obtained is segmented into samples and labeled. These data serve as the foundation for the information fusion process.

(2) Extraction of Features: Feature extraction is conducted separately on the data, yielding features that constitute the feature layer for information fusion.

(3) Training the Respective Recognition Models: The features from the feature layer undergo training using various machine learning algorithms, resulting in the creation of recognition models and their respective decision results.

(4) Decision Layer Fusion: The decision results obtained in the third step are amalgamated using a designed fusion method, ultimately yielding the final recognition result.

3. 2. Design of AUTOSAR Communication Module

The design and implementation flow of the AUTOSAR communication module is depicted in Figure 7.

Firstly, through a comprehensive study of the AUTOSAR communication module standard, the entire file structure of the communication module is designed to capture the overarching design process from a macro perspective.

Secondly, the module undergoes configuration based on the AUTOSAR methodology. The configuration set is acquired by visually representing the module configuration using the self-developed ECU configuration tool, ReDe (24, 25).

Next, the data structure and standard function interfaces of the communication driver and interface layers are implemented in accordance with the specification.

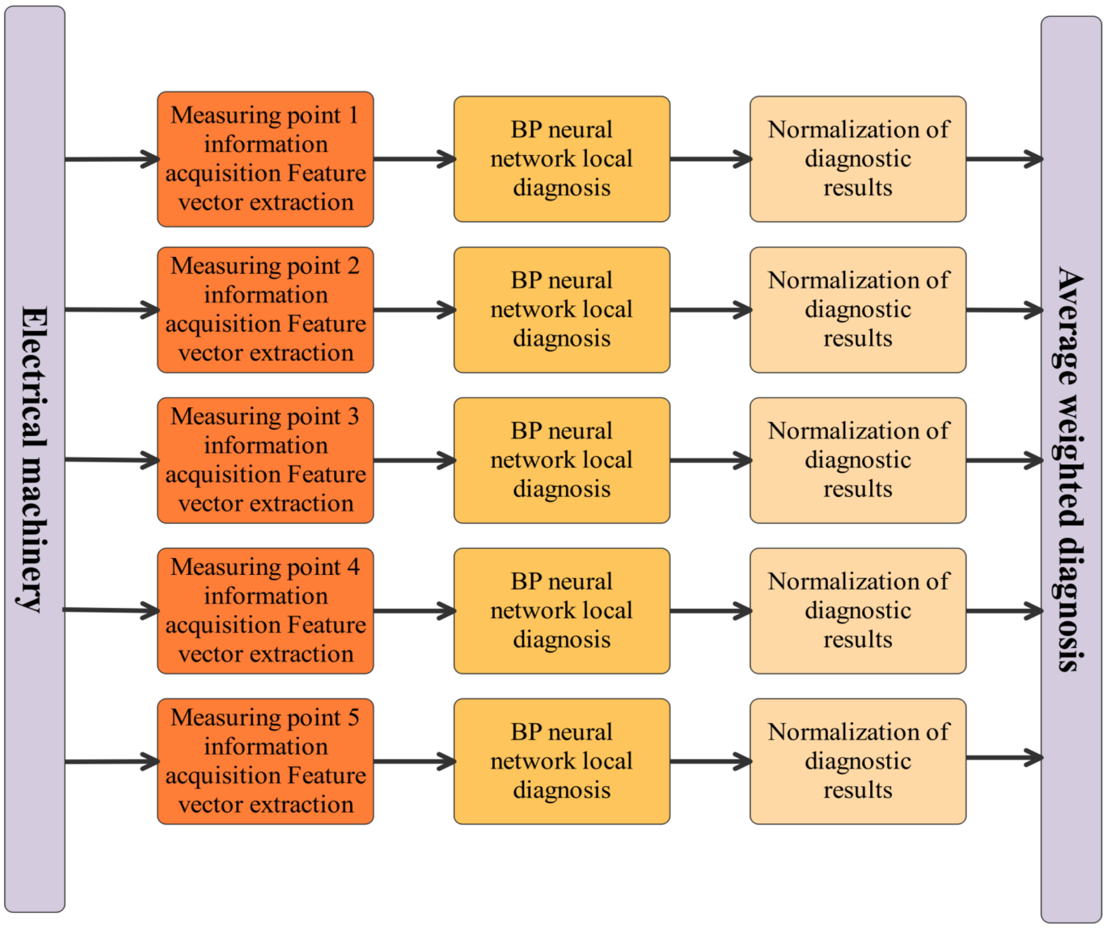

Electrical machinery
Measuring point 1 information acquisition Feature vector extraction
Measuring point 2 information acquisition Feature vector extraction
Measuring point 3 information acquisition Feature vector extraction
Measuring point 4 information acquisition Feature vector extraction
Measuring point 5 information acquisition Feature vector extraction
BP neural network local diagnosis
Normalization of diagnostic results
Average weighted diagnosis

