Innovation and Technology Studies
Print ISSN: 3007-6919
Online ISSN: 3007-6927
DOI: https://doi.org/10.61784/its3021

RECONSTRUCTION OF EMBEDDED SOFTWARE SOLUTIONS AND SERVICE-ORIENTED TRANSFORMATION FOR INTELLIGENT TERMINALS

XingNan Yu
Jiangsu University, Zhenjiang 212013, Jiangsu, China.

Abstract: With the in-depth integration of the Internet of Things, artificial intelligence and edge computing technologies, intelligent terminals have evolved from single-function hardware carriers into smart nodes with multi-scenario applicability, high interactivity and iterative capability. The traditional solution model of embedded software featuring one-time delivery, closed-loop development and hardware binding can hardly adapt to the dynamic, personalized and long-term market demands of intelligent terminals. Focusing on the scenarios of intelligent terminals, this paper analyzes the pain points and bottlenecks of traditional embedded software solutions, and constructs a reconstructed architecture of embedded software adapted to resource-constrained environments by combining domain-driven design, lightweight microservices, cloud-edge collaboration and other technologies. Meanwhile, centering on the core of service-oriented transformation, it builds a full-life-cycle service operation system and explores the implementation path of transforming from "product delivery" to "value-added service". The feasibility and benefits of solution reconstruction and service-oriented transformation are verified through empirical analysis of typical intelligent terminal cases. Finally, safeguard measures from the three dimensions of technology, management and ecology are proposed. The research results can provide theoretical reference and practical guidance for the upgrading and iteration of embedded software enterprises and the high-quality development of the intelligent terminal industry.
Keywords: Intelligent terminal; Embedded software; Solution reconstruction; Service-oriented transformation; Cloud-edge collaboration; Lightweight microservices

1 INTRODUCTION

The global intelligent terminal industry is undergoing rapid iteration at present, with the continuous expansion of categories such as smart home, in-vehicle intelligence, industrial terminals and wearable devices. Terminal functions are being upgraded towards active perception, intelligent decision-making and remote interaction, posing higher requirements for the flexibility, scalability and security of embedded software. Traditional embedded software adopts a monolithic architecture with a high degree of hardware-software coupling, which suffers from long development cycles, low module reusability and high iteration costs. In addition, its service model is dominated by one-time delivery, which fails to meet the demands for long-term operation and maintenance, functional upgrading and personalized customization, thus restricting product competitiveness and corporate profit margins. Therefore, the reconstruction and service-oriented transformation of embedded software have become an industry trend [1,2].

Foreign relevant research started early, with mature technologies in lightweight architecture, service-oriented architecture (SOA) and cloud-edge collaboration, and formed service models such as subscription and pay-as-you-go. Domestic research mostly focuses on software development and architecture optimization, lacking solutions tailored to the characteristics of resource constraints and high real-time performance of intelligent terminals, and the service models lack systematic design, leading to a disconnect between theory and practice. To this end, this paper comprehensively adopts literature research, case analysis and empirical research methods to sort out industry pain points, construct a software reconstruction framework and service-oriented transformation system adapted to intelligent terminal scenarios, verify the feasibility through empirical evidence and propose safeguard measures. This not only fills the relevant theoretical gaps but also provides practical reference for enterprise upgrading.

2 CORE PAIN POINTS OF TRADITIONAL EMBEDDED SOFTWARE SOLUTIONS

2.1 Technical Architecture Bottlenecks

Most traditional embedded software adopts a monolithic and closed architecture with high code coupling and low module reusability. The addition of new functions requires overall reconstruction, resulting in long development cycles and high debugging difficulty. Hardware and software are deeply bound, leading to poor cross-platform portability and inability to adapt to the rapid iteration of various types of intelligent terminals. The lack of standardized interfaces results in poor cloud-edge data interaction, making it difficult to achieve remote management and control as well as upgrading.

