Automatic Verification Environment for Embedded Software Reliability Testing

Fangling Zhong, Jun Ai, Qi Ao
School of Reliability and System Engineering
Beihang University
Beijing, China
zhongwuhen@dse.buaa.edu.cn, aijun@buaa.edu.cn, doublenumber@dse.buaa.edu.cn

Abstract—**Software reliability testing (SRT) is one of the key techniques in software reliability field. It is too difficult to collect enough experiment data from SRT for analysis, since SRT is extremely time-consuming. In this paper, an automatic verification environment (AVE) is designed to execute SRT and collect the detailed testing data automatically in this paper. With the AVE, defect implanted in the software under test (SUT) can be set automatically according to whether defect is found or not; and data collected during SRT can support many researches on software reliability, such as software reliability evaluation, operational profile, and so on.**

Keywords—***Embedded Software; Reliability Testing; Verification Environment; Automatic; Defect Implanting***

I. INTRODUCTION

Now embedded software is more and more important because of the widely application of embedded system. It would lead to many damages such as system failure, economic loss and even major accident if the software was in failure. So a great deal of attention is being paid to embedded software reliability testing, with its great significance to increase the reliability. However, software reliability testing (SRT) is extremely time-consuming, and embedded software testing is usually more complex than general software testing [1]. In fact, without the related hardware and physical environment, Embedded software can't operate on. That is to say, the testing environment in embedded software testing has turn out to be the most important point.

For this characteristic of embedded software, there are some researches at home and aboard until now. For example, a system solution with the need of embedded real-time software testing environment is provided in [2]. And [3] introduces a software reliability simulation testing platform for automatically testing reliability of real-time embedded software, not only architecture but also key techniques are discussed. The major difficulty of embedded software testing in general is how to build the testing platform [4]. At present, there are some shortcomings in most of embedded software testing platforms existed: some is limited by testing validity and some by testability of target.

This paper describes an automatic verification environment (AVE) for embedded software reliability testing, which could provide a validation and support research on software reliability. At first, the requirements of AVE are analyzed, and then the key techniques such as automatic test and automatic collection of failure data are discussed. In addition, this paper also introduces the architecture and implement of AVE. Finally its feasibility would be validated by the case study.

II. REQUIREMENT ANALYSIS

Generally, SRT is extremely time-consuming and labor-intensive [5], especially for embedded software. Only if testing samples are very rich, test cases generated by random sampling can represent the usage of software more accurately [6]. However, the testing cost and time would increase with test data, which makes data collection become very hard. That's why mathematical tractability is often applied to software reliability rather than widely test. For instance, test time should be over 10000h for the software whose MTBF is 10000h, while the cost will also be large. Hence, the variation of operational profiles, reliability assessment, and the relation between them would be more difficult to study.

Consequently, it's necessary to design a verification environment for automatic reliability testing, in which the validation would be supported and relation mentioned before would be more convenient to study.

A. Functional Requirement

In order to achieve those objectives, an AVE must meet the following requirements:

a) It could execute the test automatically without person in a long time, which includes: loading test data automatically, sending and dealing with motivation automatically, and collecting output data automatically.

978-1-61284-666-8/11$26.00©2011 IEEE

b) It could determine whether the software under test (AUT) was in failure, and also could remove the failure which was found in the software reliability growth testing.

c) It could record all information during testing, including test time, test paths, and coverage of infects and so on.

d) If there comes out an abnormality which makes the test can't go on, it could automatically restart the whole environment.

Once these are implemented in the verification environment, plenty of test cases can be executed, and the experiment data for reliability assessment could be obtained even if operational profile changes continually.

B. Software Requirement

Generally, there are an AUT and the corresponding test platform in AVE.

Firstly, AUT must be a typical embedded system, with universal hardware interface, complete functions and documents. Because embedded software is the object of study. In addition, it should be easy to collect data and set automatically.

Secondly, as a key that could support automatic testing completely, the test platform must be able to realize the key techniques coordinate with AUT. So apart from the transformation of test data into motivation, test platform also should be able to collect the drawbacks from AUT synchronously, and judge whether AUT is in failure. In this way, an automatic, real-time, and closed-loop testing could be performed aiming at AUT.

