Design and Implement of Automatic Test Environment for ETCS On-board Equipment

Xue Renpu1*, Geng Hongliang2, Ke Changbo3, Yang Kai4
1,2,3,4 Hunan CRRC Times Signal & Communication Co., LTD Beijing Branch
*Corresponding author's email: xue_renpu@sina.com

Abstract. ETCS on-board equipment is a typical safety-critical system, its reliability plays a vital role in the safe driving of trains. ETCS on-board equipment has huge functions, complicated structure, a large number of test cases and difficult implementation. Introducing automated testing into on-board equipment testing is a problem that needs to be solved urgently. Based on the interface and functional characteristics of ETCS on-board equipment, this paper adds the design of test management unit, the division of test engine and the separation of test scripts based on the existing test environment, and develops a set of simple and efficient ETCS on-board equipment dedicated automation test environment. The application shows that this set of test environment realizes the automatic test of all test sequences of ETCS on-board equipment, which can significantly improve the test efficiency, reduce the tester's work intensity, and improve the test accuracy.

1. Introduction
ETCS on-board equipment is the core control equipment installed on the train. Any defects in function and performance will bring hidden dangers to high-speed trains, and even lead to the result of endangering the safety of driving. It is extremely important to conduct strict and comprehensive tests on it.

At present, the test for on-board equipment generally uses manual testing in an integrated test environment. The cost of building an integrated test environment is high, and manual inspection, recording, and log analysis are time-consuming and inefficient. Besides, due to the huge functions and complex structure of on-board equipment, it requires a high professional quality of testers, but it is still inevitable to introduce human-induced errors and omissions, which cannot guarantee high test quality [1]. Therefore, it is a recognized research direction in the industry to introduce automated testing technology into on-board equipment testing. Some design schemes are also given in existing studies. References [2] proposed an automatic test platform for the train control system composed of real equipment, simulation model and real operation line data, to automate testing of on-board equipment in an integrated test environment. Some design schemes are also given in existing studies. References [3] realized a special automated test for LKJ on-board equipment through a test script-driven simulation model based on C# language. However, the use of C# language for test scripts affects reusability, maintainability, and expansibility of the test environment, and belongs to the previous modular testing framework. Keyword drive based on test script language has developed into a mainstream framework. References [4-5] have designed automatic test systems for CTCS-2
on-board equipment. But for ETCS on-board equipment, it is necessary to add JRU, RBC and STM interface adaptation, and also put forward higher requirements for the test framework.

The European Railway Administration issued the SubSet-026 System Requirements Specification [5] and SubSet-076 system test specification [6-7] for ETCS on-board equipment. To fully cover the requirements, in addition to the 692 mandatory test sequences included in SubSet-076-6-3, 606 additional test sequences have been added, for a total of 1298 test sequences. To support the testing of the above test sequence, it has become a consensus in the industry to develop a special automated test environment for on-board equipment.

Based on the existing research results, the following problems also need to be solved: (1) Design a simple and efficient automatic test framework to achieve full-function testing of ETCS on-board equipment, (2) Adapt to a variety of different interfaces of on-board equipment to achieve closed-loop control, (3) Independently maintain transponder messages and radio messages to achieve separation of data and logic.

In this paper, by designing a message-level closed-loop control scheme for each interface, by optimizing the test script structure and test execution process, and by adding functions such as test exception handling, a set of dedicated automation test environment for ETCS on-board equipment is designed and implemented.

2. Overall design

2.1 Design principles

First, the safety integrity level of ETCS on-board equipment reaches the SIL4 level. The standard requires that all tools used in the life cycle of safety software must not cause the incorrect output of safety software due to tool problems [8]. The standard divides tools into T1, T2 and T3 levels. The automated test environment of on-board equipment researched and developed in this paper belongs to T2 tools, that is, the failure of the test environment will not directly cause software errors, but will fail to check for software defects. Therefore, a simple architectural design is required. The logic part is limited to the test script written by the tester, so that the test execution part avoids the existence of complex logic as much as possible, thereby reducing the probability of introducing errors in the test environment. This is also the basic requirement for independent verification of the test environment during safety certification.

Second, the interface settings of ETCS on-board equipment are more complicated, mainly including 6 different external interfaces: Driver interface, Vehicle interface, Balise interface, NTC interface, RBC interface, and JRU interface. According to the European standard EN50128, the black box function test is a mandatory test method [9]. Therefore, the test environment must adopt an interface-oriented design method. All test environment input and output of on-board equipment need to be implemented through these interfaces. Message-level control of all interface data is required, and the timing of data transmission and reception must be precisely controlled. To realize the function of the on-board equipment can be tested completely and accurately. This is also the biggest difference between the special test environment and the integrated test environment.