2.2 Inefficient Development Processes

The waterfall development model is adopted, where the links of demand research, development, testing and delivery are fragmented, leading to a slow response to market changes. The testing phase relies on physical hardware, and the simulation test tools are inadequate, resulting in high cost and long cycle for bug fixing. The absence of a unified development platform and component library leads to prominent repeated development and serious resource waste.

2.3 Lagging Service Models

The service model is dominated by one-time product delivery, with only basic maintenance services provided after sales, and a lack of value-added services such as full-life-cycle operation and maintenance, functional upgrading and personalized customization [3]. Customer stickiness is low, enterprises rely on hardware sales for profits, and the proportion of service revenue is extremely low. The service process is non-standard with slow response speed and poor user experience, making it difficult to form brand competitiveness.

2.4 Security and Compliance Risks

Intelligent terminals involve sensitive information such as user privacy and industrial data. Traditional software lacks a sound security protection mechanism, leading to potential leakage risks in data transmission and storage. The insufficient compliance in functional safety and information security makes it hard to meet the regulatory requirements of high security level scenarios such as in-vehicle, medical and industrial control.

3 RECONSTRUCTION DESIGN OF EMBEDDED SOFTWARE SOLUTIONS FOR INTELLIGENT TERMINALS

3.1 Core Principles of Reconstruction

Combined with the characteristics of intelligent terminals, the solution reconstruction follows five core principles: first, lightweight adaptation, which balances resource constraints and function implementation and optimizes computing power and memory usage; second, loosely coupled architecture, which realizes hardware-software decoupling and module splitting to improve reusability and scalability; third, real-time performance guarantee, which meets the low-latency requirements for instruction execution and data processing of intelligent terminals; fourth, security and compliance, which runs through the entire process of development, deployment and operation to build a solid security defense; fifth, cloud-edge collaboration, which achieves efficient linkage between terminal-side execution and cloud-side management and control [4-6].

3.2 Technical Architecture Reconstruction

3.2.1 Lightweight layered service architecture

Abandoning the traditional monolithic architecture, a four-layer lightweight layered service architecture of "Resource Layer - Service Layer - Application Layer - Infrastructure Layer" adapted to intelligent terminals is designed to separate business logic from technical implementation. The overall resource usage of the architecture is reduced by more than 40% compared with the traditional monolithic architecture. Figure 1 shows the schematic diagram of the lightweight layered service architecture, clarifying the boundaries, functions and interaction standards of each layer.

Plaintext

```
+-----------------------------------------------------------+
| Application Layer: Scenario-oriented personalized applications        |
|              (Smart Home/In-vehicle/Industrial Scenarios) | Combine service components on demand, no hardware dependency
|       +----------------+----------------+----------------+          |
|       | Scenario App 1 | Scenario App 2 | Scenario App 3 |          |
+-----------------------------------------------------------+
| Service Layer: Lightweight microservice component cluster (independently deployed, flexibly invoked, standardized interfaces) |
|       +------------+-------------+----------------+--------+
|       | Perception Service | Control Service | Communication Service |          |
|       +------------+-----------+---------  ---+-----------+
|       | Security Service | Data Service | Upgrade Service      |          |
|       (Hardware Encryption/Permission) | (Local/Edge Caching) | (Remote/Local Upgrade) |          |
+-----------------------------------------------------------+
| Resource Layer: Hardware abstraction and data collection (shielding hardware heterogeneity, unified interfaces) |
|       +----------+--------------+-------------+-----------+
|       | Hardware Abstraction Layer (HAL) | Peripheral Management Module | Data Collection Module |          |
|       (Unified chip/peripheral interfaces) | (Standardized driver invocation) | (Multi-protocol data parsing) |          |
+-----------------------------------------------------------+
```