III. DESIGN OF AVE

The AVE mentioned last section includes the embedded system under test and an automatic test platform (ATP). Key techniques are proposed on testing automatically, judging failure automatically, and collecting failure data automatically. The successful achievement of this environment makes subject investigating in large sample come true. Therefore, the key techniques and architecture are discussed in next two sections.

A. Key Techniques

As analyzed, there are three kinds of key techniques should be carried out: how to test automatically; how to collect failure data automatically; how to implant and manage defect. They will be discussed in detail as follows.

1) Automatic test

Automatic test is the most basic technique of AVE. It means that AVE could execute lots of test cases automatically. The test flow is clear as shown in Figure 1.

Although test data could be generated by special tool, ATP might not be able to identify them. Therefore, the test data should be converted into recognizable test script after loaded at first. Then a test script would be opened and analyzed line by line. Next ATP would communicate with AUT so that the related information can be transported. With the collected information and files, the script would be judged passed only if there is no defect found. At last the next test file should be loaded automatically. When all of the test scripts are executed, the test report could be gotten.

There should be two links between AUT and ATP, as they are independent yet indispensable to each other. One is used for transporting test input and output, and the other is transporting assistant test information, such as restart mark and test paths. For the purpose of real-time communication, there is an agreed communication protocol between them.

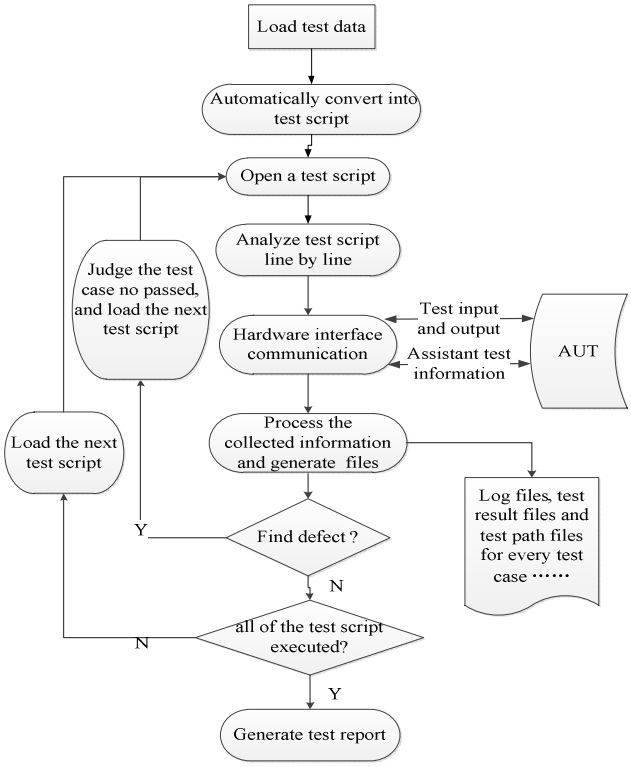

Figure 1. Automatic test flow

2) Automatic collection of failure data

This technique is also essential to the AVE, because the test and data analysis would be more convenient with it.

Some files would be generated in test process, such as log files, test result files and test path files (Figure 1.). Different kinds of files should be put into different folders with time labels. The failure data such as defect number and test time can be obtained in these files.

The related information of every test is recorded in a log file, which also should be partly displayed on the user-interface. Then the user can get basic and detail information. The auxiliary information for a test case is recorded in the

other two files. They separately include the information transported by the two links. In this way, the reason of software failure and the information of defect could be analyzed with them.

3) Implanting and management of defect

This kind of technique means implanting defect manually, and controlling whether the defect take effect or not. In other words, the source code seems not to be changed and AUT could run normally. Once the technique achieved, SRT would be executed more truly and the valid data would be obtained.

Through analyzing the typical failure mode and switch control of defect without affecting the normal operation for AUT, there would be implanting two kinds of software defect into AUT: serious defect, which might obstruct the main function of AUT; general defect, which might obstruct the secondary function of AUT.

In order to cut over AUT between correct state and implanted defect state, the logic switch should be injected at first; in the same way, some triggers should be injected into the paths where defect implanted, so as to record whether the defect is triggered. In addition, a defect configuration file would be downloaded to AUT from ATP and then the user could control whether the switch on or off. Through such way, the type and quantity of defect which need to switch on is controllable and the coverage of defect paths can be collected.