Third, the SubSet-076 test sequence is described in units of interfaces. Therefore, connecting the on-board equipment through the test sub-engine can simulate the real interface connection. Use object-oriented thinking, set up the same type and number of sub-engines; at the same time, the sub-engines should strictly follow the true connection relationship, accept the main engine scheduling, and cannot communicate with each other. This makes it easier for test environment developers and testers to understand how the test environment works, and it also simplifies scriptwriting and debugging to quickly complete the conversion from test sequence to test script.

Fourth, due to a large number of test sequences, in the design of test scripts, it is necessary to set up a general process script for invocation and separate test data from test logic, thereby improving script simplicity and maintainability. At the same time, the stability of the batch execution of test scripts requires additional design. Since the Test Main Engine mainly focuses on the single execution process,
some functions such as test preparation, exception handling, and environment recovery are difficult to achieve. Therefore, it is necessary to add a Test Manager Unit on top of the Test Main Engine to achieve 24-hour unattended testing.

2.2 Design of the system architecture

Whether the test environment can be effectively carried out depends on the quality of its system architecture design. The test environment needs to effectively organize the necessary functions such as test management, test execution, logging and report generation. The test system architecture constructed in this paper adopts a layered design idea and divides the test environment into the Test Script Layer, the Test Management Layer, the Test Drive Layer, and the Test Interface Layer. The purpose is to layer all functions and limit the functions of the same type to a fixed layer. The layers depend on each other and call each other, to achieve high cohesion and low coupling, make the test framework structure simpler and more efficient, and have better maintainability.

According to the interface and functional characteristics of ETCS on-board equipment, combined with the design principles of the automated test framework, the implementation of the test environment in this paper is that the script drives the test engine to exchange messages with the on-board equipment. The test environment consists of the Test Script, the Test Manager, the Test Main-engine, the Test Sub-engine and the Vehicle Interface Unit. The test environment architecture and data flow are shown in Figure 1.
3. Detailed design

3.1 Design of Test Script
The Test Script is a computer language description of the test sequence. The system uses the interactive script language Expect to write test scripts. Expect language is a string-based interpreted language, which shortens the editing-compile-link-run process of traditional languages, and can directly interpret and execute each statement through the language interpreter.

Each test sequence specifies the test input and expected output in steps, and gives the transponder and wireless message data required by the sequence in the form of a table. To be more concise and easy to maintain, the test script is split into the Logic Script, the Common Process Script and the Data Script. (1) The Logic Script converts all the test input and test output of the test sequence into script commands and script expectations, which can completely correspond to the test sequence; (2) The Common Process Script is to extract the standardized scenes or common function steps repeated in 1298 test sequences into functions, and be called by the Logic Scripts by keyword invocation; (3) The Data Script is directly generated by the data generation tool in batches, which realizes the separation of logic and data, and can replace the test data without modifying the script logic.

3.2 Design of Test Manager Unit
The Test Management Unit implements the unified allocation and control of test resources. Its main functions include parameter configuration, engine startup, Execution list settings, script issuance, exception handling, log summary, and test report generation. The Test Management Unit can improve the operability and stability of the entire test process.

As various abnormalities inevitably occur during the test, if the interruption of the test results in manual intervention, it not only increases the work intensity of the tester but also affects the continuity and credibility of the test. To support the continuous running of thousands of test scripts, 24 hours of stable operation of the test environment under unattended conditions is a basic requirement. However, as a test execution module, The Test Engine cannot handle some exceptions at all. Therefore, the Test Management Unit must have exception handling functions, including engine guarding, execution monitoring, power control, and script retesting.

- The implementation of the engine guarding is that the Test Management Unit is responsible for the startup, monitoring and shutdown of all Test Engines. When it is detected that any Test Engine exits abnormally, it is judged that the current script execution is abnormal, all Test Engines are destroyed, and the next script is executed after the environment is automatically restored.

- The implementation of execution monitoring is to set the maximum waiting time for each test script. When the timeout is detected, the execution of the script is judged to be abnormal, and the next script is executed after the environment is automatically restored.

- The implementation of power control is to automatically issue the power off command of the on-board equipment before the execution of the first script when an engine abnormality or script execution failure is detected. It can be avoided that the vehicle-mounted device is still in the power-on state and subsequent script execution fails.

- The script retest is implemented by retesting the abnormal script and the failed script one by one after all the scripts in the test list are executed, helping the tester to analyze and evaluate the test results more accurately.

In addition, the test management unit provides a man-machine interface for the test environment. Testers can control the test process through the interface and view the test progress and the executed test results in real-time.