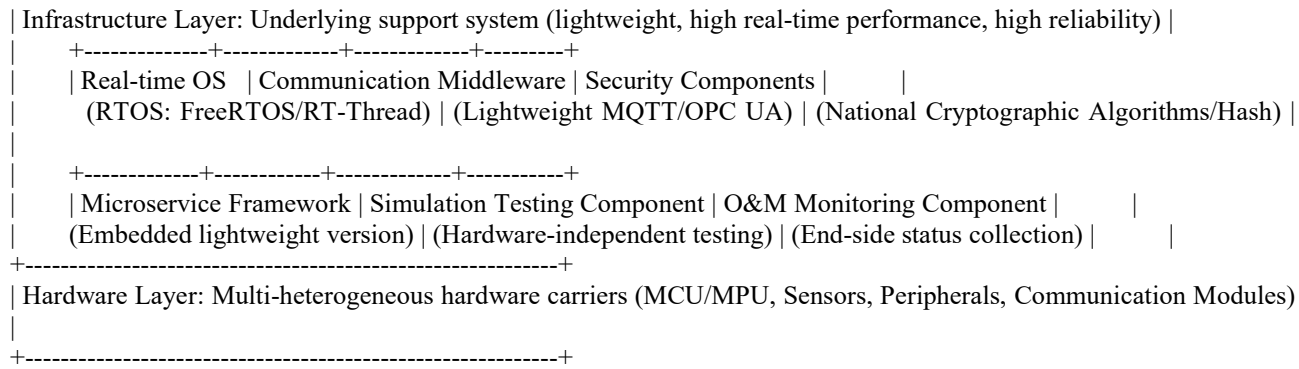


Figure 1 Schematic Diagram of Lightweight Layered Service Architecture for Embedded Software of Intelligent Terminals

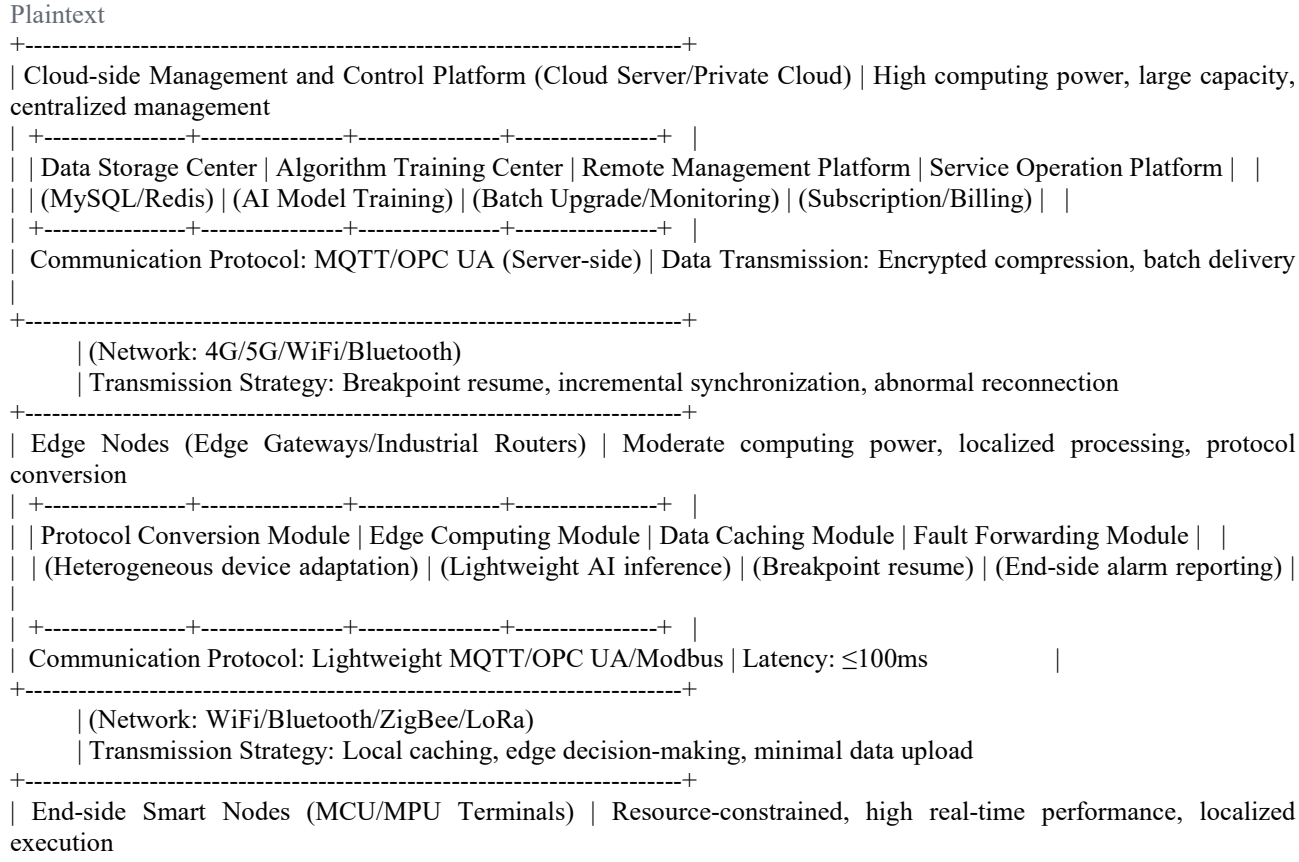
Key design points of the architecture:
Microservice components in the service layer adopt a "functional atomization" design, where a single component only implements a single core function to reduce resource usage;
All layers interact through standardized RESTful/IPC interfaces, and the interfaces adopt the Protobuf lightweight serialization protocol to reduce data transmission volume;
The infrastructure layer selects domestic lightweight RTOS (e.g., RT-Thread, AliOS Things), which is adapted to multiple hardware platforms of MCU/MPU and supports dynamic component loading.

