ISSN: 2688-8645

# Research on the Design of Upper Computer Management Software for Automotive Software Upgrade Testing

Yongjian Zhu[1, 2, a], Manna Wang[1, 2, b], Ying Jing[3, c], Zhapa Mu[1, 2, d], Nirunze Yang[1, 2, e]

[1]China Automotive Inspection Center (Tianjin) Co., Ltd., China

[2]China Automotive Technology Research Center Co., Ltd., China

[3]China Quality Certification Center Co., Ltd, China

[a]zhuyongjian@catarc.ac.cn, [b]wangmanna@catarc.ac.cn, [c]690966326@qq.com,

[d]muzhapa@catarc.ac.cn, [e]yangnirunze@catarc.ac.cn

## Abstract

**With the rapid development of automotive electronics technology, software upgrades have become a key link in ensuring vehicle performance and safety. As the core tool for software upgrade testing, the design rationality and functionality of the upper computer management software directly affect the efficiency and accuracy of upgrade testing. This article first analyzes the requirements and challenges of automotive software upgrade testing, and then proposes an upper computer management software architecture based on modular and scalable design ideas. This architecture integrates functional modules such as test task management, device communication control, data recording and analysis, effectively improving the automation and intelligence level of software upgrade testing. Through practical case verification, the upper computer management software designed in this article has shown excellent performance in improving testing efficiency and reducing human error rates, providing strong technical support for automotive software upgrade testing. This study is of great significance for promoting the continuous progress of automotive electronic technology.**

## Keywords

**Software Upgrade; Upper Computer; Testing; OTA.**

## 1. Introduction

With the rapid development of the automotive industry, the degree of automotive electronicization is increasing day by day, and the application of software in automobiles is becoming more and more widespread. It has become a key factor determining the performance, safety, and comfort of automobiles. The frequent updates and upgrades of automotive software are not only to fix known issues, but also to introduce new features and enhance user experience. However, the risks involved in the software upgrade process cannot be ignored. Once the upgrade fails, it may lead to the failure of vehicle functions and even cause safety accidents. Therefore, automotive software upgrade testing is particularly important. As a core component of automotive software upgrade testing, the upper computer management software plays a crucial role. It is responsible for coordinating communication between testing equipment and the software being tested, monitoring the upgrade process, recording and analyzing test data to ensure the security and reliability of software upgrades. However, traditional upper computer management software often has problems such as single functionality, poor scalability, and low degree of automation, making it difficult to meet the increasingly complex needs of automotive software upgrade testing. In response to this situation, this article aims to design a comprehensive, easy to expand, and highly automated

ISSN: 2688-8645

automotive software upgrade testing upper computer management software. Through in-depth research on the specific requirements and challenges of automotive software upgrade testing, combined with advanced software development concepts and technical methods, this article proposes an innovative software architecture design scheme. This plan aims to improve the efficiency and accuracy of software upgrade testing, reduce human error rates, and provide strong technical support for the continuous iteration and optimization of automotive software. The research in this article not only has important theoretical value, but also has profound guiding significance for the practical application of the automotive industry.

## 2. Functions of Upper Computer Management Software

The goal of this product is to develop a reliable and scalable OTA simulation testing system for Tianjin Automotive Research Institute, which serves as one of the upper management software for testing. Establish management and monitoring of various subsystems through this software. Establish functionality and stability for testing OTA systems. Simultaneously verify whether the operational status of the OTA system meets the design requirements in various scenarios.

This software system will include the following main functions:

**1) Script library management:** providing the function of managing test scripts, including creating, deleting, synchronizing scripts, and other operations. Users can organize and store test scripts in the script library, facilitating the execution and management of testing tasks.

**2) Test task management:** Provides functions for managing test tasks, including creating, editing, and deleting test tasks. Users can create test tasks in the system, and create, edit, and delete test cases and associated test scripts within the test tasks. Package testing tasks for the corresponding testers to execute, track the progress and results of task execution.

**3) System management**: Provides functions for managing system configuration and user permissions, including system settings, user management, permission management, and other operations. Administrators can configure and manage the system, including assigning user permissions and setting system parameters.

**4) Equipment Management:** Provides information related to managing testing equipment, including test benches, vehicle networking process simulation modules, in vehicle protocol master-slave node simulation modules, and the running status of data screens.

**5) Problem Management:** Provides functionality for managing defect issues, including module information, version, defect type, defect level, recurrence probability, description, testing environment, and other information.

