Design and Implementation of Multi-terminal Debugging System based on SSM Framework

Jun Ma\textsuperscript{1,a}, Jun Mao\textsuperscript{2,b}, Long Chen\textsuperscript{3,c} and Lili Yin\textsuperscript{4,d}

\textsuperscript{1}School of Dalian, JiaoTong University, Dalian 116021, China;
\textsuperscript{2}School of Dalian, Science and Technology University, Dalian 116021, China;
\textsuperscript{3}CRRC Dalian Locomotive Co., Ltd., Dalian 116021, China;
\textsuperscript{4}Dalian Metro Operation Co., Ltd., Dalian 116021, China.
\textsuperscript{a}majun_jdyz@163.com, \textsuperscript{b}maojungaga@126.com, \textsuperscript{c}chen.long0609@163.com, \textsuperscript{d}283343421@qq.com

Abstract

The existing bicycle simulation debugging system has the disadvantages of fixed debugging location and low debugging efficiency. Through analyzing the function requirements of the debugging system, a multi-terminal debugging system with convenient mobility, high stability and good scalability is designed, including the simulation of bicycles. The debugging system performs real-time communication, database management, user login, monitoring all data points and other functions. The test shows that the multi-terminal debugging system can establish a stable connection with the bicycle simulation debugging system, which effectively reduces the debugging burden of the operator and improves the debugging efficiency.

Keywords

Multi-terminal Debugging; SSM; Socket.

1. Introduction

In the production process of Chinese standard EMU trains, debugging is of great significance as the final process. The debugging process can eliminate potential failures before the train leaves the factory and provide guarantee for the safety of train operation. Digital debugging in the debugging process can greatly improve the quality and efficiency of the process. The existing bicycle simulation debugging system is that the debugger monitors and controls the network signal in the vehicle by operating specific data points on the bicycle simulation test bench, such as monitoring and controlling the status of the actual components in the vehicle through feedback data points. The operation control data point sends control commands to change the actions of the air switch and relays in the vehicle. This kind of work mode requires at least two people to be able to safely and accurately carry out. If only one person performs debugging work, it is necessary to turn back and observe between the console and the electrical cabinet. The workload is relatively large and the debugging efficiency is low. Therefore, a multi-terminal debugging system based on the SSM framework has been developed and designed. This system enables multiple people to debug the same train at the same time. During the debugging process, the operator can move at will, which is convenient for observing whether the devices on the train are operating normally and the communication speed is fast. The operation process is simplified, and the on-site debugging efficiency and safety are improved.
2. Debug system function analysis

2.1 Data-driven functional analysis

The original bicycle simulation debugging system is developed using Java language, and the real-time Ethernet driver of the train is driven by the MVB network card to realize the data communication between the original bicycle simulation debugging system and the MVB bus or ECN bus. Since the server side of the multi-terminal debugging system cannot directly obtain MVB or TRDP data, it is necessary to develop a data-driven layer to realize data communication. Its functions are as follows:

(1) Update the original bicycle simulation debugging system function, add Socket communication interface, monitor the connection request of the client of the multi-terminal debugging system as a server, and establish a Socket communication connection;

(2) Through a custom communication protocol, encapsulate the MVB or TRDP data packets in the original debugging system, and transmit them to the multi-terminal server in real time via the Socket interface;

(3) Receive and analyze data packets sent by multiple terminal servers, and then transmit them to the MVB or TRDP driver.

2.2 Server-side function analysis

The server is built using the SSM framework set, which mainly realizes the interaction with the database and establishes a Socket communication connection with the server of the original bicycle simulation debugging system as the client. Its functions are as follows:

(1) Receive the login form submitted by the browser-side user, and compare it with the operator information stored in the background database. If the user name and password are legal, you can jump to the main interface of the operation, and if they are illegal, you will be prompted for the corresponding error;

(2) Extract the debugging information stored in the database, and generate a three-level menu directory with three types of information "platform", "model" and "test system" for the operator to choose;

(3) According to the test system selected by the user, return the test module and test data point information under the current system;

(4) When the status of the data point on the browser side changes, it is sent by the server to the bicycle simulation debugging system.

2.3 Browser function analysis

(1) Realize that the operator enters the work number and password to log in;

(2) Open and operate the operation interface through the hierarchical tree menu;

(3) Distribute reasonable control and feedback data points for operators to monitor.

3. Debugging system software design

The development framework of the multi-terminal debugging system is B/S mode. The software uses the eclipse development integrated environment, MySQL database management system, and uses Maven to build and manage the entire project. The technical means of front-end and back-end separation are used to build the project. It is set by the SSM framework. Hierarchical management of the back-end functions, combined with the layUI front-end framework for engineering construction. SSM (SpringMVC+Spring+MyBatis) framework is a technical framework integrated by three technical frameworks: Spring, Spring MVC, and MyBatis. It divides the entire software into a presentation layer, a controller layer, a service layer and a dao layer. It uses Spring to implement business object management, SpringMVC is responsible for request forwarding and view management, and MyBatis is used as a persistence engine for data objects. layUI is a front-end UI framework written with its own module specifications. It follows the writing and organization of
native HTML/CSS/JS and provides many beautiful and convenient styles. The biggest advantage of this framework is that it defines many front-end interaction styles. Interface, you only need to configure the interface on the front end, and the back end will return data according to the defined interface rules to complete the page display. According to requirements, the overall structure of the software is divided into three parts: communication driver, database management and debugging system interface. The overall software structure is shown in Figure 1.

![Software overall architecture diagram](image)

**Figure 1. Software overall architecture diagram**

### 3.1 Communication drive module

The communication driver adopts Socket communication technology to realize data interaction, including two modules: Socket communication on the server side of the original bicycle debugging system and Socket communication on the client side of the multi-terminal debugging system. The detailed design is carried out as follows.