3.2.2 Hardware-software decoupling technology

The Hardware Abstraction Layer (HAL) and instruction set abstraction layer are introduced to unify hardware interfaces and communication protocols, shield the differences between different chips and peripherals, and realize cross-platform portability of software code [7]. Standardized API interfaces are adopted to achieve low-coupling interaction between service components and improve software reusability. A dynamic loading mechanism is supported to complete the update of functional modules without restarting the terminal, reducing iteration costs.

3.2.3 Optimization of cloud-edge collaboration architecture

A three-level cloud-edge collaboration architecture of "cloud-side management and control platform - edge node - terminal-side smart node" is constructed. Aiming at the characteristics of resource constraints of intelligent terminals, the communication protocol, data transmission strategy and edge decision logic are optimized. Figure 2 shows the overall design of the cloud-edge collaboration architecture, and Table 1 lists the functional and performance quantitative indicators of the cloud, edge and terminal layers.



| +----------+----------+----------+----------+ |
| | Real-time Control Module | Local Data Processing | Edge Decision Execution | Local Abnormal Alarm | |
| | (≤100ms response) | (Lightweight computing) | (Cloud strategy implementation) | (Available without network) | |
| +----------+----------+----------+----------+ |
| Communication Protocol: Ultra-lightweight MQTT-SN/CoAP | Data Upload: Only feature values/abnormal data uploaded |
+----------+

Figure 2 Design Diagram of Cloud-edge Collaboration Architecture for Embedded Software of Intelligent Terminals

Table 1 Functional and Performance Quantitative Indicators of the Three-level Cloud-edge-terminal Architecture

Layer	Core Functions	Hardware Computing Power Requirements	Response Latency Requirements	Data Transmission Strategy	Resource Usage Constraints
Cloud-side	Centralized management, algorithm training, data storage	≥4-core 8G server	No hard constraints	Encrypted compression, batch delivery, incremental synchronization	No hard constraints
Edge node	Protocol conversion, edge inference, data caching	≥dual-core 1G edge gateway	≤100ms	Local caching, breakpoint resume, protocol conversion	Memory ≤512MB
Terminal-side	Real-time control, local collection, abnormal alarm	8/32-bit MCU/low-spec MPU	≤100ms	Minimal data, abnormal reporting, on-demand upload	Flash memory ≤16MB, Memory ≤1MB