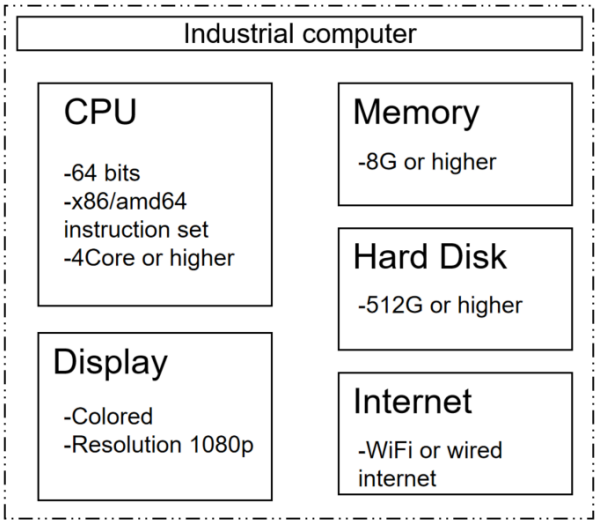


**Fig 1.** Runtime environment dependencies

ISSN: 2688-8645

This article designs a complete software system based on the above functional requirements, as shown in Figure 1 for the software's operating environment dependencies. The automotive software upgrade test upper computer management software system is designed specifically for efficient operation in specific environments. It requires a Windows 10 64 bit operating system, hardware configuration with a 64 bit CPU, support for x86/amd64 instruction set, and at least a 4-core processor, with 8GB or more of memory to ensure smooth operation. In addition, the software also relies on a series of key software components, including 64 bit versions of Git for version control, Docker 20 or higher for containerized deployment and testing environment management, and JDK (Java Development Kit) 8 or higher as the foundational environment for software development and operation. These requirements together constitute the fundamental support system for the operation of the software system.

## 3. System Architecture Design

### 3.1. System Introduction

The upper management software is a comprehensive tool platform, whose core function is to efficiently configure and manage tasks and use cases for automotive software upgrade testing. The software mainly consists of three core components: front-end page UI, back-end interface API, and database. The front-end UI serves as a direct interface for user interaction with the system, designed as an intuitive and user-friendly web interface. Users can easily create, edit, and view test tasks and cases through this interface, achieving seamless interaction with the system. The backend API plays a crucial role in providing data storage and identity authentication support for the system. Through a series of carefully designed API interfaces, the system is able to securely and efficiently store and manage data related to testing tasks and use cases, and ensure accurate verification of user identities, thereby ensuring the security and stability of the system. The database, as the core component for storing business data in the system, provides a solid foundation for data storage and modeling. Through reasonable database design and optimization, the system can efficiently store, retrieve, and analyze data of testing tasks and cases, providing strong support for the automation and intelligence of the testing process.

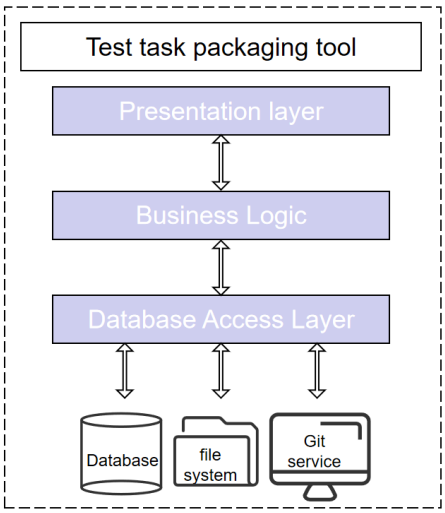


**Fig 2.** System Interaction Diagram

These three components collaborate and work together to enable the upper management software to efficiently generate and manage testing task packages, providing a comprehensive and reliable solution for automotive software upgrade testing. The upper management

ISSN: 2688-8645

software system adopts a typical three-layer architecture, as shown in Figure 2, which includes the Presentation Layer: responsible for receiving user input and presenting information to the user, achieving interaction with the user. Business Logic Layer: It includes the system's business logic and processing rules, and is responsible for implementing the core functions of the system. Data Access Layer: Responsible for interacting with data storage and performing operations such as adding, deleting, modifying, and querying data.