Figure 6. Information Fusion

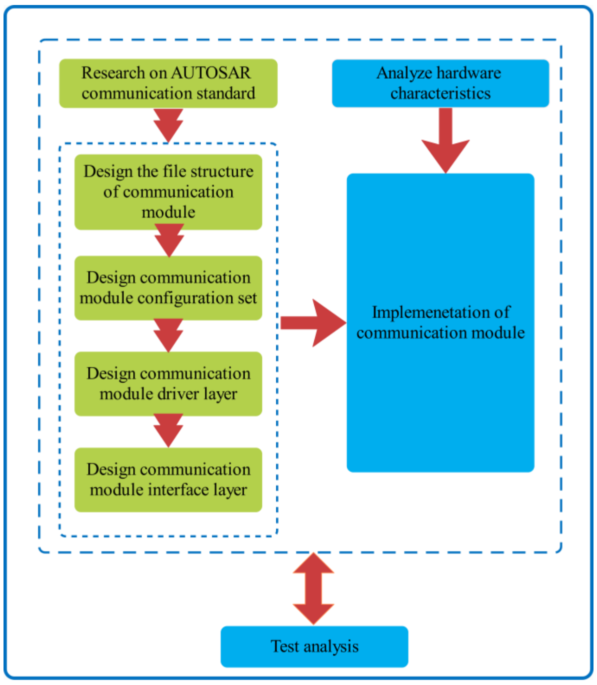

Research on AUTOSAR communication standard
Analyze hardware characteristics
Design the file structure of communication module
Design communication module configuration set
Design communication module driver layer
Design communication module interface layer
Implemenetation of communication module
Test analysis

Figure 7. Design of AUTOSAR communication module

Finally, the complete AUTOSAR communication module is implemented in conjunction with the hardware ECU characteristics. To ensure the reliability and reusability of the module, thorough testing and analysis are conducted post-implementation to identify and rectify any errors that may have occurred throughout the entire design and implementation process.

3. 2. 1. AUTOSAR Communication Module File Structure Given the extensive and intricate nature of the AUTOSAR software architecture, we illustrate the communication system using the CAN bus as an exemplary case. The CAN bus stands out as a highly prevalent Fieldbus in the automotive domain and is widely adopted in current vehicle communication systems. Notably, it serves as the primary Fieldbus for European and American models, which constitute a substantial portion of the vehicle fleet in China. As a result, the CAN bus is ubiquitously employed in virtually all bus technology-equipped models in China (26, 27). In Figure 8, the file structure is presented, meticulously designed in accordance with the AUTOSAR communication specification.

3. 2. 2. Communication Module File Structure The communication driver layer provides a 'Can.h' header file encompassing the definitions of the CAN module API, incorporating variables, global data, and types meant exclusively for internal use by the CAN driver. Simultaneously, the CAN layer furnishes 'Can_Cfg.h' to house configuration parameter information necessary during the pre-compilation phase. The specific services are then implemented in 'Can.c' (28).

Concurrently, the communication interface layer contributes the 'CanIf.h' header file, featuring external variables, global parameters, and services outlined in the specification. These elements are declared in 'CanIf.c' and are restricted to internal usage within the CanIf layer. 'Can_GeneralTypes.h' defines the general data structures

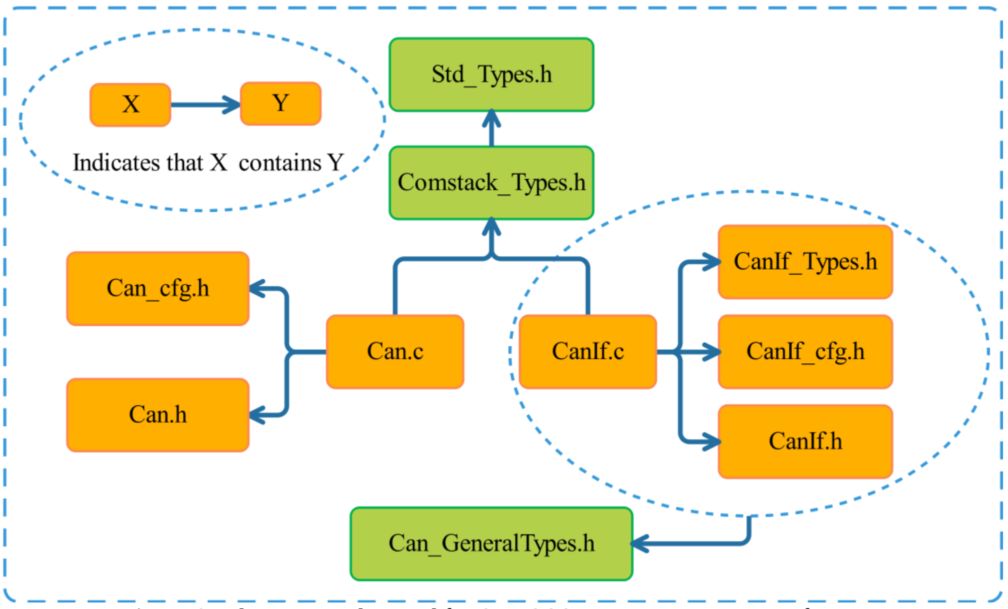


Figure 8. File structure designed for AUTOSAR communication specifications

of the Can driver layer, utilized by the CanIf layer (29, 30). Additionally, 'ComStack_Types.h' encapsulates the definitions of communication-related types, while 'std_Types.h' contains standard type definitions for AUTOSAR.