Core optimization technologies:
Lightweight communication protocol: MQTT-SN (MQTT for Sensor Networks) is adopted on the terminal side to replace the traditional MQTT, with the minimum protocol header of only 2 bytes, reducing the terminal's communication resource usage;
Lightweight data transmission: The terminal side only uploads feature value data (instead of raw data). For example, a smart door lock only uploads the feature code of abnormal unlocking, reducing the data volume by more than 90%;
Edge decision algorithm: Lightweight machine learning models (e.g., decision tree, Naive Bayes) with a volume ≤1MB are deployed on edge nodes to realize localized anomaly identification and reduce cloud-edge interaction.

3.3 Development Process Reconstruction

Abandoning the traditional waterfall development model, the agile development and DevOps model are adopted to build an integrated development and testing platform. The entire process of demand management, code development, simulation testing, deployment and launch, and iterative optimization is integrated to realize the integration of development, testing and operation and maintenance. A component-based development library is built to deposit general functional modules and reduce repeated development. Virtual simulation test tools are introduced to complete the preliminary testing without physical hardware, shortening the development cycle and reducing debugging costs. A closed-loop feedback mechanism is established to quickly respond to market demands and user feedback and achieve continuous iterative optimization [8,9].

3.4 Security System Reconstruction

A full-link security protection system of "terminal-edge-cloud" is constructed. On the terminal side, hardware encryption, secure boot and permission management and control mechanisms are adopted to prevent firmware tampering and data leakage. Firewalls and intrusion detection modules are deployed on edge nodes to ensure the security of data transmission. On the cloud side, data encryption, access authentication and log audit mechanisms are established in compliance with the requirements of the Data Security Law, Personal Information Protection Law and other regulatory rules. Differentiated security solutions are formulated for different scenarios to meet the functional and information security standards of industries such as in-vehicle, medical and industrial control.

4 CONSTRUCTION OF SERVICE-ORIENTED TRANSFORMATION MODEL FOR EMBEDDED SOFTWARE

4.1 Core Logic of Service-oriented Transformation

Centering on user value, service-oriented transformation breaks the traditional thinking of "one-time delivery", upgrades embedded software from a single product to a sustainable service, and constructs an integrated solution of

"product + service" [10]. It realizes the transformation of profit model from hardware sales to "hardware + subscription service + value-added service", improving customer stickiness and the long-term income of enterprises.

Core logic: With the reconstructed lightweight software architecture as technical support and the cloud-edge collaboration platform as the operation carrier, it covers the entire life cycle of intelligent terminals from early customization, mid-term deployment to late value-added, and provides standardized, personalized and long-term services to achieve a win-win situation of value for both enterprises and users. Figure 3 shows the relationship between the core logic of service-oriented transformation and technical support.