The upper management software shown in Figure 3 is a web service built on Node.js, designed to help users manage Git script repositories, test cases, testing tasks, and problem management. In addition, the monitoring vehicle networking process simulation module and the in-vehicle protocol master-slave node simulation module. The front-end is based on Vue.js to implement user interface and interaction, while the back-end is based on Vimesh to handle business logic, service interfaces, and data management, including cross platform services: running in the form of backend services (or Docker containers) on local or remote Linux or Windows hosts. Front end framework: Use popular front-end frameworks Vue.js and Element UI to build dynamic and responsive user interfaces. API interface: A well-defined API interface based on RESTful, used for data transmission and interaction between the front-end and back-end. Transmission format: Use the JSON standard data format for data transmission to ensure readability and interoperability of the data. Identity authentication: Use JWT identity authentication mechanism to ensure the security and reliability of the interface. Business layer: Based on the Vimesh framework, the core business logic and processing rules of the system are implemented to maintain the independence and reusability of the business logic. Data layer: Based on Sequelize, responsible for interacting with MySQL database, implementing data read and write operations, and providing data access interfaces to the business logic layer. Service layer: The Vimesh framework provides various service interfaces for the system, including object storage and other services.

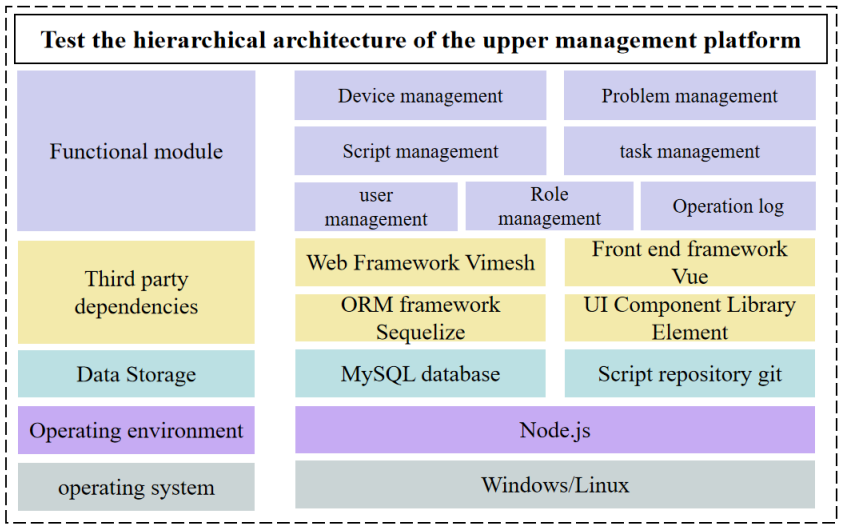


**Fig 3.** Upper level management software architecture

## 3.2. Data Architecture

The system has chosen a database system that is suitable for business needs to support data storage and management. Our company uses MySQL. It is a widely used open-source relational database management system (RDBMS) that uses a database query language called Structured Query Language (SQL). Create, query, update, and delete data in the database using SQL language. At the same time, it also provides a series of tools and interfaces that allow developers to easily interact with the database. MySQL performs well in handling large amounts of data and has high scalability. Through configuration and optimization, it can handle complex queries

ISSN: 2688-8645

and a large number of concurrent users. In addition, MySQL provides various security features such as user permission management, encrypted connections, audit logs, etc. to protect the database from unauthorized access and attacks.

The data flow of the system describes the flow and processing of data within and outside the system. The data process includes the following steps:

1) Data Input: describes how data enters the system, including user input, third-party interfaces, etc.
2) Data Processing: Describes the process of processing data within a system, including data cleaning, conversion, computation, etc.
3) Data Output: describes how data is output to the outside of the system, including user interface display, report generation, data export, etc.

### 3.3. Security Architecture

In modern software development and testing environments, the security architecture of upper-level management software is the key to ensuring data security and stable system operation. The security architecture of the upper management software designed in this article mainly includes four core components: identity authentication, access control, data encryption, and security vulnerability prevention.

**Identity authentication** mechanism is the first line of defense in security architecture, and its core is to ensure that only legitimate users can access system resources. This system adopts a multi factor authentication mechanism, combined with the traditional authentication method of username and password, and introduces dynamic SMS verification codes or biometric technologies such as fingerprint recognition or facial recognition to enhance the accuracy and security of identity authentication. This mechanism effectively prevents the intrusion of unauthorized users and ensures that system resources are not accessed by unauthorized users.

**access control and data encryption** access control mechanism is designed based on the Role Based Access Control (RBAC) model, which assigns different roles and permissions to users to strictly control their access to system resources. This mechanism not only enhances the flexibility of the system, but also ensures the protection of sensitive information. At the same time, the system has implemented strict data encryption measures for the transmitted and stored data. During data transmission, SSL/TLS protocol is used for encryption to prevent data from being intercepted or tampered with. When storing data, sensitive information is encrypted to ensure that even if the database is accessed illegally, the data cannot be directly read, greatly improving data security.

