Third-party Evaluation of the Flight Control Software

Li Yuan, Xueyang Sun and Xingyu Liu
Shaanxi Zhongtian Rocket Technology Co., Ltd, Xi’an, Shaanxi, 710000, China
*Corresponding author’s e-mail: yuanli_2132@126.com

Abstract. Flight control software which is the core of missile flight control system guides and controls missile flight along program trajectory. Effective testing of flight control software is an important means to ensure the quality of flight control system. Software testing and evaluation is the comprehensive measurement and assessment of the software quality in accordance with relevant standards taken by the independent third-party testing and evaluation organizations according to the requirements of entrustors. This paper mainly tests the configuration items of flight control software. Firstly, the characteristics of the software under test are introduced. Then, the testing strategies, methods and environment are given. Finally, the test results are analyzed.

1. Introduction
The third-party evaluation is the last step before the software is delivered to users. Unlike internal evaluation, it is evaluated by independent third-party organizations with relevant qualifications. According to relevant national standards and industry standards, third-party independent organizations conduct comprehensive testing and evaluation of the quality of the software, verify whether the software under test meets the overall requirements of development and the degree of satisfaction, and find the problems of the software as far as possible[1].

Flight control software guides and controls missile flight along program trajectory, which makes it possible to accomplish intelligent tasks such as automatic navigation, target recognition and precise control without human intervention. Flight control software has the characteristics of complex interface, strong real-time and complex logic and once problems arise, may cause disastrous consequences. The third-party evaluation of flight control software is an effective means to ensure the quality of flight control system.

2. Brief introduction of the software under test
Flight control software which is embedded in missile-borne computer belongs to the typical real-time embedded software and has the following characteristics:

   (1) Complex Interfaces: As shown in figure 1, the software under test interacts with various external components, such as launch control box is 1553B digital communication interface. IMU, seeker, GPS and steering gear are RS422 communication interfaces. Battery voltage is A/D acquisition interface. Electrical separation is discrete input switching signal. Control/power electrification is discrete output switching signal.
Figure 1. Interface Diagram of Flight Control Software.

(2) Real Time: Flight control software’s main business cycle is strictly limited to 2ms. Each cycle must complete information collection, navigation calculation, control and guidance law calculation, and communication with various peripherals. To achieve precise control, flight control software not only has fast calculation speed but also requires high accuracy.

(3) High Reliability: In order to ensure flight safety and hit the target, flight control software should make logical processing for various possible abnormal situations, such as gyro data abnormality, launch command is jammed, detection target is jammed, etc.

The above characteristics of the flight control software determine the complexity and difficulty of the software test system.

3. Testing strategies and methods
This paper combines static test with dynamic test to test configuration items of flight control software. The test strategies are as follows:

(1) Each feature of the configuration item is covered by at least one normal test case and an exception test case.

(2) The input of test cases includes at least valid equivalent class value, invalid equivalent class value and boundary data value.

(3) Test the total storage of configuration items, throughput capacity of input/output channels, and processing time margin.

(4) Test the solutions used in design to improve the security and reliability of configuration items, such as fault tolerance, redundancy, interrupt handling, etc.

3.1. Static test
Based on the principle of "consistency of documents and code", the completeness, consistency and correctness of documents related to the software under test, such as task book, requirement specification, design report and mathematical model description, are checked first, and then the code is checked according to the documents, including code format, memory usage, expression judgment, etc.

The test tools of Logiscope and testbed are used to analyze the static state of flight control software, check the standardization of software programming, evaluate the quality of programming, and obtain the structural metrics of software design, including cycle complexity, call hierarchy, hierarchical complexity, etc.

3.2. Dynamic test
On the platform of real-time simulator, dynamic testing combines white-box testing with black-box testing. According to the input of design requirements, interface files and mathematical models, it divides function, performance, interface, security, boundary, margin, data processing and other test items.
White-box testing adopts the technology of pile-inserting, which inserts some probe functions into the program on the basis of maintaining the original logic integrity. For example, when testing the running time of interruption function, discrete output switches signals are inserted at the beginning and end of the function respectively. When white-box testing is carried out on two-dimensional interpolation function and atmospheric solution model, the input values of pile-inserting function and model are obtained.

Black-box testing does not consider the internal logic structure of the program. It relies on the requirements specification of the program, designs test cases according to the test strategy, and checks the correctness of the program[4].