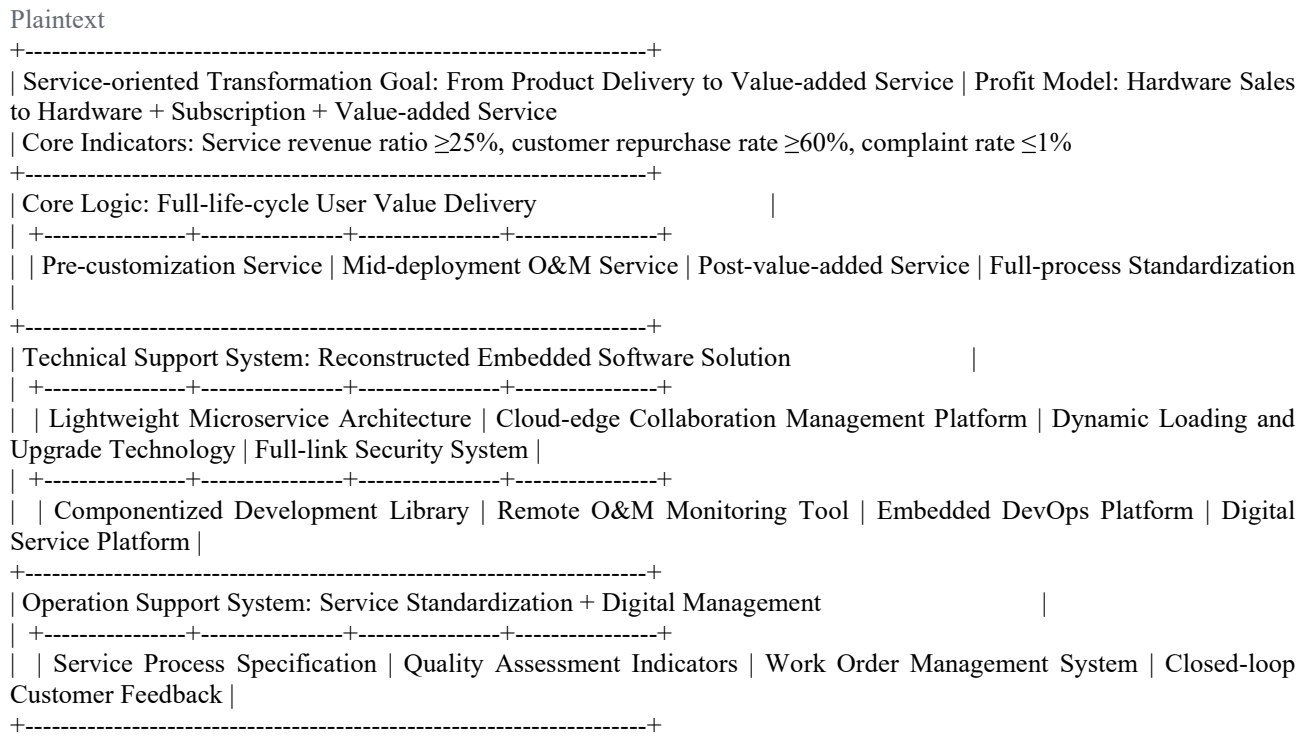


Figure 3 Relationship between Core Logic and Technical Support of Service-oriented Transformation for Embedded Software

4.2 Design of Full-life-cycle Service System

4.2.1 Early customized services

In response to the demands of intelligent terminals in different industries and scenarios, pre-sales services such as demand research, solution design, customized software development, and hardware-software adaptation testing are provided. A personalized configuration platform is built to support users to independently select functional modules and realize on-demand customization. Technical consulting and solution evaluation services are offered to assist customers in optimizing product design and reducing early investment costs.

4.2.2 Mid-term deployment and operation services

One-stop deployment services such as on-site deployment, remote debugging and system initialization are provided. A remote operation and maintenance monitoring platform is built to real-timely monitor the operation status, fault early warning and log analysis of intelligent terminals. 24/7 fault troubleshooting and repair services are provided to ensure the stable operation of terminals. Batch upgrade and patch update services are offered to timely fix vulnerabilities and optimize performance.

4.2.3 Late value-added services

Functional iterative upgrading services are launched to continuously optimize software functions according to technological development and user demands. Data analysis services are provided to tap the value of data collected by terminals and support user decision-making. Services such as technical training, after-sales customer service and secondary development support are available. For high-end customers, personalized value-added services such as exclusive customization and one-on-one operation and maintenance are provided to expand profit space.