**Security vulnerability prevention and continuous monitoring**. Security vulnerability prevention is an indispensable part of system security architecture. This system ensures that it is protected from known vulnerabilities through regular security audits, timely patch management, and strict security configurations. At the same time, professional vulnerability scanning tools are used to regularly scan the system, promptly discovering and fixing potential security vulnerabilities. In addition, the system has established a continuous monitoring mechanism to monitor the operating status of the system in real time. Once abnormal behavior or potential threats are detected, the warning mechanism is immediately triggered to ensure that the system can respond quickly and take measures, thereby effectively preventing risks caused by security vulnerabilities.

## 4. Database Design

In the design of the upper computer management software for automotive software upgrade testing, data design is a key link to ensure that the software can efficiently and accurately manage and process test data. Data design not only concerns the structure and efficiency of

ISSN: 2688-8645

data storage, but also directly affects multiple aspects of software testing processes, result analysis, and report generation.

Firstly, in terms of data model design, we adopted a relational database to store and manage test data. According to the requirements of automotive software upgrade testing, we have designed multiple interrelated tables, including software version information table, test case table, test result table, test log table, etc. These tables are linked through foreign keys to form a complete data chain, ensuring data integrity and consistency. For example, the software version information table records detailed information about each software version, including version number, release date, feature description, etc.; The test case table records key information such as the identification, description, and expected results of each test case; The test result table records key information such as actual results, testing time, and testing personnel for each test, and is associated with the test case table and software version information table through foreign keys, achieving data traceability and correlation analysis. In the design of data tables, we focus on the reasonable selection of fields and optimization of indexes. For fields frequently used for querying and filtering, we have set indexes to improve query efficiency.

At the same time, in order to avoid data redundancy and improve data consistency, we follow the standardization principles of database design, ensuring that each field is atomic and indivisible, and that each non primary key field is completely dependent on the primary key. When necessary, in order to optimize query performance, we conducted moderate anti normalization design by adding redundant fields or creating redundant tables. In addition, in terms of data design, we have also considered backup and recovery mechanisms for data. To ensure the security and availability of test data, we have adopted a combination of regular automatic backups and manual backups, storing backup data on secure and reliable storage media.

At the same time, we have also designed a data recovery process to quickly recover data in the event of data loss or damage, ensuring the continuity and integrity of testing work.

## 5. Conclusion

This article explores in detail the important role of data design in software design by studying the design of upper computer management software for automotive software upgrade testing. By adopting a relational database, designing a reasonable data model and table structure, optimizing field selection and index design, and considering data backup and recovery mechanisms, we ensure that the software can efficiently and accurately manage and process test data. These designs not only improve the testing efficiency and accuracy of the software, but also provide a solid foundation for the subsequent maintenance and expansion of the software. In the future, we will continue to optimize data design and introduce more advanced technologies and methods to further enhance the performance and functionality of the upper computer management software for automotive software upgrade testing. At the same time, we will also pay attention to the development trends of the automotive software industry and changes in user needs, continuously iterate and improve software design, and contribute to the digital transformation and intelligent upgrading of the automotive industry.

## References

[1] Gao Fei; Yang Chengxiao, The Development of Over the Air (OTA) System for Automotive Vehicle Upgrades [J]. Heavy duty vehicles, 2022 (05).

[2] Wu Zhi; Liu Tianyu; Jia Xianfeng, OTA upgrade safety design for intelligent connected vehicles [J]. Practical Automotive Technology, 2022 (03).

ISSN: 2688-8645

[3] Li Jian; Zhang Chao; Interpretation and Testing Practice of Automotive Information Security Standards [J]. Close the morning Quality and Standardization, 2021 (08).

[4] Li Lian; Zhao Guojuan; Ren Guangle. OTA Implementation [4]Scheme and Automotive Design Analysis [J]. Practical Automotive Technology, 2020 (14).

[5] Zhang Qian. A Comprehensive Framework for OTA Automotive Industry Applications [J]. Practical Automotive Technology, 2020 (11).

[6] Wang Dongliang; Tang Lishun; Chen Bo; Liu Xu; Liu Chuang, Research on OTA Function Design of Intelligent Connected Vehicles [J]. Automotive Technology, 2018 (10).