Upon the incorporation of a communication service module, the headers of each service are added to both the driver and interface layers. The file structure of the communication module is visually represented in Figure 9.

The data sending and receiving process is illustrated in Figure 10. During the transmission, the interface layer assumes the responsibility of assembling data from the upper layers into CAN protocol layer data units, adhering to the CAN specification format. It then invokes 'CanIf_Transmit()' and transfers this data frame to the driver layer. Subsequently, the driver layer employs 'Can_Write()' to initiate the transfer request from the controller. In cases where no hardware object is available, the request is buffered and transmitted once the hardware becomes available. Upon successful transmission, a transmission success confirmation is dispatched to the upper layer module as a callback function, signifying a successful transmission action when received by the sender.

On the reception end, the driver layer reads data from the bus through polling or interrupt mechanisms. Following data regularization, it invokes 'CanIf_RxIndication()' to signal the arrival of the data to the interface layer. The interface layer, upon receiving the CAN data frame from the driver layer, validates and filters the Data Length Code (DLC). After extracting pertinent information, the interface layer communicates the reception event to the corresponding module in the upper communication service layer via 'user_RxIndication()'. If an error is detected during reception, the corresponding processing function is invoked. Additionally, the indication of the data arrival to the upper layer is halted.

3. 3. Design of AUTOSAR diagnostic module The Diagnostic Event Manager (DEM) module, in collaboration with the Software Component (SWC), undertakes the diagnosis of an event within the AUTOSAR system. Upon a change in the event status, the DEM is responsible for notifying the relevant SWC indicator module and various software modules. This notification allows for the display or handling of the detected fault. Additionally, the DEM enables other modules to query and modify the event's status at any given time.

Within the DEM, a counter records the judgment result, with a minimum value of -128 and a maximum value of 127. Upon receiving a message marked as 'PREPASSED,' the counter is decremented by one step. When the counter reaches a predefined threshold value, the event is deemed to be a fault. Following the diagnosis of a fault, the DEM generates a Diagnostic Trouble Code (DTC) based on the collected information and relevant criteria. This DTC provides valuable information about the detected fault.

The AUTOSAR diagnostic process is visually represented in Figure 11.

After diagnosing a fault, the Diagnostic Event Manager (DEM) calls the relevant Non-Volatile Random Access Memory (NVRAM) interface to store data. Events may involve storing various data types, broadly categorized as FreezeFrame and Extended Data Record.

