The Effectiveness of Real-time Embedded Software Testing

Bo Zhang[1,2]
1. Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences
Changchun, China
2. Graduate University of Chinese Academy of Sciences
Beijing China
Zhanghui_net@tom.com

Xiangheng Shen
Changchun Institute of Optics, Fine Mechanics and Physics
Chinese Academy of Sciences
Changchun, China
shenxiangheng@yahoo.com

Abstract—**Real-time embedded software is widely applied in the fields with high reliability and security like aviation and space flight etc. In those systems, software quality is of vital importance, and software testing, as a means of guaranteeing the quality of software, is gaining increasingly more attention. The effectiveness of software testing enables people to gain confidence in software correctness. The real-time and embedding features of embedded software lead to the particularity in real-time embedded software testing. On the basis of particularity analysis of embedded software testing, the author of this thesis further conducted research on the effective Software Testing, and proved that it is valid to adopt testing techniques and strategies through experiments.**

Keywords-software testing; real-time embedded software; the effectiveness of software testing.

I. INTRODUCTION

With the wide application of embedded software in the fields with high reliability and security, embedded software testing is facing increasingly severe challenges, and we have to take the features of embedded software into consideration, while conducting researches on new software testing methods and strategy. At present, although researches on software testing methods are thorough enough, researches on the evaluation of the quality of software testing are insufficient. Software testing is an approach to guarantee software quality, and with its effectiveness people can establish confidence in the correctness of software which has been previously tested.

II. FEATURES OF EMBEDDED SOFTWARE

Compared with general applicable software, embedded software acquires its unique features owing to the specialty of its operating environment and tasks to be finished. Following are the analyses on its features:

1) Strong specificity

Usually, embedded software repeatedly carries out one specific task, and provides single function such as control and simple calculation etc.; once it is loaded to the system to be controlled, its function and action can be fixed.

2) Embedding feature

Generally speaking, embedded software runs in a tailor-designed hardware environment, and the embedded system not only interacts with the external physical world through devices like sensor and actor, but also it can possibly connect and exchange information with other systems through various types of I/0 interface or bus lines. A great amount of hardware operation information is also included in the software. This feature poses new challenge to embedded software testing.

3) Real-time feature

The correctness of many embedded systems is decided not only by the function and behavioral feature of the system, but also by its time performance. We name the real-time system with embedded structure as real-time embedded system, while the software in that system as real-time embedded software. The processing speed of real-time software isn't necessarily quick, but what matters most is punctuality and promptness. Single test and strength test should be run on the time response of real-time embedded software.

4) Variety of the realization of the software

Since embedded software has close relationship with its operating environment, its realization varies with different software; some are supported by operating system, while others operate hardware through assembly language.

Previous analysis have shown that compared with traditional object-oriented and procedure-oriented software, embedded software has its own features which pose new challenge to embedded software testing.

III. THE PARTICULARITY OF EMBEDDED SOFTWARE TESTING

Embedded software acquires features that are different from common software, which leads to the particularity of embedded software testing. The analyses on the particularity of embedded software are listed in the following:

1) Extremely strong pertinence

978-1-61284-666-8/11$26.00

Usually, the software testing method of embedded system is specific to certain type of system or even one typical system, and at the same time it requires auxiliary test tools which are needed to be developed in particular, and those tools are hardly applicable in other embedded software testing.

2) Test cases are large in number, and situation is complex

Cross-linking relationship of an embedded system is complex, and has strong reaction and instantaneity, which results in large input scale and other restraints like sequential relationship, and there is much complexity in input, so both the number and quality of test cases should be increased and enhanced in order to run a sufficient test.

3) High dependability on test tools and environment

The outstanding feature of embedded software lies in that its operational environment differs from its developmental environment, which divides typical embedded software testing into host test and target test. Since embedded software contains large amount of hardware information, many tests conducted on the software aren't sufficient before it integrates with hardware environment, and the real effective test would be the system testing after the integration of embedded system.

At present, software test methods are abundant in kind, and those test techniques and methods are applicable to embedded software, and particularity exists in embedded software test, which decides that test technique of embedded software can't simply follow the traditional way. On the basis of the features of embedded software, research on how to conduct effective embedded software test is necessary and bears significance [1].

IV. RESEARCH ON THE EFFECTIVENESS OF SOFTWARE TESTING

Effectiveness refers to the degree of the completeness of planned activities and the achievement of expected results. Research on the effectiveness of software testing proceeds from the perspective of software testing quality, and its study object is software testing itself instead of software quality [3].

When it comes to embedded software which is widely applied in areas of high reliability and security, the objective of software testing is to pin out the potential faults and defects in the software at the lowest time and human resource cost. If the test is successfully conducted, we can recognize faults existing in the software. The side benefit lies in that it proved that the function and property of target software conform to requirement statements. Furthermore, test result data collected in the test process can provide evidence for reliability analysis.

