Research on Software Failure Modes and Key Testing Methods of the Smart Meter

Fukuan Pang¹, Wenwu Cui¹, Chen Wang¹, Di Han¹, Yi Pan” and Pingping Wang²

¹State Grid JiBei Electric Power Co.Ltd Research Institute, Beijing, 100045, China
²Holley Technology Co.Ltd, Hangzhou, Zhejiang, 310023, China

*Corresponding author’s e-mail: yi.pan@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, this paper studies the software failure modes and influence analysis of the smart meter based on SFMEA, analyzes software failure modes and mechanisms, and establishes fault-knowledge base; the key technologies of static testing, dynamic testing, black-box testing and white-box testing were carried out for different software failure modes. The test programs of the life cycle of the smart meter software and the support of the automated testing tools are designed to improve the breadth and depth of the software testing, improve the reliability, reduce the operational failure and improve the service level of the electricity.

1. Introduction
With the improvement of the intelligent level of the smart meter, the software scale and complexity have doubled, which has become a key factor affecting the quality of the smart meter. It has placed high demands on the reliability and depth of detection of the smart meter software design. In this paper, based on the software quality and the actual operating fault data of the smart meter, the software failure mode analysis is carried out based on SFMEA, and the smart meter software fault-knowledge base is established. The smart meter software life cycle testing programs and testing methods are studied for different failure modes. They provide testing methods and means for improving the reliability and maturity of the smart meter software.

2. SFMEA (Software failure mode and effects analysis)
Failure Mode and Effects Analysis (FMEA) is an important work item for product reliability analysis and the basis for maintenance analysis, safety analysis, test analysis and support analysis. This paper adopts the software FMEA (SFMEA software failure mode, impact and hazard analysis) in the standard GJB/Z 1391-2006 Failure Mode, Impact and Hazard Analysis Guide, mainly through identification of the software failure modes, researches and analyzes of the causes of various failure modes and their consequences, and finds ways to eliminate and reduce their harmful consequences to identify potential problems as early as possible and take corresponding measures to improve the reliability and security of the software [1].

The purpose of software failure mode analysis is to find out all possible failure modes. The failures are generally caused by various defects formed during the development process. The causes of software failures are classified by defects and typical examples are shown in Table 1 below [2].
Table 1. Reasons for software failures are classified according to their defects and typical examples.

| No. | Software defect type                  | Detailed software defects                                                                 |
|-----|--------------------------------------|------------------------------------------------------------------------------------------|
| 1   | Demand defect                        | 1)Software requirements are unreasonable or incorrect; 2)The demand is not complete; 3)Have a logic error; 4)The requirements analysis document is incorrect |
| 2   | Functional and performance defects   | 1)Functional and performance regulations are incorrect, or missing features, or redundant features; 2)Providing users with information that is incorrect or inaccurate; 3)Wrong handling of exceptions |
| 3   | Software structure defect            | 1)Program control or control sequence is incorrect; 2)Wrong process                       |
| 4   | Data defect                          | 1)The data definition or data structure is incorrect; 2)Incorrect data access or production; 3)The variable scaling ratio or unit is incorrect; 4)The data range is incorrect; 5)Data error or loss |
| 5   | Software implementation and coding defects | 1)The code or button is incorrect; 2)Violation of coding style requirements or standards; 3)Grammatical error; 4)The name of the data is wrong; 5)Local variables are confused with global variables |
| 6   | Software/hardware interface defect   | 1)The internal interface and external interface of the software are incorrect; 2)The 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 important and common deficiencies are functional and performance deficiencies, including conventional class unsatisfied features, performance requirements and poor reliability design for no redundancy design or abnormal fault tolerance design; deep defects are easily overlooked and difficult to test including structural logic control defects (incorrect control sequence or incorrect processing), data defects (access data, unconventional input, output data), interface errors (internal and external interfaces, I/O timing errors, etc.); general conventional class defects are software implementation and coding defects. They are usually not in accordance with the coding specification, and the syntax, data, variables, etc. are written incorrectly.

3. Software failure mode analysis of the smart meter

3.1 Software Failure Mode and Mechanism Analysis of the Smart Meter

According to the collected failure information of the smart meter, 105 failure cases caused by software defects were classified, including 50 cases in the field and 55 cases in the laboratory.

From the results, the power, communication, display, cost control and battery undervoltage in the product failures are the modules with high probability of failure in the field and the laboratory. From the specific failure mode inside each module, the field failure modes are more focused on software reliability, the laboratory failure modes are more focused on software functionality. According to the detailed fault mode analysis of SFMEA, the software failure mode and product fault mapping relationship of each function module of the smart meter are shown in Table 2 [3].

In the table, ★ represents the failure mode that the smart meter often occurs in the field and laboratory conditions.★ represents the failure mode that often occurs in the laboratory. The failure mode of some of the functional modules is not limited to the software failure mode, or it may be an unlabeled software failure mode. The table is based on the software failure mode with a high frequency occurring in the 105 fault cases collected and may be generated. The analysis of the product function module failure effects (that may cause product failure).