4.3 Innovation of Commercial Service Models

In terms of service models, a three-level diversified service system is constructed. First, the subscription service model is implemented, breaking the traditional one-time charging model. Three types of packages (basic, advanced and premium) are launched, supporting monthly or annual payment. Users can enjoy services such as operation and maintenance guarantee, functional iteration and security protection, and can flexibly upgrade or downgrade to adapt to different user demands, stabilizing the corporate cash flow. Second, the pay-as-you-go model is adopted. For temporary

and personalized demands, charging is based on the number of times or functions, covering services such as fault repair, functional customization and data desensitization, meeting niche and fragmented scenarios and improving service flexibility. Third, the ecological service model is created. An industrial ecosystem is built in collaboration with multiple stakeholders such as chip, hardware and application developers. Interfaces and components are opened to provide technical support, adaptation testing, operation and promotion services for partners, realizing resource sharing and mutual benefit, and expanding service boundaries.

Meanwhile, a service standardization and management system is established, formulating service process, quality and response specifications. Assessment is carried out with customer satisfaction, fault response time and problem solving rate as the core indicators. Relying on the digital service platform, the full-process management and control of orders, work orders and feedback is realized to comprehensively improve service efficiency and transparency.

5 EMPIRICAL CASE ANALYSIS

5.1 Case Overview

An embedded software upgrading project of smart door locks of a smart home enterprise is selected as the case. The enterprise's original smart door locks adopted traditional monolithic embedded software with a high degree of hardware-software coupling, making it impossible to realize functions such as remote unlocking, fingerprint algorithm upgrading and abnormal alarm in the later stage. The service model only provided 1 year of free maintenance, resulting in a high customer complaint rate and low repurchase rate. This project focuses on solution reconstruction and service-oriented transformation, optimizes the software architecture, innovates the service model, and improves product competitiveness.

5.2 Implementation of Solution Reconstruction

Technically, the lightweight layered service architecture is adopted to split microservice components such as fingerprint recognition, communication transmission, door lock control and security encryption, and the Hardware Abstraction Layer is introduced to realize hardware-software decoupling. A cloud-edge collaboration platform is built to realize cloud-side remote management and control and terminal-side real-time execution, supporting remote upgrading and abnormal alarm. The development process is optimized with the agile development model, shortening the development cycle from 3 months to 1.5 months. In terms of security, secure boot, data encryption and anti-cracking modules are added in compliance with smart home security standards.

5.3 Implementation of Service-oriented Transformation

A full-life-cycle service system is constructed, providing personalized functional customization services in the early stage, launching remote operation and maintenance and fault troubleshooting services in the mid-term, and offering value-added services such as fingerprint algorithm upgrading and function expansion in the late stage. The commercial model adopts "hardware sales + annual subscription service", where the subscription service includes remote upgrading, lifetime maintenance, security protection and other contents.

5.4 Evaluation of Implementation Effects

After the project implementation, the software reusability is increased by 60%, the development and iteration cost is reduced by 45%, and the customer complaint rate is decreased by 70%. The proportion of subscription service revenue is increased from 0 to 25%, and the customer repurchase rate is raised by 35%. The functional scalability and security of the product are significantly improved, and the market competitiveness is greatly enhanced, which verifies the feasibility and benefits of the reconstruction of embedded software solutions and service-oriented transformation.

6 TRANSFORMATION SAFEGUARD MEASURES AND OPTIMIZATION SUGGESTIONS

6.1 Technical Safeguard Measures

Increase R&D investment in core technologies such as lightweight architecture, cloud-edge collaboration and security encryption to overcome the technical challenges of service-oriented transformation in resource-constrained scenarios; build a standardized development platform and component library to improve software development efficiency; improve simulation testing and remote operation and maintenance tools to consolidate the technical support system; keep up with technological trends such as RISC-V and edge AI to continuously optimize software solutions.