3.3 Design of Test Main Engine
The Test Main Engine is the control center of the test execution process and an important part of the automated test process. The Test Main-Engine is responsible for parsing the test script and sending the test commands in the script to the Test Sub-Engine. And responsible for receiving the test feedback
sent by the Test Sub-engine, comparing the test feedback with the expectations in the script to generate test results. Then send the test results and test records to the Test Management Unit. The Main-Engine consists of two parts: The Script Controller Module and The Main-Engine Kernel. The structure of Test Main Engine is shown in Figure 2.

The Script Controller Module is responsible for parsing script commands, starting the Main-engine Kernel, and controlling the progress of tests. The Main-engine Kernel is responsible for the communication between the Script Controller Module and the Sub-engine. It packs the commands sent by the Script Control Module to the Sub-engine according to the command string format, parses the messages returned by the Sub-engine into strings, and sends them back to the Script Control Module. The message packets communicated between the Main-engine and the Sub-engine are encoded and decoded using Protocol Buffer. First store the defined data structure in the Proto file, and then compile to generate the Protobuf stream format. The advantage is that the data is processed and stored in a structured way, which can realize the serialization of the data.

3.4 Design of Test Sub-Engine
The Test Sub-engine is responsible for the data interaction between the test environment and each interface of the on-board equipment. ETCS on-board equipment has 6 different interfaces, and the connection methods and communication protocols of each interface are different. Therefore, to simulate the real interface connection situation of on-board equipment, RBC Sub-engine, Vehicle Sub-engine, Transponder Sub-engine, STM Sub-engine, JRU Sub-engine, and DMI Sub-engine are designed to realize interface testing.

Each Sub-engine is uniformly scheduled by the Main-engine and uses the basic structure of the Communication Layer, the Protocol Data processing Layer, and the Application processing layer, but the Sub-engines are independent of each other and do not communicate with each other. The main function of the Sub-engine is to encode or decode the received content and forward it. When the Sub-engine receives the excitation string of the Main-engine, it is converted according to the real data format between the different Sub-engines and the tested interface and sent to the on-board equipment. When the Sub-engine is responsible for receiving the feedback of the tested interface, it is parsed according to the protocol and then converted into the Protobuf stream format string and sent to the Main-engine.

3.4.1 Design of Vehicle Sub-Engine
Vehicle Sub-engines adapt to vehicle interfaces, including I/O, MVB and SFI interfaces. When the Vehicle Sub-engine interacts with the on-board equipment, the interface conversion is performed
through The Vehicle Interface Unit. The I/O and MVB protocol messages only need 100ms as the cycle, while the SFI speed data uses the 20ms transmission cycle.

The dynamic model calculation also belongs to the main function of the Vehicle Sub-engine. In order to facilitate script control, the steering wheel handle is no longer retained, and the running direction, acceleration and target speed are directly set by the script. The acceleration model uses the acceleration set by the script. The deceleration model queries the pre-configured "speed-deceleration" table. The constant model automatically sets the constant speed after reaching the target speed.

3.4.2 Design of DMI Sub-Engine

The DMI Sub-engine is adapted to the Driver interface. To achieve closed-loop control, the on-board DMI device needs to increase the interface with the test environment and design a custom transmission protocol. In particular, the program related to the test interface should also be released and certified as part of the DMI software to ensure the effectiveness of the test. The DMI Sub-engine sends key operations to the DMI instead of sending logical instructions to simulate real driver key operations. The on-board DMI device forwards the original information packet is received from the on-board host to the DMI sub-engine. The transmission protocol preferentially uses the original protocol between DMI and the on-board host device to reduce the modification of DMI software.

3.4.3 Design of RBC Sub-Engine

The RBC Sub-engine is adapted to the RBC interface. The RBC Sub-engine connects to the MT Emulator in the Vehicle Interface Unit to realize radio message interaction with on-board equipment. The on-board equipment supports the simultaneous establishment of two ground RBC equipment, so at least three RBC Sub-engines need to be configured for functional testing such as RBC handover.

The communication between RBC and on-board equipment adopts the RSSP-II safety communication transmission protocol, it also belongs to a SIL4 safety software product. Therefore, the difficulty of the RBC Sub-engine is that the RSSP-II protocol library that has been independently certified by a third party must be integrated before it can be used in the test environment of ETCS on-board equipment.

3.4.4 Design of Balise Sub-Engine

The Balise Sub-engine adapts to the Balise interface, receives the Balise command from the Main-engine and generates a Balise telegram protocol data packet, and then sends it to the Vehicle Interface Unit in advance. To reduce the delay, the Vehicle Interface Unit determines the timing of the Balise telegram transmission based on the location, speed, and acceleration sent by the Vehicle Sub-engine. When the train passes the Balise, the corresponding Balise telegram is sent to the on-board BTM device, and then the BTM is sent to the on-board host.