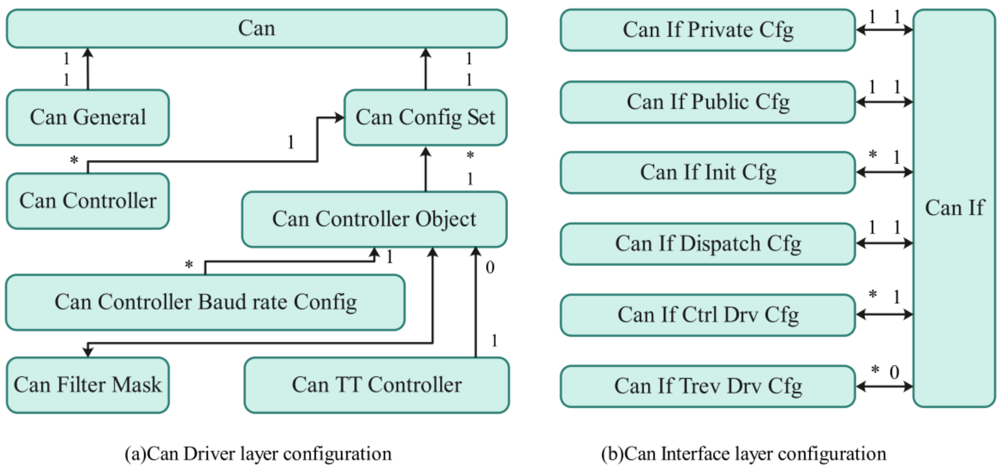


Figure 9. Can driver and interface layer configuration

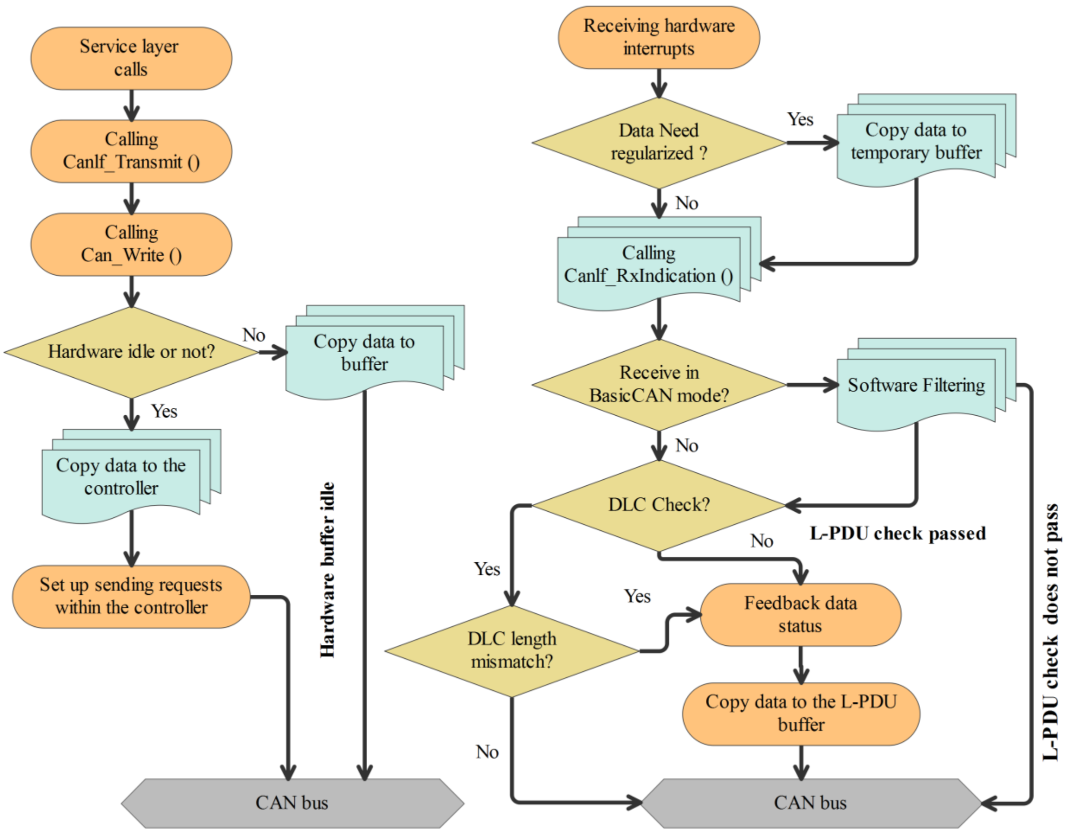

Service layer calls
Calling CanIf_Transmit ()
Calling Can_Write ()
Hardware idle or not?
No
Copy data to buffer
Yes
Copy data to the controller
Set up sending requests within the controller
Hardware buffer idle
CAN bus
Receiving hardware interrupts
Data Need regularized ?
Yes
Copy data to temporary buffer
No
Calling CanIf_RxIndication ()
Receive in BasicCAN mode?
Software Filtering
No
DLC Check?
L-PDU check passed
No
Yes
Yes
Feedback data status
DLC length mismatch?
Copy data to the L-PDU buffer
No
CAN bus
L-PDU check does not pass

Figure 10. Flow chart for sending and receiving data

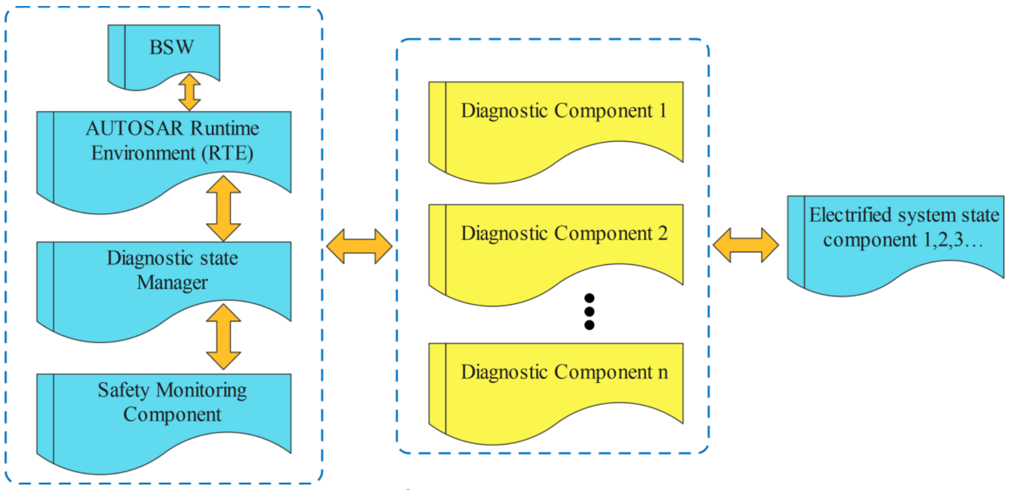

BSW
AUTOSAR Runtime Environment (RTE)
Diagnostic state Manager
Safety Monitoring Component
Diagnostic Component 1
Diagnostic Component 2
Diagnostic Component n
Electrified system state component 1,2,3…

Figure 11. ASW for Electric Vehicle Diagnostic System

FreezeFrame captures information about the environment and data at the time of the fault, while Extended Data Record includes information from software modules, such as frequency clock data. Due to the possibility of multiple data collection instances, there can be more than one FreezeFrame for a single event. The continuity of time-related data may be influenced by different sources and storage times.

In the diagnostic service processing, the Diagnostic Communication Manager (DCM) module follows a defined flow illustrated in Figure 12. Three sub-modules, namely DSL, DSD, and DSP, are developed within the DCM module to meet specific requirements. DSL interacts directly with the Protocol Data Unit Router (PduR), facilitating the reception and transmission of service response messages to fulfill service requests.