6.2 Management Safeguard Measures

Transform the enterprise business philosophy, establish a development thinking of "taking service as the core", and set up a professional service team; improve the service management system and assessment mechanism to enhance the professional quality of service personnel; establish a closed-loop customer feedback mechanism to continuously optimize service content and quality; promote digital management, build a service management and control platform,

and realize the visualization and high efficiency of service processes.

6.3 Ecological Safeguard Measures

Strengthen industrial chain collaboration, and build a symbiotic and win-win industrial ecosystem in cooperation with partners such as chip manufacturers, hardware suppliers and operators; participate in the formulation of industry standards to promote the standardization of embedded software interfaces and service specifications; build an open service platform to attract third-party developers to settle in and expand service scenarios and ecological boundaries.

6.4 Risk Prevention and Control Measures

Establish prevention and control mechanisms for technical, market and security risks to predict the bottlenecks and problems in the transformation process in advance; strengthen compliance management to ensure that software security and data privacy meet regulatory requirements; formulate differentiated response strategies for problems such as customer churn and profit fluctuations after service-oriented transformation to ensure the steady transformation of enterprises.

7 CONCLUSIONS AND PROSPECTS

Aiming at the core pain points of embedded software for intelligent terminals such as architectural coupling, inefficient development and lagging services, this paper proposes a systematic path for solution reconstruction and service-oriented transformation. Technically, it breaks the bottlenecks of the traditional model relying on lightweight layered architecture, hardware-software decoupling, cloud-edge collaboration and other technologies, and optimizes the efficiency of software development and deployment. In terms of services, it builds a full-life-cycle service system and innovates commercial models such as subscription and pay-as-you-go, promoting the industrial transformation from product delivery to value-added service. Empirically tested with smart home terminals, this path can effectively reduce costs and increase efficiency, improve customer stickiness and corporate profit space, and is fully adapted to the development needs of the intelligent terminal industry.

In the future, with the in-depth penetration of artificial intelligence and the Internet of Everything technologies, the embedded software of intelligent terminals will advance towards lightweight, intelligent and ecological directions, and the service-oriented business format will be fully popularized. Follow-up research can further delve into the directions of AI-enabled adaptive services, cross-terminal ecological collaboration and lightweight architecture optimization, continuously improve the theoretical system and practical solutions for the service-oriented transformation of embedded software, and provide more solid support for the high-quality development of the intelligent terminal industry.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Ye L J. Research on Configuration Management and Anti-tampering Scheme of Embedded Software for Program Group. Computer and Information Technology, 2025, 33(06): 110-114.

[2] Fu Z Z. Research on Testing Technology and Security Verification of Automotive Embedded Software. Auto Electric Parts, 2025(07): 129-131.

[3] Chen X Q. Design and Implementation of Heterogeneous Security Control System Based on Embedded Technology. Electronic Technology, 2025, 54(08): 86-88.

[4] Du S B, Sun G L, Chen X P, et al. Design and Implementation of Smart Home System Based on the Internet of Things. Automation Application, 2025, 66(13): 208-212.

[5] Johansson N. Integration of Service-Oriented Embedded Systems with External Systems in Software Product Lines. 2024.

[6] Aziz M W, Ullah N, Rashid M. A process model for service-oriented development of embedded software systems. IT Professional, 2021, 23(5): 44-49.

[7] Olsson H H, Bosch J. Going digital: disruption and transformation in software-intensive embedded systems ecosystems. Journal of Software: Evolution and Process, 2020, 32(6): e2249.

[8] Blanco D F, Le Mouël F, Lin T, et al. A comprehensive survey on Software as a Service (SaaS) transformation for the automotive systems. IEEE Access, 2023, 11: 73688-73753.

[9] Chamari L, Petrova E, Pauwels P. An end-to-end implementation of a service-oriented architecture for data-driven smart buildings. IEEE Access, 2023, 11: 117261-117281.

[10] Yangui S, Goscinski A, Drira K, et al. Future generation of service-oriented computing systems. Future Generation Computer Systems, 2021, 118: 252-256.