An Automated Testing Method for UDS Protocol Stack of Vehicles

Yu Jinghua[1, a], Luo Feng[1, b]

[1]Clean Energy Automotive Engineering Center, Tongji University

No. 4800 Cao An Road, Shanghai, China

[a]yujinghua@outlook.com, [b] luo_feng@tongji.edu.cn

Keywords: Diagnostic Protocol Stack, Orthogonal Design Method, ODX Database, Automated Testing.

Abstract. An automated testing method for vehicles' UDS (Unified Diagnostic Service) protocol stack [1] based on CAN bus is presented in this article. According to different diagnostic services, which are specified in ISO 14229-1, a strategy of generating test cases is designed by using a comprehensive method, which is mainly based on orthogonal design method. Then a test tool is developed in Delphi development environment. It can configure test parameters, import standard ODX (Open Diagnostic Data Exchange) database files and generate test scripts automatically. Finally, these scripts can be executed in another test environment directly and get test reports automatically. During the whole test process, users do not need to study the details of the test specification and write test scripts, or do the test steps manually. They can test the target stack just by configuring test tools and click 'Execute'. This automated test method can improve the efficiency of this UDS protocol stack test, and reduce the workload of developers.

Introduction

Diagnostic technology is an essential part in vehicle industry. Through communication between diagnostic interface of vehicle ECUs and diagnostic tester, the working situation and failure records of ECUs inside can be monitored and obtained.

In order to ensure the correctness and standardization of a UDS diagnostic communication, the functional test of a developed protocol stack plays an important role during the development. There are several kinds of ways to do this test, which are as follows. The first one is to test all the functions manually. According to the test specification, developer can use bus communication tool to set and send diagnostic message one by one, and then check the response message to judge whether the data is correct. This way is direct and inefficiently. It is just for the simple test at very beginning. The second way is to develop some simple PC applications, which are used as diagnostic testers and communicate with the target devices. [2] These applications are used in specific development project and cannot be used generally. The developer also need to spend lots of time studying the protocol specification and writing test codes. Besides the tests are still operated manually. The third way is to use professional test tools from the company called Vector Informatik. [3] Diagnostic database is made by CANdela or ODXStudio. Test specification and scripts are generated by CAN Option DiVa. And all tests are executed in CANoe environment. By using these tools, users can test the whole target diagnostic protocol stack automatically and correctly. Nowadays this method is widely used in the automobile industry. However, these professional tools are experience. And the whole test should totally relay on this Vector's testing tool chain.

A new test case design method is proposed and a new scripts generator is developed in this article. By importing a ODX project file, [4] which is specified by ISO22901, the generator generates test specification reports and test scripts. These scripts can be executed in corresponding test environment. This method prevents developers from writing complex test scripts, reduces the workload and increases the efficiency of development. Besides this method does not relay on specific test environment. By means of configuring script template, the generator can generate different test scripts to meet the needs of different test executing environment.

System Overview

Fig.1 shows the system structure, including script generator part and test environment part.

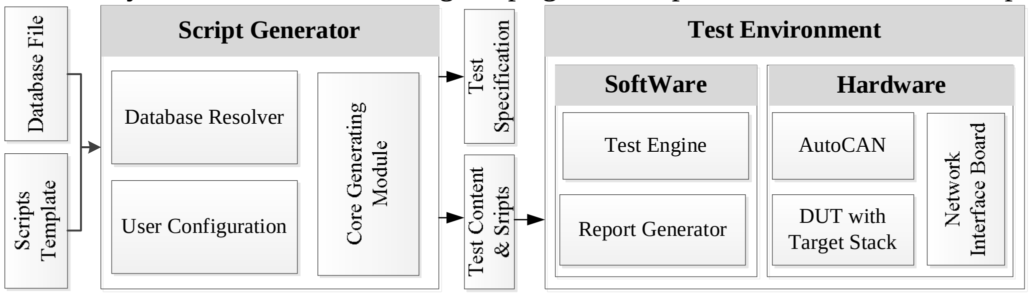


Fig. 1 System Structure

The function of the generator is to generate the test specification reports and scripts according to the imported ODX files and users' configuration. The database file resolver and user configuration module gather information from imported ODX files and user configuration interfaces respectively. Both kinds of information are transmitted into the core generating module. And finally the generator generates specifications and scripts.