![Figure 2.Dynamic test flow chart.](image)

### 4. Testing environment

Test environment includes real-time simulator, oscilloscope, power supply, cable and so on. The real-time simulator is composed of the host computer and the target computer and they are connected through the wire. The host computer is responsible for model building, compiling, downloading and the monitoring of real-time simulation. Through the HiGale simulation platform, the rapid modeling based on MATLAB/Simulink is realized. The designed model mainly includes dynamic model, launch and control model, steering gear model, seeker model, etc. After compiling, the model is downloaded to the target computer to run. The rich virtual instruments provided by HiGaleView software can be used to observe and analyze variables in real time. The target computer is composed of chassis, processor board and data acquisition I/O card. It is responsible for the operation of real-time simulation model and the control of controlled unit. The real-time operating system QNX is used to ensure the real-time performance of simulation. The communication between processor board and I/O card is based on PXI bus. The PXI system has the characteristics of high reliability, easy maintenance and easy expansion of hardware. Specific functions are as follows:

1. Rich interface, has RS422, RS232, AD, DA, CAN, 1553B and other interfaces.
2. Synchronization between interfaces reaches 1ms level, realizing multi-input and multi-output functions.
3. It can change the test sequence and communication data format on the spot, and can simulate all kinds of timing and communication faults.
4. User friendly interface, real-time monitoring and storage of data.

The architecture of the real-time simulator system is shown in figure 3 and the connection of flight control software evaluation system is shown in figure 4.
5. Test results

Static testing mainly checks out the inconsistency between documents, code inconsistency with C language coding rules, low code annotation and other standardization problems. Table 1 shows static test results.

| Metric elements non-conforming quota | description                              | quota  | Number of |
|--------------------------------------|------------------------------------------|--------|-----------|
| Notes                                | Valid notes for software unit            | ≥20%   | 33        |
| Fan out number                       | Number of calls to underlying functions  | <7     | 4         |
| Function parameters                  | Number of function parameters            | ≤7     | 1         |
| Structure measurement                | Cyclomatic complexity                    | ≤10    | 9         |
| Structure measurement                | Basic cyclomatic complexity              | ≤4     | 6         |

A total of 362 test cases were designed for dynamic testing, of which 39 failed. The specific results are shown in Table 2. Twenty-eight of the failed test cases belong to functional testing, mainly due to problems in the self-inspection process of a device. The self-inspection process of a certain equipment...
consists of five steps. Each step shakes hands twice with the flight control software. The program defines five variables represent five steps. Due to errors in the use of variables, the second handshake can not be carried out if the first handshake is unsuccessful.

Serial simulation assistant is used in software internal testing. The time precision of serial simulation assistant is s level, while the two handshakes between flight control software and equipment are ms level, so this part can not be tested in internal testing.

6. Conclusion
The dynamic test environment of the simulator built in this paper is easy to operate and test cases are easy to implement, which meets the requirements of real-time embedded software testing. Using this platform, a flight control software bug is tested, which improves the reliability of the software.

Third-party evaluation is the last link before the software is delivered to users. Before third-party evaluation, the software usually passes the internal test, so the software problems are generally hidden, and it is difficult to find problems through code inspection. According to the evaluation results of a flight control software, dynamic testing is easier to find program problems than static testing.

From the problems exposed by dynamic testing, we realize the importance of programming standardization. When there are many logical branches, it's better to use case jump statement to avoid variables as much as possible. Reflecting on the importance of fault injection method in software testing, software problems often occur in redundant design and processing part, so we must simulate all kinds of faults to fully test the logical branches of software redundancy processing.

References
[1] Wang, F., Zheng Y.X., Bao,Y. (2008) Software third-party evalution. Computer Research and Development, 45: 345–350.
[2] Yuan, Y.Q. (2014) Design and Implementation of Flight Control Software for Aviation Time Sensitive Guided Bomb. Computer Research and Development.
[3] Liu, L.M., Wang W.Y, Tang,K. (2005) Test method and technology of embedded software. Computer and Modernization, 4: 123–126.
[4] Ma, R.F., Wang, H.R.(2003) Research on software testing method[J]. Minicomputer system, 12: 2210–2213.
[5] Wang, Q., Zhang, X.H., Zhou, M.G., Huang, H.(2008)Research on test method of UAV flight control software. Aeronautical Computing Technique, 2: 78–81.
[6] Li, N., Li, Z.H.(2009)Research and practice of software test policies based on black box testing. Application Research of computers,3: 923–926.