Research on Key Test Methods of the Smart Meter Software Based on Failure Modes

Zhang Leping¹,a, Wang Baoshuai¹,b, Hu Shanshan¹, Luo Yi¹, Pan Yi²,c, Wu Zhanhe²,d

¹China Southern Power Grid Research Institute Co., Ltd. Guangzhou, China
²Holley Technology Co., Ltd. Hangzhou, China
azhanglp@csg.cn, bwangbs@csg.cn, cyi.pan@holley.cn, dzhanhe.wu@holley.cn

Abstract—Software quality has become a key factor affecting the overall quality of the smart meter. To improve the reliability of the smart meter software, analyzed the typical failure modes and mechanism based on actual failure cases of smart meters. For different software failure modes, the test techniques were studied from the perspective of the software development cycle. Designed a test program of the smart meter software and provided automated test tools to improve the breadth and depth of the software test and improve the product reliability.

1. INTRODUCTION
As the intelligent level of the smart meter increases, the scale and complexity of software grows exponentially, and it gradually becomes a key factor affecting the quality of smart meters. Therefore, the reliability and detection depth of the smart meter software design are very high. In particular, the IR46 standard has added software inspection projects, which puts high requirements on the reliability of the energy metering software [1-2]. This paper aims at the software quality of the smart meter. According to the failure data of the smart meter in actual operation, the software failure mode analysis is conducted based on the software failure modes and effects analysis (SFMEA) to establish the software failure mode of the smart meter. The life cycle test program and test method of the smart meter software are studied for different failure modes, which provides test methods and means for improving the reliability and maturity of the smart meter software.

2. EMBEDDED SOFTWARE DEFECT TYPES
The software failure mode is a manifestation of software failure. The purpose of the software failure mode analysis is to find out all possible failure modes for each software unit being analyzed, and then analyze the possible causes according to the failure mode. The cause of the software failure is often caused by various defects formed during the software development process [3].

The smart meter software is the basic embedded software. This paper is based on the software failure modes and effects analysis (SFMEA) in the standard GJB/Z 1391-2006 Failure Modes, Impact and Hazard Analysis Guide. The analysis of general embedded software failure types, combined with the software defect research of the actual failure case of smart meters, proposes the software failure mode suitable for the smart meter [4]. The types of defects that the embedded software may have from its own attributes and the causes of the defects are shown in Tab.1 below.
TABLE 1. DEFECTS TYPES AND CAUSES OF EMBEDDED SOFTWARE

| No. | Defect type                  | Defect cause                                                                 |
|-----|------------------------------|-----------------------------------------------------------------------------|
| 1   | Demand                       | 1) Software requirements are unreasonable or incorrect; 2) requirements are incomplete; 3) logic errors; 4) requirements analysis documents are incorrect. |
| 2   | Functional and performance   | 1) Function and performance regulations are incorrect, or miss functions, or have redundant functions; 2) information provided to the user is wrong or inaccurate; 3) abnormal situation is handled incorrectly. |
| 3   | Software structure           | 1) Program control or control sequence is incorrect; 2) processing is incorrect. |
| 4   | Data                         | 1) Data definition or data structure is incorrect; 2) data access or operation is incorrect; 3) variable scaling ratio or unit is incorrect; 4) data range is incorrect; 5) data is incorrect or missing. |
| 5   | Software implementation and coding | 1) Coding or button is wrong; 2) violates the coding style requirements or standards; 3) grammatical errors; 4) data name errors; 5) local variables are confused with global variables. |
| 6   | Software/hardware interface  | 1) Internal interface and external interface of the software are incorrect; 2) relevant parts of the software are not coordinated in terms of time coordination or data throughput; 3) I/O timing error causes data loss. |

The most fundamental and important defects in embedded software are classified into functional defects, performance defects, and reliability defects. The specific analysis is as follows\(^5\):

1. General class does not meet the basic requirements of functional and performance defects: generally, cannot meet the user's requirements for time, data processing or concurrency; reliability design lacks redundancy or abnormal fault-tolerant design;
2. Data defects: refer to normal class parameters, normal input data, normal output data, normal order of data, data format, data precision, data conversion, etc.\(^6\)
3. General conventional defects (software implementation and coding defects): Usually, they do not conform to the coding specification, and the syntax, data, variables are written incorrectly.

Depth defects of the difficulty of testing include the following aspects:
1. Structural logic control defects: the control sequence is incorrect or the processing is incorrect;
2. Data defects: unconventional input and output data, including boundary data, limit data or fault-tolerant classes, concurrent data; access data exceptions, including data access errors, data loss, and data corruption;
3. Interface errors: internal and external interface, I/O timing error, etc.
4. Others: The hardware components are disturbed or damaged, resulting in software anomalies and artificial illegal operations.