The function of the test environment is to execute test scripts. Software environment can run the scripts, control the hardware tool to stimulate communication on real CAN bus, record test steps and data flows in details and finally generate test reports. Hardware environment is a real communication network. AutoCAN is a CAN bus design tool from the company called ihr. It is controlled by PC and stimulate or monitor the behaviors on the CAN bus. DUT represents device under test.

Design of Test Method

Test Object. UDS protocol stack is the main test object, which is installed in a network node inside a vehicle and responsible for the diagnostic communication process. The structure of a protocol stack base on CAN bus is shown below.

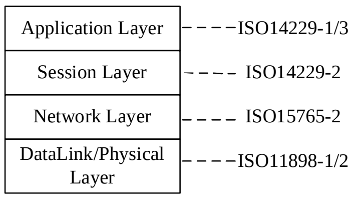


Fig. 2 Structure of Target Protocol Stack

The non-application layers ensure that the application data can be transmitted correctly to the upper layer. The application layer is responsible for processing the request and give the response according to ISO14220-1. The purpose of the test is to verify whether this diagnostic stack can process the request correctly and give corresponding response.

Test Inputs. The input of this test system is a specification file of the target stack, which records all the supported services and their attributes. ODX file is used as the input file, which is a standard format according to ISO22901. This standard format makes sure that there is only one source of the diagnostic data during the whole life cycle of a ECU. It simplifies the data communication among vehicle factories, OEMs, tool suppliers and after sale servicers and increases the efficiency. [5]

ODX file is stored in XML format. Different types of data have different extended file names. For example, the file with ‘odx-e’ extended name stores the information about ECU configuration. The ‘odx-c’ and ‘odx-cs’ files are related with communication parameters. Information about diagnostic services is stored in file called ‘odx-d’.

Test Case Design. The orthogonal design method along with other assistant method are used to design each test case.

By using scenario method, test cases are classified into different classes according to its service. For example, ‘Diagnostic Control Service’ and ‘Security Access Service’ are two different kinds of test classes.

By using equivalence class classifying method, a test value can be selected to represent a set of values, which will get the same result. This step can reduce the number of test cases significantly.

By using orthogonal design method, all test cases can be designed clearly and orderly. Firstly, the factor number and factor values should be decided. The table below shows the factor information of ‘Diagnostic control service’ test class.

Table 1 Factors of 'Diagnostic Control Service' Test Class

Type	Sub-function	suppressPos-RspMsgBit	Error Situation
OK	0x01	0	Sub-function invalid (NRC=0x12, sub-function ID = 0xaa)
Error	0x02	1	Sub-function invalid (NRC=0x12, sub-function ID = 0xcc)
-	0x03	-	Data too long(NRC=0x13, invalid data = 0x11)
-	-	-	Data too long (NRC=0x13, invalid data = 0x22)
-	-	-	No sub- function(NRC=0x13)

Then all the factors above can be arranged and combined with each other in an orthogonal table. Finally, a test case table is formed. The following is an example table, which shows parts of the test cases in ‘Diagnostic control service’ test class. Each row in this table describes a test case in details.