It can be seen from Table 2 that the laboratory faults are mainly reflected in functional faults that are inconsistent with the requirements, including parameter setting errors, missing functions, and normal input and output data errors. The field failure modes are mainly data classes (based on special data boundaries, limits, fault tolerance and stored data), normal or abnormal power-off classes, program fault classes, hardware and external interference software exception classes, etc. Possible product failure modes include failure of each functional module, that is, communication failure, power
failure, display failure, clock deviation, cost control failure, battery undervoltage, clear fault, load curve storage error, etc.

### Table 2. The smart meter software failure mode and matrix relationship that may cause product failure.

| Software failure mode | Product failure mode | Data class | Power up and down | Program fault class | Hardware and external interference software exception class |
|-----------------------|----------------------|------------|-------------------|---------------------|-----------------------------------------------------------|
| Communication data class | | | | | |
| Communication baud rate | | | | | |
| Communication address | | | | | |
| Protocol | | | | | |
| Inconsistent communication | | | | | |
| Changed communication path | | | | | |
| Protocol | | | | | |
| Communication is not on | | | | | |
| Freeze error | | | * | * | * |
| Storage error | | | * | | |
| Metering error | | | * | | |
| Metering error | | | * | | |
| Display control failure | | | | | |
| Display content error | | | | | |
| Black screen, etc. | | | | | |
| Clock deviation | | | * | * | * |
| Pull and close control error | | | * | * | |
| Time rate setting | | | * | | |
| Battery under voltage | | | * | | |
| Programming state | | | | * | |
| Clear fault | | | | | |
| Incident record | | | | | |
| Alarm light control | | | | | |
| Load curve storage error | | | | | |

### 3.2 Establishment of the smart meter fault-knowledge base

According to the above analysis, the smart meter software fault-knowledge base is established, and the software fault case record and fault mechanism analysis of the smart meter field and laboratory test in recent years are completed, including product basic information record, fault function unit division, failure mode definition, problem description, cause analysis, severity, fault location, etc., and fault query according to different categories (module, time, failure mode, etc.). Failure mode classification is based on the typical software defect mode of the smart meter, including input fault, output fault, data fault, interface fault, program fault, unsatisfied function or performance requirement fault, power up and down fault, redundancy fault and so on. The fault-knowledge base can realize the import and
export of the excel file. According to the specified form template, the content can be directly imported into the fault-knowledge base of the smart meter software platform.

The fault-knowledge base implements the management and analysis of the smart meter software failure case and improves the failure diagnosis capability.

4. Research on key testing methods for software typical failure modes of the smart meter

4.1 Software testing

Software quality needs to be guaranteed by full life cycle testing. According to GBT 15532-2008 Computer Software Testing Specification, the complete testing work runs through the whole process of development and validation. The testing must include unit testing, assembly testing, validation testing and system joint testing, and it is tested in the order listed [4]. Software testing methods include static testing, dynamic testing, or black-box testing, white-box testing from different categories [5].

(1) Static testing and dynamic testing

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, etc., such as inappropriate loop nesting, mismatched parameters, casts, null pointers Quotes, etc. Experience has shown that static testing can effectively detect 30-70% of errors in the software itself, especially in the early detection of product defects, reducing the cost of modifying the program later.

Dynamic testing is to actually run the tested program, input the corresponding test case, check the difference between the running result and the expected result, and determine whether the execution result meets the requirements. The test content includes function confirmation, interface test, coverage analysis, performance analysis, memory analysis, etc. The key to dynamic testing is how to design test cases to find as many problems as possible. Because of late intervention, it is not conducive to early detection of defects.

(2) White-box testing and black-box testing

White-box testing treats the test object as a transparent box. Based on the information about the internal logic structure of the program, the test case is designed or selected, and the program logic path is tested to check whether the internal motion of the product is normally performed according to the specifications of the design specification. Including control flow and data flow test, control flow test detects the percentage of the measured source code in the total number of codes, including statement coverage, decision coverage, condition coverage, etc., to prevent the code from never executing to contain potential defects, coverage evaluation is one of the main software evaluation methods; data flow testing is the analysis of the definition and reference of variables, to find undefined variables or variables that are defined but not used.

Black-box testing is that the tester does not consider the logic structure inside the program at all. The test object is regarded as a black box. Only the program function is checked according to the requirements specification whether the program can properly receive input data to produce correct output information.