3. SOFTWARE FAILURE MODE ANALYSIS OF THE SMART METER

The analysis of 121 failure cases caused by the quality of the smart meter software itself shows the current smart meter software failure modes. Laboratory tests are mainly functional failures, and the main failures occurred in the operation site are program reliability failure and strong interference. The failure modes and the corresponding failure causes are classified as shown in Tab.2 below.
TABLE 2 THE FAILURE MODES AND CAUSES OF THE SMART METER SOFTWARE

| No. | Failure mode | Failure cause |
|-----|--------------|---------------|
|     |              | Incorrect parameter setting |
| 1   | Data class   | Inconsistent communication protocol |
|     | Normal functional input and output data | Cannot accept normal data input and input order |
|     |              | Output format error, parameter incomplete, etc. |
|     |              | Output data accuracy is out of tolerance |
|     | Reliability input and output data | Baud rate change |
|     |              | Communication address change |
|     |              | Accept special input data boundaries, limits, failure tolerance class handling errors |
|     | Storage class | Data access error |
|     |              | Data confusion |
|     |              | Data overflow |
| 2   | Program failure | Loss of function |
|     | Functional design failure | Inconsistent function and demand |
|     |              | Logical control (data stream) error |
|     | Reliability design class | Abnormal reset |
|     |              | Logical control (data stream) error |
| 3   | Normal or abnormal power-on and power-off | |
| 4   | Hardware damage causes software exception | |

4. RESEARCH ON TEST MODEL AND TEST METHOD OF THE SMART METER SOFTWARE

4.1 Software test model

Software defects are formed during the development process. A typical software test V model reflects the relationship between test activities and analysis and design, as shown in Fig.1 below. The left side of the model is the development phase, and the right side is the test phase. The development phase begins with defining the software requirements, then transforms the requirements into a summary design and detailed design, and finally forms the program code. After the code is coded, the test phase begins with unit testing, then integration, system testing, and acceptance testing [6].

![Software test process V model](image)

Figure1. Software test process V model

From coding, unit testing, integration to systems and validation testing, different testing techniques can be used at each stage to trigger different software defects. Software testing techniques include static testing, dynamic testing, white-box testing, black-box testing, and gray-box testing from different categories.

Static testing is the process of not actively executing a program and looking for possible errors in the code or evaluating the quality of the program code. Mainly through code review and static analysis,
check the rationality of the code structure, the consistency of code and design, the implementation of coding standards and so on. Experience has shown that static testing can effectively detect 30-70% of errors in the software itself. At present, static testing is usually introduced during the coding phase. Especially with the development of static defect analysis tool and coding rule detection tool, static testing has high efficiency and good effect, and can effectively solve basic coding irregularities, typical software defects (array out of bounds, buffer overflow, memory leak, variable not initialized, etc.) [7].

Dynamic testing requires the design of specific test cases. The validity of the test cases is particularly important by determining whether the input and expected output results meet the requirements. Dynamic testing typically includes white-box testing, black-box testing and gray-box testing. White-box testing is a test of the internal logic structure of the program, with statement coverage, branch coverage, etc. from different test depths. Black-box testing is based on the input and output of the program function according to equivalence partitioning, functional analysis, boundary value analysis, the intellectual tour, scenario method and other methods to verify the software correctness of different input. Gray-box testing between white-box testing and black-box testing, not only pay attention to the correctness of the output and input, but also pay attention to the internal conditions of the program. And the internal operating state is judged by some characteristic phenomena, events and signs [8].

4.2 Design of Key Test Scheme Based on failure Mode of the smart Meter Software

According to the analysis of the typical failure mode of the smart meter software, emphasis includes data type failure modes, that is, it cannot accept normal input/output, output format error, boundary and limit data processing error, data access or operational failure, etc.; program-type failures and power-on and power-off failures, including software structure defect errors, functional and performance defects; general parameter setting failure or program function design and demand inconsistency failure, mainly due to software implementation and coding defects. Different software defects are formed in the process of software development. Combined with different testing methods in the software development phase, a software test model for designing a full life cycle of the smart meter is shown in Fig.2 below.

The smart meter software testing model includes static testing, dynamic white-box testing, gray-box testing and black-box testing, and provides automated test tools for testing defects at different stages. Software testing is based on automated testing, with the goal of automating complex test, either completely or partially, reducing test overhead, improving test effectiveness and efficiency, and reducing human error. Usually white-box testing, focusing on the defects in the natural properties of the software, general-purpose commercial software test tools are available; the whole machine black-box testing and gray-box testing are based on the product application function required by the user as a direct test point, and a specific test tool needs to be developed.