Table 2 Examples of the Orthogonal Table (Diagnostic Control Service)

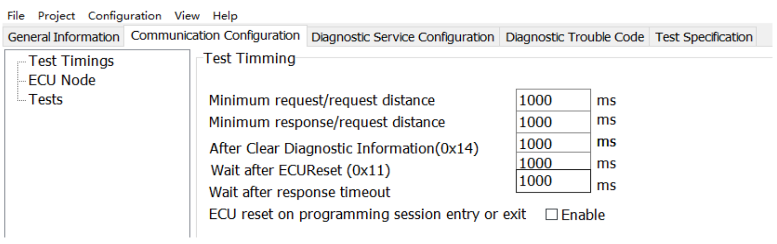
Index	Type	Sub-function	suppressPosRspMsgBit	Error Situation
1.1.1	OK	0x01	0	-
1.1.2	OK	0x02	0	-
1.4.4	Error	0x02	0	NRC=0x13 long + 0x22
1.4.5	Error	0x03	0	NRC=0x13 long + 0x11

Test Script Generator

Workflow of Script Generator. Test scripts are generated by a self-developed generator. The workflow is as follows.

1) Import an ODX project file;
2) Extract useful data and store them into internal memory of the generator;
3) Get users’ configuration from user interface;
4) Form test case content according to the imported data, users’ configuration and built-in test case generation strategy;
5) Print test specification report and export PDF file by using FastReport tool;
6) Fill in test script template and generate scripts for each test case.

Parameter Configuration. Test parameters can be configured though user interface, which is shown below.


File Project Configuration View Help

General Information | Communication Configuration | Diagnostic Service Configuration | Diagnostic Trouble Code | Test Specification

Test Timings
ECU Node
Tests

Test Timming

Minimum request/request distance 1000 ms
Minimum response/request distance 1000 ms
After Clear Diagnostic Information(0x14) 1000 ms
Wait after ECUReset (0x11) 1000 ms
Wait after response timeout 1000 ms
ECU reset on programming session entry or exit ☐ Enable

Fig. 3 Generator User Interface

Different kinds of parameters can be found in corresponding tab.

'Communication Configuration' tab is related to communication parameters, such as minimum request distance, timeout time and so on.

'Diagnostic Services Configuration' tab includes a list, which lists all services supported by this target stack. User can select the target test services in this test round.

'Diagnostic Trouble Code' tab includes configuration about DTC and related test parameters.

'Test Specification' tab is used to show the current generated test content.

Generator Outputs. There are three parts of the outputs.

The first one is the content of the generated test cases this time, including case indexes and synthetic names. This content is used to configure test environment.

The second one is a test specification report, which is a reference for the user. Fig.4 is an example of this specification, which describes the test request and expected response in details.

```
1.4.6 DiagnosticSessionControlExtendedSession_err_supFalse_0x13_+0x22

Request
RequestID 0x10
SubFunction 0x03
LongData 0x22

Response
NegRespID 0x7f
requestID 0x10
NRC 0x13
```

Fig. 4 Example of Test Specification

The third one is test scripts, which can be executed directly in target environment.

Test Implement

Establish the test environment. The first step of a test is to build a test environment. The following figure shows the structure of the hardware environment.

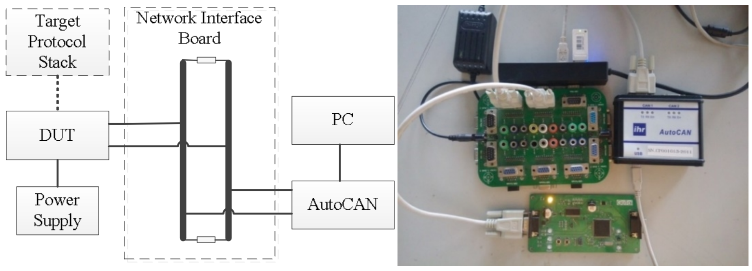


Fig. 5 Structure of Hardware Environment

The target stack is download into a DUT. which is connected with AutoCAN though a CAN bus interface board. AutoCAN is connected with PC and stimulates bus behaviors according to the commands from PC.

The test software is ihr Test System. There is a test case list on the left side, which can be configured flexibly. The right part can show test logs and summary. By double-clicking a term in the case list, a script editor pops up, in which test scripts can be edited. In this article, test scripts are generated by the generator automatically.