Furthermore, if there is some defect founded, it could be switched off automatically as required. In other words, besides general reliability testing, software reliability growth testing also could be executed with this technique.

B. Architecture

The AVE for embedded software reliability testing introduced in this paper includes two important parts (Figure 2.): AUT and ATP.

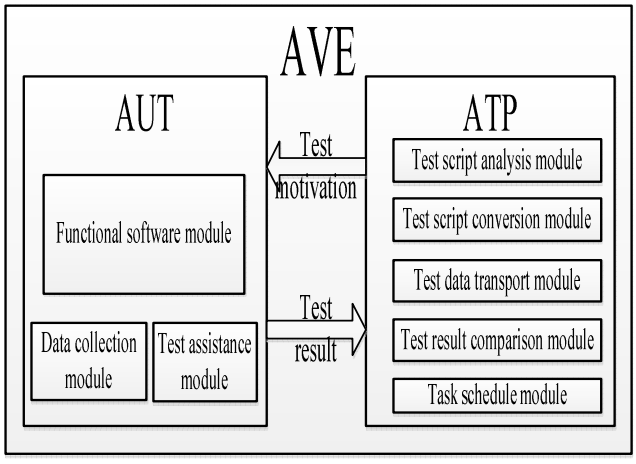

Figure 2. Architecture of AVE

1) AUT

For the purpose of supporting automatic test and data analysis being better, AUT needs to be developed allodium. As analyzed, it should include the follow modules: functional software module, data collection module and test assistance module.

a) Functional software module

This module is in charge of the specific function in AUT. So there should be all of the typical characters about embedded software in this module. When departing from AVE, the corresponding orders could be operated regularly.

b) Data collection module

This one is in charge of injecting the control switch and triggers. It's used for collecting information such as the coverage of path, branch and defect. The data analysis later would be easier with these messages.

c) Test assistance module

Test assistance module is mainly used for AUT to communicate with ATP in the test process. For example, getting the defect configuration file, and sending the collected information back to ATP.

2) ATP

To achieve the goal of subject research, a special ATP towards the special software (AUT) needs to be developed. And the ATP should include five modules at least as follows:

a) Test script analysis module

Usually there is not only the motivation but other assistant information in a test script. So it needs to choose appropriate script for reliability testing at first. On this basis, ATP should have test script analysis module, which is responsible for analyzing script automatically and picking out the motivation.

b) Test script conversion module

As mentioned before, the test data generated by tool may not be recognizable, and they should be converted automatically. So it's necessary to make a study of script conversion, in order to realize the seamless connection between data generation tool and ATP.

c) Test data transport module

When every motivation is sent to AUT through the hardware interface, there are some feedbacks sent back usually. So test data transport module is in charge of these tasks. Except real-time information being collected, there also would be generated some files mentioned before. In these files every input and output are twins, so as to analyze whether there comes cross a problem when AUT is running.

d) Test result comparison module

This module is mainly used for telling whether the AUT is in failure. If there is some defect founded, it could record

relevant information. And then the experiment data can be applied to follow-up research on reliability evaluation.

e) Task schedule module

Because there are lots of tasks in reliability testing, this module is responsible for scheduling the task automatically, so that the testing could be operated without human.

IV. IMPLEMENT OF AVE

The two parts of AVE described last section have been developed now: AUT is designed to simulate a store management system (SMS), and ATP is designed as an ATP of SMS. The structure of AVE is shown in Figure 3. .

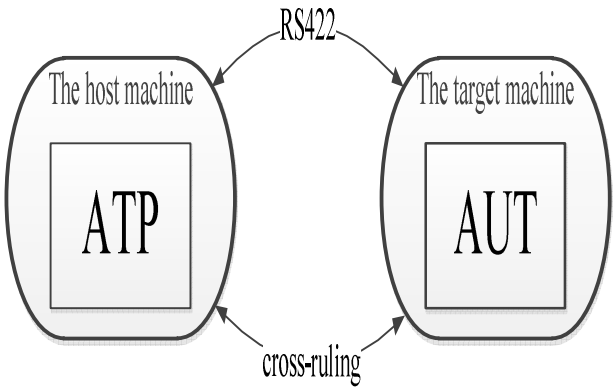

Figure 3. Implement of AVE

AUT is developed in Tornado2.2 under the host machine (PC of Windows XP) and yet running environment is the target machine (PC of VxWorks5.5). In addition, it's designed with C and mainly simulated as SMS. Usually a SMS is responsible for dealing with the orders from interconnected systems.