Upon receiving a DiagnosticsessionControl (0x10) service, DSL switches the diagnostic session mode, providing timing parameters such as the time interval for the requesting party to receive the service response message (P2CAN_Client). This interval sets the timeout mechanism of the application layer in the current session mode. When receiving a SecurityAccess (0x27) service, DSL returns the seed, verifies the received key, and decides whether to grant security access. DSL resets the session timeout timer, maintaining the current session mode without forwarding the service to the Diagnostic Service Dispatcher (DSD) for further processing.

DSD, the second module, verifies the validity of the service request message, checking supported services, session modes, security rights, and ECU status. If the message is valid, DSD routes the request to the Diagnostic Service Processor (DSP) module for execution. DSP, the third module, executes the precise service request operation. For tasks like reading or clearing fault information, DSP accesses the DEM module. For data upload/download or reading data streams, DSP accesses the memory stack. For input and output control requests, DSP uses DCM_Send/ReceiveSignal() to interact with the Runtime Environment (RTE) and access the Software Component (SWC).

4. TROUBLESHOOTING TESTS

To validate the accuracy of the fault determination in the diagnostic system and assess the system's configurability, this paper conducts tests on several commonly used services, as outlined in Table 1.

The Diagnostic Trouble Code (DTC) serves as an identification code presented when a fault occurs or is detected in an Electronic Control Unit (ECU). The fault information corresponding to the DTC can be retrieved by referencing a lookup table. A DTC comprises two parts: DTC Category and Failure Type. The DTC Category can be further categorized into four subsystems: Powertrain, Body, Chassis, and Network.

For our testing purposes, two DTCs, 0x00599A and 0x00559C, are configured with DID 0x2345, parameters P2 set to 50 ms, and P2* set to 5 s. CAN messages are transmitted and parsed following the ISO15765-2 and ISO11898-1 protocol specifications.

We conduct tests on the three sub-services of the 0x10 service using the diagnostic control tool in CANoe, and the results of these tests are summarized in Table 2.

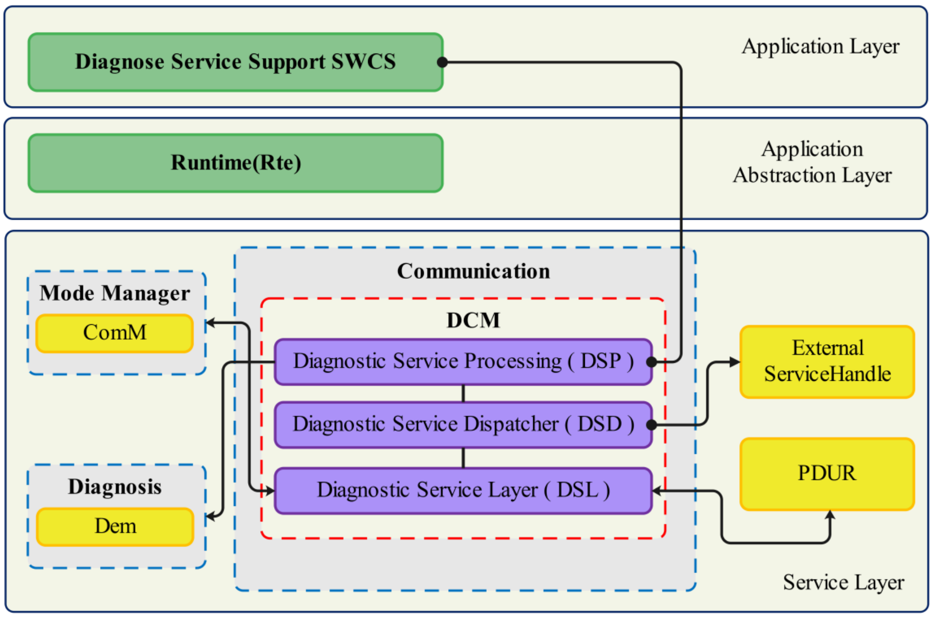

Diagnose Service Support SWCS
Application Layer
Runtime(Rte)
Application Abstraction Layer
Communication
Mode Manager
ComM
DCM
Diagnostic Service Processing (DSP)
External ServiceHandle
Diagnostic Service Dispatcher (DSD)
Diagnosis
Dem
Diagnostic Service Layer (DSL)
PDUR
Service Layer
Figure 12. DCM flow chart

TABLE 1. Diagnostic Services

Service ID	Service name
0X10	Diagnostic session control
0x27	Secure access control
0x22	Read data according to the data identifier
0x2E	Write data according to the data identifier
0x19	Read DTC according to the status mask
0x14	Clear DTC

TABLE 2. Diagnostic session-related service tests

Service ID	S/R	Data	Result
10 01	Send	02 10 01 00 00 00 00 00	Default session switching through
	Receive	06 50 01 00 32 01 F4 00	
10 02	Send	02 10 02 00 00 00 00 00	Refresh session switch passed
	Receive	06 50 02 00 32 01 F4 00	
10 03	Send	02 10 03 00 00 00 00 00	Extended session switching through
	Receive	06 50 0300 32 01 F4 00	

Based on the preceding test results, all three sub-service switches of the Diagnostic Session Control 0x10 service were successfully executed, with the corresponding parameters returned. Subsequently, service tests related to diagnostic fault codes (DTC) are detailed in Table 3.