3.4.5 Design of STM Sub-Engine

The STM Sub-engine is adapted to the NTC interface. The physical interface uses the Profibus bus, so the interface conversion needs to be performed through the Vehicle Interface Unit. The function of the STM Sub-engine is to convert the script command of the Main-engine into STM protocol message and send it to the Vehicle Interface Unit, and unpack the STM protocol message received from the on-board equipment into Protobuf stream format string and feed it back to the Main-engine. ETCS on-board equipment supports the simultaneous connection of multiple NTC systems, so multiple STM Sub-engines can be configured so that on-board equipment can switch between different NTC systems for testing.

3.4.6 Design of JRU Sub-Engine

The JRU Sub-engine is adapted to the JRU interface and receives maintenance diagnostic data of the on-board JRU equipment. The function of the JRU Sub-engine is to convert the script command of the Main-engine into JRU protocol data and send it to the JRU device and convert the JRU protocol data
received from the on-board equipment into Protobuf stream format string and feed it back to the Main-engine.

3.5 Design of Vehicle Interface Unit
The Vehicle Interface, the Balise Interface, the NTC interface and the RBC interface of the ETCS on-board equipment interface use Electrical Interfaces or Bus Interfaces, which cannot be directly connected to the Test Sub-engine. Therefore, it is necessary to design the Vehicle Interface Unit, adapt all the physical interfaces of the on-board equipment from the hardware, and implement the interface communication protocol from the software to assist the Sub-engine connection with the on-board equipment.

In addition to the above-mentioned interface functions, the vehicle-mounted interface unit also has functions such as MT simulation, SFI frame custom period adaptation, Balise telegram buffering and transmission timing judgment.

4. Conclusion
This paper analyzes the characteristics and difficulties of introducing automated testing into ETCS on-board equipment testing. A set of script-driven dedicated automated test environment for ETCS on-board equipment is designed and implemented. It can perform message-level closed-loop control on all interfaces of ETCS on-board equipment. The layered design of the system architecture achieves high cohesion and low coupling of the test environment, and the separation design of scripts improves the simplicity and maintainability of test scripts. By adding a test management unit, centralized management of test resources and exception handling are achieved.

The test environment implemented in this paper is continuously optimized and perfected in the long-term practice. It supports all the development, debugging, testing and verification of the ETCS on-board equipment R&D project. During the project, a total of 4 test environments were built, more than 40 rounds of system tests were completed, and more than 70,000 test scripts were executed. The application shows that this set of test environments can improve the test efficiency and accuracy, and at the same time ensure the large-scale and stable operation of test scripts, and have achieved good results in reducing labor costs, time costs.

References
[1] He Guangyu, Fan Ming, Cheng Jianfeng, Huang Hongfu. The Automatic Testing Platform of CTCS-3 level Train Control System Base on AdmiTest[J]. China Railway Science,2013,34(04):128-136.
[2] He Xiezhen, Wang Yeliu, Zhou Zhifei. Key Technologies of Automatic Test System for Train-control Vehicle-borne Software[J]. Electric Drive for Locomotives,2014(05):89-91.
[3] Zhang You-bing, Zhang Bo. Research and Implementation of the Automatic CTCS-2 ATP Test Platform Based on Tool Command Language[J]. Journal of Lanzhou Jiaotong University,2013,32(03):86-93.
[4] Luo Feibao, Li Yinan. Design and Implementation of CTCS-2 Level Train Control on-board Equipment Automatic Simulation Test Platform[J]. Railway Signalling & Communication,2018,54(04):75-79.
[5] ERTMS/ETCS. SUBSET-026. System Requirements Specification. v3.6.0[S]. ERTMS. 2016.
[6] ERTMS/ETCS SUBSET-076-5-2: Test Case. v3.1.0[S]. ERTMS. 2016.
[7] ERTMS/ETCS SUBSET-076-6-3: Test Sequences. v3.0.0[S]. ERTMS. 2016.
[8] Qiu Zhaoyang. Research on Specifications for Safety-Related Tool in Railway Signal Products[J]. Railway Signalling & Communication Engineering,2019,16(07):9-13.
[9] CELELEC.EN50128 Railway Applications-Communication,Signalling and Processing Systems-Software for Railway Control and Protection Systems[S].Brussels: CENELEC, 2011.
[10] Martin Johne, Martin Busse. RailSiTe®(Rail Simulation and Testing)[J]. Journal of large-scale
research facilities, 2016, 2, A88.

[11] Yang Kai, Ke Changbo, Xue Renpu. Design and Implementation of the Interface Unit in Onboard ATP Simulation Test Environment[J]. Control and Information Technology, 2019(03):52-55.

[12] Gregorio Barberio, Aniello Amato. An Interoperable Testing Environment for ERTMS/ETCS Control Systems[C]. Lecture Notes in Computer Science. Springer Verlag, 2014:147-156.