![Server-side data-driven main thread execution flow chart](image)

**Figure 2. Server-side data-driven main thread execution flow chart**
3.1.1 Socket communication on the server side of the original bicycle debugging system

Create a new multi-terminal communication class under the original bicycle debugging system communication driver package. In this class, the program will implement Map encapsulation of MVB or TRDP data and pass it to the multi-terminal debugging system client, and can read and parse the client sent Packets. The custom start() method in this class, when the start() method is called, a new thread is created, and the Socket interface is established in the thread to monitor the client's connection request. When the Socket interface is connected, two sub-threads of "read" and "write" are created. In the "write" sub-thread, the data driver processes the MVB or TRDP data and sends it to the multi-terminal system. According to the "read" sub-thread The user-defined communication protocol acquires the data, reforms it into Map<String,byte[]> type data, and sends it to the vehicle bus for data transmission and reception. The server-side data-driven main thread execution flow chart is shown in Figure 2, and the execution flow diagrams of the "write" and "read" sub-threads are shown in Figures 3 and 4, respectively.

![Figure 3. "Write" sub-thread execution flow chart](image1)

![Figure 4. "Read" sub-thread execution flow chart](image2)
3.1.2 Multi-terminal debugging system client Socket communication

The data obtained by the client of the multi-terminal debugging system is encapsulated MVB or TRDP data, which needs to be parsed according to a custom communication protocol, and the parsed data is logically processed and displayed in real-time on the debugging operation interface, and the bicycle can be simulated and debugged through this interface. The system sends data packets whose status has changed. The execution flow chart of the main thread driven by the client data is shown in Figure 5.

Figure 5. Client data-driven main thread execution flow chart
3.2 Database management module
Multi-terminal debugging system uses MySQL database to manage all data information, including user information table and monitoring data point information table.

3.2.1 User information table
The user information table stores user name, password, and authority information, which is used as the basis for verifying user login information. When a new user or user authority needs to be added, changes can be made in the user information table to improve the flexibility of the debugging system.

3.2.2 Monitoring data point information table
For on-site debugging personnel to conduct statistics and sorting, the monitoring data point information is divided into three data tables: A, B, and C. Table A stores the platform, vehicle type and test system. The three-level tree menu of the debugging operation interface is based on the information of Table A. To generate. Table B stores vehicle models, test systems, test modules, locations and operating instructions. Table C stores location, type, port, word offset and position offset information. When the commissioning personnel clicks on the tree menu to complete the test system selection, follow B and C The table information queries all corresponding test modules and control and feedback operation data points, and feeds it back to the browser to generate the corresponding operation interface.

3.3 Debug system interface
The debugging system interface uses the layUI front-end framework to draw, including the user login interface and the debugging monitoring interface.

3.3.1 User login interface
After starting the multi-terminal debugging system software, enter the login interface, the operator enters the user ID and password in turn, clicks the login button, and submits the login application to the background. First, judge whether the content of the user ID and password input box is empty, if it is empty, proceed Corresponding prompts, if the format of the user ID and password are correct, they will be submitted to the background for comparison with the user information in the database, and if they match, they will jump to the main interface of the debugging system.

3.3.2 Debug monitoring interface
The debugging monitoring interface is mainly composed of a tree-shaped menu and an operation part. The function of the tree-shaped menu is to assist the operator to quickly find the vehicle model and test system that needs to be debugged. The operation part is the core of the multi-terminal debugging system. It needs to complete the data point status display, Numerical display, data point status change and other operations. Partitioned by test modules, the data points in each area are arranged in order according to the type, followed by "DI" data point, "DO" data point, and "AI" data point. Each data point includes data type, data value, and Information description. We define the “DI” type data point gray to indicate the disconnected “0” state, green indicates the on-state “1” state, the internal yellow of the “DO” type data point indicates the disconnected “0” state and is operable, and the green outer frame indicates the conduction. Through the "1" state, the "AI" type data points need to be displayed in decimal after the analog data is processed by the formula.

4. Debug system function test
In order to verify the function of the debugging system, a test platform was built in the laboratory, and DO data was normally sent and received between the bicycle simulation test bench and the multi-terminal tablet. Use the test bench and pad to build the test environment, and display the test results through the intuitive changes of the interface. Connect all equipment to the laboratory LAN, run the bicycle simulation debugging software in the bicycle simulation test bed, open the debugging software and start the multi-terminal debugging, select the model and system to enter the corresponding operation interface, and open the multi-terminal debugging software on the multi-
terminal pad. Enter the job number and password to enter the debugging interface, take the interior lighting system under the MB05 car as an example to verify the function. Change the state of the first "DO" data point in the pad operation interface, and change the state of the second "DO" data point in the operation interface of the bicycle simulation test bench. The final result can be observed that the states of all the "DO" data points of the two devices are all Consistent, in order to simulate the process of data receiving and sending between the multi-terminal debugging system and the bicycle simulation debugging system, the laboratory function test is shown in Figure 6.

![Debug system laboratory function test chart](image)

**Figure 6. Debug system laboratory function test chart**

### 5. Conclusion

Aiming at the problem that the existing bicycle simulation debugging system cannot provide portable operation, combined with the analysis of the function requirements of the debugging system, this paper designs a multi-terminal debugging software based on the SSM framework from three aspects: the communication drive module, the database management module and the debugging system interface. The function has been tested to verify that the communication has high reliability and meet the needs of on-site debugging. The multi-terminal debugging system has been put into use in the main engine factory, which obviously improves the debugging efficiency, reduces the work intensity of the debugging personnel, and the application feedback effect is good.

**References**