From the aforementioned tests, the Read DTC (0x19 02) and Clear DTC (0x14) tests were successfully executed based on the status mask, while the Report DTC Number (0x19 01) test, also based on the status mask, did not pass. The configuration for the 0x19 01 service was not in place during the setup, and it appropriately returned a Negative Response Code (NRC) of 0x12, indicating that the sub-service is not supported—a correct result.

Service tests related to reading and writing data based on Data Identifiers (DID) are elaborated in Table 4.

Building on the preceding test outcomes, the Read and Write Data by Data Identifier (0x22 and 0x2E) service passed successfully. However, when attempting to read an unassigned Data Identifier (DID), the service appropriately returned a Negative Response Code (NRC) of 0x31.

To enhance security, the Vehicle Security Bridge (VSB) was reconfigured to establish the security access level and session for the DTC reading service. This adjustment ensures correct reading and clearing of DTCs if the security access level and session credentials are successfully authenticated.

TABLE 3. DTC-related service testing

Service ID	S/R	Data	Result
19 02	Send	03 19 02 2F 00 00 00 00	Read the DTC according to the status mask and pass the test
	Receive	10 0B 59 02 FF 00 59 9A	
	Send	30 00 14 00 00 00 00 00	
	Receive	21 2F 00 59 9C 2F 00 00	
14	Send	04 14 FF FF FF 00 00 00	Clear DTC, service test passed
	Receive	01 54 00 00 00 00 00 00	
19 01	Send	03 19 01 7F 00 00 00 00	Service failed
	Receive	03 7F 19 12 00 00 00 00	

TABLE 4. DID-related service testing

Service ID	S/R	Data	Result
22	Send	03 22 23 45 00 00 00 00	Read DID value successfully
	Receive	04 62 23 45 55 00 00 00	
2E	Send	04 2E 23 45 EB 00 00 00	The value of DID was modified successfully
	Receive	03 6E 23 45 00 00 00 00	
22	Send	03 22 23 45 00 00 00 00	Successfully read and re-write the DID value
	Receive	04 62 23 45 EB 00 00 00	
22	Send	03 22 23 44 00 00 00 00	Failure to read a value that does not support DID
	Receive	03 7F 22 31 00 00 00 00	

TABLE 5. Security access-related testing

Service ID	S/R	Data	Result
10 03	Send	02 10 03 00 00 00 00 00	Extended Session Switching
	Receive	50 03 00 32 01 F4 00 00	
27 01	Send	02 27 01 00 00 00 00 00	Request seeds and return them successfully
	Receive	67 01 12 34 00 00 00 00	
27 02	Send	04 27 02 12 39 00 00 00	Successfully send the key and successfully match the key
	Receive	07 02 00 00 00 00 00 00	
19 02	Send	03 19 02 2F 00 00 00 00	Read DTC according to the status mask; the test passes
	Receive	10 0B 59 02 FF 00 59 9A	
	Send	30 00 14 00 00 00 00 00	
	Receive	21 2F 00 59 9C 2F 00 00	

14	Send	04 14 FF FF FF 00 00 00	Clear DTC, the test passes
	Receive	01 54 00 00 00 00 00 00	
10 01	Send	02 10 01 00 00 00 00 00	Default session switching is done
	Receive	06 50 01 00 32 01 F4 00	
19 02	Send	03 19 02 2F 00 00 00 00	Security level not passed, cannot read DTC
	Receive	30 7F 19 33 00 00 00 00	

The results presented in Table 5 demonstrate the successful reading of Diagnostic Trouble Codes (DTC) when the session and security levels are validated. Conversely, in cases of session and security level failure, the service appropriately returned a Negative Response Code (NRC) of 0x33, indicating failed security verification.

To further substantiate the superiority of the proposed method, a comparative analysis of fault detection time and fault detection rates was conducted among three different methods. The experimental comparison results are detailed in Table 6.

The results depicted in Table 6 unequivocally showcase the superiority of our proposed method over the other two methods (5, 6), particularly in terms of fault detection rate and detection time.

TABLE 6. Comparison of Fault Detection Rate

Solution	Single core CPU occupancy rate	Fault detection success rate	Fault detection time
CAN bus [5]	74.94%	85.73%	0.0319
Structural analysis method [6]	68.53%	89.72%	0.0247
Our	40.68%	98.70%	0.0217

5. CONCLUSION

Building upon the existing electric vehicle fault diagnosis system, this paper delineates the design and implementation of an electric vehicle fault detection system adhering to the AutoSAR standard. The design encompasses diagnostic communication and function modules based on the diagnostic protocol, and comprehensive testing has been conducted. The proposed method showcases remarkable attributes, boasting a single-core CPU utilization rate of merely 40.68%, a fault detection time as low as 0.0217 seconds, and an impressive fault detection success rate of 98.70%. In direct comparison with the CAN bus and structural analysis methods, our proposed method outperforms, exhibiting a 12.97% and 8.98% improvement in fault detection success rates, respectively. Notably, this achievement is accompanied by more efficient test indices, resulting in heightened accuracy in fault detection results.