1) Static testing: It includes software defect analysis, focusing on specific software defect rules such as array out of bounds, variable initialization, buffer overflow, etc., using commercial Klocwork, polyspace, etc. for rapid automated testing. The coding specification is generally based on the embedded MISRA C programming specification, which is a common specification in the embedded software industry. It implements more than 100 standardized writing requirements, such as code declaration and definition implementation, arithmetic type conversions, pointer type conversions, expressions, control flow, functions, structures, unions, etc.; software quality measurement: quality measurement such as software circle complexity, fan-in/fan-out, etc. can optimize software structure, reduce complexity, and reduce risk; coding specifications and software quality measurement can be used in commercial QAC, C++Test, Testbed and other automated test tools [9].
2) Dynamic white-box testing: It focuses on the code logic coverage test. The unit test implements the code statement coverage, branch coverage, condition coverage, etc. to ensure that all the software codes are tested and verified. Prevent potential defects in the code that is never executed, and provide a complete measure of the test. The integration test is based on the interface test of the module and the module to verify the correctness of the interface interaction. Automated testing can be performed using commercially available test tools such as Tessy, C++Test and Testbed.

3) Dynamic gray-box testing: For the target code, the internal storage and interface interaction tests of the smart meter software are performed in the semi-physical simulation environment. Based on the technical characteristics and advantages of FPGA reprogrammable, two-way communication, high operating speed and high data processing capability [10], simulated smart meter metering MCU (or management MCU, IR46 meter) peripheral measurement core and interface, clock chip and interface, EEPROM and interface, Flash and interface, communication module interface and ESAM interface, and made semi-physical simulation test environment. Instead of traditional software, hardware environment simulation is used to implement software module interface test or software failure injection test, and performs reliability testing such as parameter protection and software protection for failure modes such as the smart meter calibration parameters and important storage energy data. And performing the IR46 meter legal software and illegal software separation test, online upgrade test, etc.

4) Dynamic black-box testing: It is mainly based on the system verification of the whole machine. From the user's perspective, it is simpler and more convenient. It comprehensive uses of black-box testing function analysis method, equivalence class division method, boundary value method, guessing method, causal map method and other test case design methods to improve the depth and breadth of the test case writing. It improves the coverage of the test. Design test cases to trigger the smart meter
software defects from the basic function test, fault-tolerance test, boundary test, limit test, and power-on/off test.

5. CONCLUSION

Based on the basic theory of software failure mode, this paper establishes a typical failure mode model of the smart meter software by analyzing the software defects and causes of 121 smart meter failure cases. According to different failure modes, from the perspective of software development life cycle, the smart meter software test program was designed. The program provides testing methods and corresponding automated test tools from static coding testing, software white-box unit testing, gray-box testing and whole machine black-box testing. The research results can help standardize the test method of smart meter software, improve the breadth and depth of test, improve the reliability of software, and provide the means for software testing and quality assurance of energy metering equipment.

ACKNOWLEDGMENT

This research was financially supported by China Southern Power Grid Co., Ltd. Science and Technology Project (ZBKJXM20180127).

REFERENCES

[1] Song Xiqiang, Wang Pingping, and Zhou Shaoyuan, “Overview of intelligent energy meter software testing technology,” Electrical Measurement & Instrumentation. China, vol. 51(11), pp. 18-22, 2014.
[2] Pang Fukuan, Wang Yang, Yuan Ruiming, Ju Hanji, Song Xiqiang and Wang Pingping, “Research on key technology of intelligent energy meter software detection based on R46 standard,” Electrical Measurement & Instrumentation. China, vol. 55(19), pp. 130-134, 2018.
[3] GJB/Z 1391-2006, Guide to failure mode, impact and hazard analysis.
[4] Li Wei, “Failure mode and impact analysis of single-phase smart energy meter,” North China Electric Power University, 2012.
[5] Sun Zhian, “Software reliability engineering,” Beijing University of Aeronautics and Astronautics Press, China, 2009.
[6] Cai Jianping, “Embedded software testing practical technology,” Tsinghua University Press. China, pp. 192-200, 2010.
[7] Liu Chunlu and Huang Zihe, “Software reviewer course,” Tsinghua University Press. China, pp. 57-59, 2005.
[8] GB/T 15532-2008, Computer software testing specification.
[9] Hu Danrui, “Research and implementation of software static test based on LDRA Testbed,” Computer Security. China, vol. 6, pp. 69, 2012.
[10] Hong Yingsen, “Research on fault injection system based on FPGA,” Harbin Institute of Technology. China, 2014.