Evaluation on the effectiveness of software testing includes two aspects: test sufficiency and test efficiency. Test efficiency can be guaranteed through management, assessment, and optimization of the planned test process. Test sufficiency can be conducted through test coverage and evaluation on test case validity, elaboration on which is also the key part of this thesis.

Test coverage is one of the most important methods to conduct qualitative metric and control the process of software test; metric approach based on test coverage can be elaborated as conducting statistics on various coverage rates, and measuring the adequacy degree of the software testing while aiming to satisfy relevant test adequacy standards.

Test coverage can be shown in the following formula:

$$CR = \left(\frac{i_e}{i_t} \right) \times 100\% \qquad (1)$$

CR ——Coverage rate;

i_e ——Number of items executed at least once;

i_t ——Total item number;

In order to measure the adequacy degree of software test from different perspectives, test coverage indicators with different focuses are used, including coverage rates based on code (e.g. sentence coverage, branch coverage, path coverage, branch-condition coverage, and combination coverage), coverage rates based on requirement (e.g. function coverage, requirement coverage, and interface coverage) [4]. In engineering practice, owing to the restrains of test time and cost, several kinds of relevant test coverage are usually calculated respectively according to a few test adequacy standards.

In the process of conducting dynamic test on embedded software, acquiring dynamic code coverage rate information of the tested software and supplementing test case accordingly are considerably useful ways to evaluate the completeness of test case and further enhance the coverage rate of software testing.

Because of the particularity of real-time embedded software testing, i.e. high dependability on test tools and test environment, in the real work of embedded software testing, the acquirement of dynamic coverage rate information of tested software is quite difficult. Especially when the tested software is real-time embedded system, precise measurement of time performance information is very attention-catching [5].

In order to acquire test coverage data, the operating time between different functions or modules and random statement blocks in a module should be exactly calculated, and it would be insufficient to run the test with the assistance of oscilloscope and logic analyzer only, and proper software test tools are needed to chosen. When test tools are at hands, it is still probable that the measurement couldn't be continued because no test interface is left by the experimenter. Simulative test environment has to be developed accordingly and so does test target board.

V. CASE STUDY OF REAL-TIME EMBEDDED SOFTWARE

In this thesis, the author takes control software in the subsystem of a shuttle program of certain type as an example, introduces to the readers present research conducted on the effectiveness of test.

A. Introduction to the software

The target software is real-time embedded software, operating on the hardware platform of microprocessor SMJ320C3X, and functions which have been completed including:

- The function of external communication: receive and process control command, various message and parameters that a principal computer sends; download various engineering specifications reflecting the state of subsystems;

- The function of interior communication: send control command and parameters to subsystems through RS-422 bus line; receive the engineering specifications of those subsystems;
- The function of control: adjust and control focal distance, control position and angle, control temperature;
- The function of calculation: real-time computation on image motion compensation parameter;
- The function of data collection: collect engineering specifications.

Programming language: hybrid programming of C programming language and DSP assembly language, and the developmental environment is C3X Code Composer.

B. Choose test tools

After the comparison and probation of various embedded software test tools which are popular nowadays, the ultimate measuring method of coverage rate is: TestBed, an embedded software test tool developed by LDRA Company, and RTInsight, a real-time hardware data collection tool auxiliary to TestBed, and RTInsightPro, a software to analyze test result and also display results.

C. Set up test environment

Develop test target board based on DSP SMJ320C3X, set up dynamic test platform of embedded software, run coverage test, time performance test and fault model test to embedded software on the platform, etc.

In order to collect and receive characteristic data written to specific address in the program, we need to link data bus of target system with address bus, and also link chip select signal and write signal correspondent to monitored address and also grounded signal of the target system.

Schematic diagram of simulative test environment is like the following.

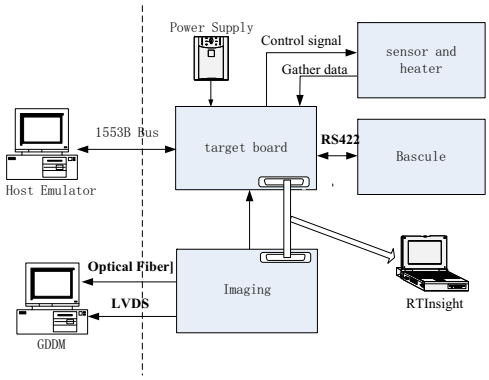

Figure 1. Schematic Diagram of Simulative Test Environment

D. Collect coverage rate data

Process of dynamic test with the assistance of RTInsight can be divided into 5 phases:

- Hardware linkage: link data bus, address bus, and control bus of target system with the interface of the test tool;
- Source Code Instrumentation: Use TestBed to make instrumentation to the source code;
- Download program: compile the profile after insertion to generate .Obj document; convert .Obj document into target board binary code, and program the binary code into the program storage of the target board;
- Set up project: when there is power operation on the target board, run RTInsight Pro on host emulator, set up-project, RTInsight Pro displays dynamically the coverage rate information of tested program through analyzing history files RTInsight has uploaded;
- Data storage: since the software is large in size, test case is large in number, and the test requires much time, collected data must be stored timely in order to make the analysis more convenient.