6. DATA AVAILABILITY STATEMENT

The labeled dataset used to support the findings of this study is available from the corresponding author upon request.

7. FUNDING STATEMENT

This study was supported by Self-made teaching instruments and equipment project of Tianjin Sino German University of applied technology. （No. ZDZY2020-02A）

8. AUTHORSHIP CONTRIBUTION STATEMENT

Xiaogang YANG: Writing-Original draft preparation Conceptualization, Supervision, Project administration.

9. REFERENCES

1. Abedinia O, Shorki A, Nurmanova V, Bagheri M. Synergizing Efficient Optimal Energy Hub Design for Multiple Smart Energy System Players and Electric Vehicles. IEEE Access. 2023;11:116650-116664. 10.1109/ACCESS.2023.3323201.
2. Lu M, Abedinia O, Bagheri M, Ghadimi N, Shafie-khah M, Catalão JPS. Smart load scheduling strategy utilising optimal charging of electric vehicles in power grids based on an optimisation algorithm. IET Smart Grid. Wiley Online Library; 2020;3(6):914–23. 10.1049/iet-stg.2019.0334
3. Abedinia O, Lu M, Bagheri M. An improved multicriteria optimization method for solving the electric vehicles planning issue in smart grids via green energy sources. IEEE Access. 2019;8:3465–81. 10.1109/ACCESS.2019.2960557
4. Gholami M, Sanjari MJ. Optimal Operation of Multi-Microgrid System Considering Uncertainty of Electric Vehicles. International Journal of Engineering. Materials and Energy Research Center; 2023;36(8):1398-1408 10.5829/ije.2023.36.08b.01
5. Bhosale AP, Mastud SA. Comparative Environmental Impact Assessment of Battery Electric Vehicles and Conventional Vehicles: A Case Study of India. International Journal of Engineering. Materials and Energy Research Center; 2023;36(5):965–78. 10.5829/ije.2023.36.05b.13
6. Ahmadigorji M, Mehrasa M. A robust renewable energy source-oriented strategy for smart charging of plug-in electric vehicles considering diverse uncertainty resources. International Journal

of Engineering, Transactions A: Basics. 2023;36(4):709–19. 10.5829/ije.2023.36.04a.01