4.2 Research on key test schemes for the smart meter software failure modes

According to the analysis of the typical failure modes of the smart meter software in Section 2, combined with the software testing method, the software testing program for designing the smart meter is shown in Figure 1[6].
Figure 1. The smart meter software testing solution

(1) From R&D design to acceptance confirmation including static testing, dynamic source code white-box testing, dynamic whole smart meter black-box system testing, static testing including code defect detection, focusing on specific software defect rules such as array out of bounds, variable initialization, buffer overflow, etc. Coding specifications based on MISRA: C Encoding Specification, is a general specification of the embedded software industry, to achieve standardized writing of code, while performing quality metrics such as circle complexity, fan-in and fan-out, optimizing software structure, reducing complexity and reducing risk;

(2) Dynamic white-box testing focuses on code logic coverage testing. Unit testing implements code statement coverage, branch coverage, conditional coverage, etc., ensuring that all code in the software is tested and verified to prevent potential defects in the code that is never executed. It provides a complete evaluation of the test; the integration test is based on the interface test between the module and the module, verifying the correctness of the interface interaction;

(3) Dynamic black-box testing is mainly based on the system verification of the whole smart meter. From the user's point of view, it is simpler and more convenient. The comprehensive use the function analysis method, the equivalence class division method, the boundary value method, the guessing method, the causal map method, etc. These test case design methods improve the depth and breadth of test case writing and improve the coverage of the test. Test cases are designed from basic functional test and fault tolerance, boundary, limit and power up and down tests.

4.3 Automatic test implementation of the smart meter software

Software testing is based on automated testing, with the goal of automating complex testing, either completely or partially, reducing test overhead, improving test effectiveness and efficiency, and reducing human error. Usually white-box testing, focusing on the defects of natural attributes, general-purpose commercial software testing tools are available; the whole smart meter black-box testing takes the user's requirements as the direct test point, and requires specific testing tools. According to the above design of the smart meter software testing program, the white-box testing based on commercial tools and the black-box testing of the customized tool are mainly adopted.

(1) Code static defect detection uses the tool Klocwork, a software that can quickly detect code logic defects based on source code analysis. It can implement null pointer error, memory leak, array out of bounds, buffer overflow, variable uninitialization and other defects master code detection [7];
(2) Static coding specifications, quality metrics and dynamic white-box code testing using the LDRA Testbed tool, with open template configuration or target adaptation technology, fully supports all 8-64 bit power meter microcontrollers, supporting MISRA C: 2012 Encoding Specification, circle complexity, fan-in, fan-out and other quality metrics [8]; Testbed supports dynamic unit testing, integration testing, and can automatically insert code into the object to obtain dynamic execution information for dynamic testing analysis;

(3) The whole smart meter black-box testing according to the requirements of Q/GDW 1354-2013 The Smart Meter Functional Specification, based on the black-box testing theory, using the functional analysis method to decompose the main functions of the smart meter layer by layer, refine it to each specific small function points and design test cases for each test point; it uses the effective equivalence class division method to divide all input data of each function point into several equivalence classes, takes one data in each equivalence class as a test input conditions, using a small number of representative test data to obtain better test results; using the black-box testing case design method of invalid equivalence class method, boundary value method, pressure method covering the above smart meter software communication data class, parameters, special input failure modes such as boundary, limit, fault tolerance. Based on the software script editing technology, the automated test platform HL4800 was developed to realize the black-box automatic testing of the smart meter software operation, which effectively improved the software reliability.

Based on the combination of the above tools Klocwork, LDRA Testbed and customized black-box HL4800, the smart meter software is tested and verified from static testing to black-box verification.

5. Conclusion

Based on the basic theoretical analysis of SFMEA, this paper analyzes the failure mode analysis and corresponding key test methods of the smart meter software under the typical field and laboratory conditions of the smart meter by 105 pieces of fault information that has been collected. The paper analyzes the fault mode and mechanism of the smart meter software, establishes the smart meter software fault-knowledge base, studies the key test schemes for the different smart meter software failure modes, and how to realize the automatic test of the smart meter software.

The research results will help improve the test coverage and defect detection rate of the smart meter software, improve the analysis capability, fault processing efficiency and software reliability of the smart meter field operation failures, reduce the failure rate, and improve the quality of the smart meter software. It has very important practical significance for the construction of the smart grid.

References

[1] Huang, B., Zhang, H., Lu, M. (2009) Software FMEA approach based on failure modes database. International Conference on Reliability, Maintainability and Safety (pp.749-753).
[2] Yang, J., Zhang, J. P., Zhao, L. N., Sa, Y. (2013) A criticality analysis method of failure modes for ordnance software. International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (pp.1592-1595). IEEE.
[3] Bowles, J. B., Wan, C. (2001) Software failure modes and effects analysis for a small embedded control system. Reliability and Maintainability Symposium, 2001. Proceedings (pp.1-6). IEEE.
[4] Xiqiang, S., Pingping, W., Shaoyuan, Z. (2014) Overview of Smart Meter Software Testing Technology. Electric Measurement & Instrumentation, 51(11):18-22.
[5] Jing, Hu. (2008) Analysis of Black-box Testing and White-box Testing. Journal of Hengshui University, 10(1): 30-32.
[6] Liman, S., Xing, H. (2016) Research on Fault Detection Method of Smart Meter Software. Communication World, 2016(11):127-128.
[7] Foster, J. C. (2008) Klocwork Insight 8.0. Information Security.
[8] Danrui, H. (2012) Research and Implementation of Software Static Test Based on LDRA Testbed. Computer Security, 2012(6):69-71.