Diagram 2 shows the dynamic increase situation of statement coverage, branch coverage and call coverage by RTInsight Pro on host emulator during the operating process of target board.

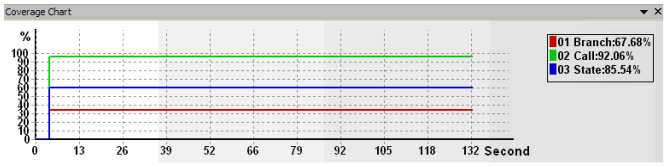

Figure 2.Dynamic coverage chart

E. Test time performance

Through setting up a dynamic test platform of embedded software, we can precisely acquire performance information of maximum operation time, minimum operation time and accumulative operation time of each function and those information between two statements at random which is concerned in the tested software.

- Initiate RTInsight Pro, set up a project, and select a function to conduct time performance analysis;
- Conduct performance instrumentation to the tested codes with relevant template, i.e. respectively add a line of assignment statements to the entrance and the end of concerned function;
- Compile the inserted codes and convert them into target board binary code, and program the binary code into the program storage of the target board;
- When there is power operation on the target board, run the tested program, and the measured value of time performance of concerned functions is shown on the interface of RTInsight Pro of host emulator.

F. Analyze the effectiveness of the test

After the completion of execution of designed test case, check code coverage result: Statement coverage 85.5%, Branch coverage 67.7%, and Call coverage 92.1%, far from the requirement of 100%.

Double click and check which statements or branches haven't been executed, and which functions haven't been invoked. Then 48 test cases were added accordingly and 7 program problems were recognized after the analysis.

After the execution of newly added test case, the result of code coverage rate is: Statement coverage 98.3%, Branch coverage 96.4%, and Call coverage 98.9%.

After artificially walkthrough on codes, the author confirmed that one part of the unexecuted codes are protected by safety protection mechanism, and can't be executed under present test environment, and the other part is redundant code. In the present test, the author identified 4 redundant codes.

Through adopting the dynamic analyzing technique of test coverage, the author identified problems which haven't been found by previous studies, and quickly and precisely pinned out the problem, hence software test efficiency is enhanced, and so is the reliability of the software.

VI. CONCLUSIONS

In the present thesis, the author, based on the particularity of real-time embedded software test, conducted the research on how to adopt the method of test coverage to evaluate the effectiveness of real-time embedded software testing, and took control software in the subsystem of a shuttle program of certain type as a case to practice, and made an introduction to the research, hoping to offer references to peer researchers in the work of real-time embedded software test with high reliability. Relevant researches need to combine many test coverage data and manage many data in the test process [6], and the author sincerely hopes that further studies on the dynamic and quantitative evaluation on the effectiveness of real-time embedded software test would be made in cooperation.

REFERENCES

[1] Sun Changai, Jin Ruoruing, Liu Chap, Jin Maozhong, "Test Techonlogy of Real Time Adn Embedded Software,"Mini-Micro System, Vo1. 21, pp. 920-924, September 2000.

[2] Li QY, Lu MY, Ruan L, "Theoretical research on software reliability testing adequacy,"Journal of Bering University of Aeronautics and Astronautics, Vo1. 29, pp. 312-3167, April 2003.

[3] Elfriede Dustin, "Effective Software Testing: 50 Specific Ways to Improve Your Testing,"Published by Addison Wesley, December 2002.

[4] LIN Xiao-yu, SHI Lei, "The Research and Realization of System Testing Model for Embedded Software," Science Technology and Engineering, Vo1. 9, pp. 7515-7519, September 2009.

[5] Wang Pu, Zhang Zhenjtan, Wang Yuxi, "A Research on Coverage−− Based Software Testing Technique in the Real Time Embedded Computer System,"Computer Engineer and Design, Vo1. 19, pp. 45-49, December 1998.

[6] An JX, Wang GQ, Li SF, Zhu JH, "Dynamic evaluation method based multi-dimensional test coverage for software testing,"Journal of Software, Vo1. 21, pp. 2135−2147,September 2010.