7. Jian Y, Qing X, Zhao Y, He L, Qi X. Application of model-based deep learning algorithm in fault diagnosis of coal mills. Mathematical Problems in Engineering Hindawi Limited; 2020;2020:1–14. 10.1155/2020/3753274
8. Ochando FJ, Cantero A, Guerrero JI, León C. Data Acquisition for Condition Monitoring in Tactical Vehicles: On-Board Computer Development. Sensors. MDPI; 2023;23(12):5645. 10.3390/s23125645
9. Wang Y. Design of Electric Drive System of Electric Vehicle Based on CAN Bus. In: Journal of Physics: Conference Series. IOP Publishing; 2021;1982(1): 012131. 10.1088/1742-6596/1982/1/012131
10. Haur I. AUTOSAR compliant multi-core RTOS formal modeling and verification (Doctoral dissertation, École centrale de Nantes). 2022. https://theses.hal.science/tel-04025811/
11. Staron M, Staron M. AUTOSAR (AUTomotive Open System ARchitecture). Automotive Software Architectures: An Introduction. Springer; 2021;97–136. 10.1007/978-3-030-65939-4_5
12. AUTOSAR GbR. AUTOSAT Methodology V1.2.1. 2008;
13. AUTOSAT GbR. Specification of the Viretual Functional Bus V1.0.1 . 2008;
14. AUTOSAT GbR. Specification of Run-Time Environment V2.0.1 R3.0 Rev001 . 2008;
15. Khenfri F, Chaaban K, Chetto M. Efficient mapping of runnables to tasks for embedded AUTOSAR applications. Journal of Systems Architecture. Elsevier; 2020;110:101800. 10.1016/j.sysarc.2020.101800
16. Amato F, Coppolino L, Mercaldo F, Moscato F, Nardone R, Santone A. CAN-bus attack detection with deep learning. IEEE Transactions on Intelligent Transportation Systems. IEEE; 2021;22(8):5081–90. 10.1109/TITS.2020.3046974
17. AUTOSAR GbR. Specification of CAN Interface [EB/OL]. . 2009;
18. Chen W, Wang Y, Zhang Z, Qian Z. Syzgen: Automated generation of syscall specification of closed-source macos drivers. 2021 ACM SIGSAC Conference on Computer and Communications Security. 2021:ACM.749–63. 10.1145/3460120.3484564
19. Zhang K, Liu Y, Zhang J, Zhang G, Jin J, Li Y, et al. TDCA: improved optimization algorithm with degree distribution and communication traffic for the deployment of software components based on AUTOSAR architecture. Soft comput. Springer; 2023;27(12):7999–8012.10.1007/s00500-023-07989-1
20. Sandhya Devi RS, Sivakumar P, Balaji R. AUTOSAR architecture based kernel development for automotive application. International Conference on Intelligent Data Communication Technologies and Internet of Things (ICICI) 2018. 2019:Springer.10.1007/978-3-030-03146-6_104
21. Ran Z, Yan H, Zhang H, Li Y. Approximate optimal AUTOSAR software components deploying approach for automotive E/E system. International Journal of Automotive Technology.2017;18:1109-1119. 10.1007/s12239-017-0108-3
22. Lv J, Qu C, Du S, Zhao X, Yin P, Zhao N, et al. Research on obstacle avoidance algorithm for unmanned ground vehicle based on multi-sensor information fusion. Mathematical Biosciences and Engineering.2021;18(2):1022–39. 10.3934/mbe.2021055
23. Duan M, Darvishan A, Mohammaditab R, Wakil K, Abedinia O. A novel hybrid prediction model for aggregated loads of buildings by considering the electric vehicles. Sustain Cities Soc. 2018;41:205–19. 10.1016/j.scs.2018.05.009
24. Kuspan B, Bagheri M, Abedinia O, Naderi MS, Jamshidpour E. The influence of electric vehicle penetration on distribution transformer ageing rate and performance. 2018 7th International Conference on Renewable Energy Research and Applications (ICRERA). 2018: IEEE. 10.1109/ICRERA.2018.8566966
25. Nurmanova V, Sultanbek A, Bagheri M, Ahangar RA, Abedinia O, Phung T, et al. Distribution Transformer Frequency Response Analysis: Behavior of Different Statistical Indices During Inter-disk Fault. 2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe). 2019:IEEE. 10.1109/EEEIC.2019.8783252
26. Devi RSS, Kumar BV, Sivakumar P, Lakshmi AN, Tripathy R. Bootloader design for advanced driver assistance system. Software Engineering for Automotive Systems. Boca Raton:CRC Press;2022. p. 31-44
27. Devi RSS, Sivakumar P, Lakshmi AN, Tripathy R. 3 Design Bootloader Advanced Driver Assistance System. Software Engineering for Automotive Systems: Principles and Applications. Boca Raton: CRC Press; 2022. p.31-44
28. Xiao D, Wang N, Shen X, Landulfo E, Zhong T, Liu D. Development of ZJU high-spectral-resolution LiDAR for aerosol and cloud: Extinction retrieval. Remote Sensing. 2020;12(18):3047. 10.3390/rs12183047
29. Wang N, Shen X, Xiao D, Veselovskii I, Zhao C, Chen F, et al. Development of ZJU high-spectral-resolution lidar for aerosol and cloud: Feature detection and classification. Journal of Quantitative Spectroscopy and Radiative Transfer. 2021;261:107513. 10.1016/j.jqsrt.2021.107513
30. Liu D, Yang Y, Cheng Z, Huang H, Zhang B, Shen Y. Development of the ZJU polarized near-infrared high spectral resolution lidar. International Symposium on Photoelectronic Detection and Imaging 2013: Laser Sensing and Imaging and Applications. 2013: SPIE. 10.1117/12.2035435

Persian Abstract

چکیده

با افزایش ادغام ECU در وسایل نقلیه مدرن، پیچیدگی شبکه های وسایل نقلیه نیز به طور مداوم در حال افزایش است. ارتباطات تشخیصی، به عنوان یک عملکرد کلیدی در شبکه های وسایل نقلیه، با چرخه های توسعه طولانی تر و دشواری های بالاتری مواجه است. به منظور بهبود قابلیت استفاده مجدد و قابل حمل بودن نرم افزار، این مطالعه تحقیقات مربوطه را تجزیه و تحلیل کرد و یک سیستم تشخیص عیب خودروی الکتریکی را بر اساس گذرگاه CAN با استفاده از معماری ارتباط تشخیصی توصیه شده توسط استاندارد AUTOSAR پیشنهاد کرد. با اتخاذ AUTOSAR، هدف ما بررسی یک روش توسعه نرم افزار جدید برای سیستم های تشخیص عیب خودرو برای رفع این محدودیت بود. ماژول ارتباطی و تشخیصی این مطالعه با استفاده از AUTOSAR پیاده‌سازی شد و نیازی به توسعه‌دهندگان برای بررسی پیچیدگی سخت‌افزار و پیاده‌سازی ارتباطات را از بین برد. توسعه دهندگان اکنون می توانند روی طراحی ویژگی های نرم افزار برای تشخیص عیب تمرکز کنند. نتایج تجربی نشان می دهد که نرخ استفاده از CPU تک هسته ای روش پیشنهادی در مقاله تنها ٤٠.٦٨٪ است. زمان تشخیص خطا ۰.۰۲۱۷ است. میزان موفقیت تشخیص عیب ۹۸.۷۰ درصد است که به ترتیب ۱۲.۹۷ درصد و ۸.۹۸ درصد بیشتر از روش تحلیل سازه و گذرگاه CAN است. شاخص های تست به طور موثر کاهش می یابد، و نتایج تشخیص خطا دقیق تر است. بررسی این روش جدید توسعه نرم افزار در محصولات الکترونیکی خودرو، کارایی نرم افزار سیستم عیب یابی خودرو را تا حد زیادی بهبود می بخشد.