In order to communicate with AUT, the interconnected systems are acted by ATP. The running environment of ATP is the host machine. It is designed with C++ and the user interface is implemented through MFC.

As shown in Figure 3. , the target machine is connected to host machine with RS422 serial cable and crossover cable. The former charging serial communication, makes the control words and status words transported through it. And the latter charges network communication, so the assistant test messages are transported by it after the control words.

Besides, the defect implanted into AUT can be configured as the type and amount of defect are controllable. And the process of software reliability growth testing could be performed, since the defect can be removed after founded.

V. CASE STUDY

Currently, this AVE has been used in a subject of reliability verification. To verify its feasibility, take the reliability growth testing for example.

There are some work need to do before testing: generating 1000 test cases by TCS (a tool of reliability test data generation), and dividing them into 10 groups (100 per group); determining the deployment of defect---20 of general and 5 of serious; the testing termination condition is fixed as there are 150 test cases between two failures.

In addition, the testers shouldn't be the builders of AVE for better verification. Little know about AVE, they just need to know the usage. When AVE is ready, ATP would convert and test automatically while the tester just needs to load test data. The automatic test process is shown in Figure 4. .

In TABLE I. presented to us that the test case of No.298 is the last one appears defect, so the testing can be suspended after finishing five groups. Exactly, it should be stopped when finished No.447 according to termination condition.

TABLE I. FAILURE DATA

Defect number	Test case number	Time between failures
23	001	25s
26	001	32s
32	037	1049s
…	…	…
41	161	1676s
48	298	3663s

As it shown in TABLE II. , the automatic testing introduced in this paper could save a lot of time and energy than manual testing. It has improved the efficiency of test a lot.

TABLE II. AUTOMATIC TESTING VS. MANUAL TESTING

Test type	Test time	Tester energy
Automatic testing	6h	Preliminary loading work
Manual testing	Two weeks even more	Sustained input of human

After analyzing the collected failure data, the GO model is chosen to evaluate the reliability of AUT by SRET (a reliability assessment tool). And then come to the conclusion that the MTBF of AUT is 4854.37s. Consequently, the key techniques on collecting failure data and implanting defect are feasible. In addition, the experiment data is valid and could be support to research on software reliability.

Figure 4. Automatic Test Process

VI. CONCLUSION

Through analyzing the necessity of automatic test for reliability testing, this paper introduces an automatic verification environment (AVE). And then not only the architecture but also the key techniques are discussed, such as automatic test, automatic collection of failure data and implanting and management of defect. At last the usability of AVE is verified by practice of a total reliability growth testing.

The practice has proved that the AVE could execute plenty of test scripts automatically, which saves test time and tester energy a lot and moreover make the data collection become easier. It can be applied to not only reliability growth testing, but also other reliability studies such as validation test and accelerated test. In addition, the successful implement of AVE lays a certain foundation to follow research, which is about the relationship between operational profiles and reliability evaluation.

REFERENCES

[1] Y.H. Zhao. “Integrated test platform technology of embedded software”[D], University of Electronic Science and Technology, 2010.

[2] X.L. Cui, B. Liu, D.M. Zhong, Q. Ruan. “Design of Computer Architechture for Embedded Real-Time Software Testing Environment”[J], Control Technology, 2003.

[3] D.Y. Liu, G.X. Shen. “Study of Techniques in Embedded Software Reliability Testing Platform”[J], Electronic Measurement and Instrument, 2000, 14:580-584.

[4] W.B. Wei. “Study of dynamic testing platform for embedded software”[A], Symposium for Computer Professional Application Committee of Chinese Society of Astronautics in 2004[C], 2004.

[5] J.D. Musa. Software reliability-engineered testing. Computer, 1996,29(11):61~68.

[6] J. Clarke. “Automated test generation from a behavioral model”[R]. Presented at Software Quality Week, 1999.