Test Steps. The standard test steps are listed as follows.

1) Get ODX project files of the target stack and import them into the generator;

2) Configure test parameters and select target test services;
3) Use generator to generate test case content, specification report and scripts;
4) Import test content into the test system environment and configure it;
5) Select target cases, compile and execute test scripts;
6) Finish the test and get the test report automatically.

Test Verification. In order to verify the test method described in this article. A standard DUT is used as a standard device. Finally, the test results are compared with the standard test results. By this comparison, the new method can be verified.

The verification process is divided into two steps. The first step is called OK test. In this kind of tests, all the data in ODX file is matched with the target DUT. The second step is called error test. Some data is changed in the data base to make error test result on purpose. This step can verify whether the system can find the errors.

38 test cases are used to do the system verification, including two common service classes. In the first step, all cases are passed as expected. In the second step, the key value of the security access service is changed, which results failure in two relevant test cases. The statistic figures of these two test rounds are shown below.

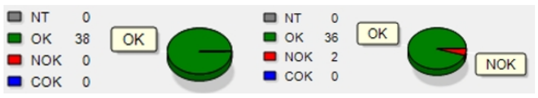
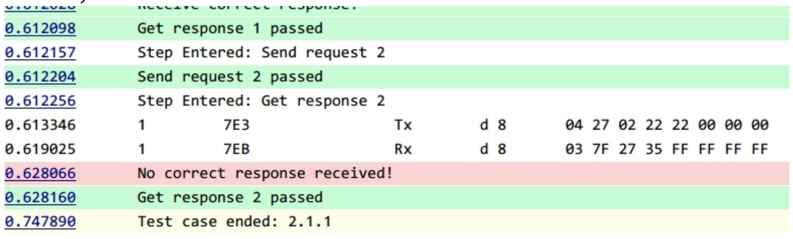


Fig. 6 Statistic Figure (Left: OK Test, Right: Error Test)

A test report is generated automatically after each test step finishes. In the report, test steps and data flow on the CAN bus are recorded in details. If these is any failure during the test, the error step is highlighted by red. This is helpful for the developer to locate errors. Fig.8 is an example of a failed test report.

Though the tests above, the functions of this test method is verified.



```
0.612098    Get response 1 passed
0.612157    Step Entered: Send request 2
0.612204    Send request 2 passed
0.612256    Step Entered: Get response 2
0.613346    1       7E3             Tx      d 8     04 27 02 22 22 00 00 00
0.619025    1       7EB             Rx      d 8     03 7F 27 35 FF FF FF FF
0.628066    No correct response received!
0.628160    Get response 2 passed
0.747890    Test case ended: 2.1.1
```

Fig. 7 An Example of Failed Test Report

Conclusions

This article mainly describes an automated test method for UDS protocol stack of vehicles. A case generating strategy is designed according to international standards and a test scripts generator is developed. It can import target database files and generate test specification and scripts. These scripts are executed in ihr test system and control the bus behaviors thought a CAN bus tool. When all test steps finish, a report is generated automatically. This method can improve the development efficiency of UDS protocol stack and reduce the workload of developers.

References

[1] ISO14229-1. Road vehicle - Unified diagnostic services (UDS) - Specification and requirements[S]. 2013.

[2] Han Xin, Bao KeJin, Development and test of CAN bus network layer protocol stack[J], Computer Engineering, 2011, 15: 232 - 234 + 237.

[3] Peti, Philipp, et al. A quantitative study on automatic validation of the diagnostic services of Electronic Control Units. Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on. IEEE, 2008.

[4] ISO22901-1. Road vehicles - Open diagnostics data exchange (ODX) - Part 1: Data model specification[S]. 2008

[5] Chen Yupeng, Ge Liang et al. Research and application of ODX based on UDS diagnostic protocol[C]. Annual Conference of Society of Automotive Engineers of China, 2014:4.