The software test case design Based on the SFTA and equivalence class

Liu Wen-hong, Wu Xin
Beijing Institute of Tracking and Telecommunications Technology
Beijing, China
e-mail: wx263@139.com

***Abstract*—Effectiveness and adequacy of test case design is the important part of the research in the field of software testing. In this paper, we analysis the software failure which may occur, based on software-related documentation and data, to establishment the software fault tree and get the minimal cut sets. We use the equivalence class methods for test case design, based on the Minimal cut sets model. Based on the set of test cases Obtained by this method, we can use the minimum test cases to cover all the test requirements.**

Keywords- software testing; test case;fault tree;equivalence class

I. INTRODUCTION

With the continuous development of computer science and technology, the software play an increasingly important role in various industries, in many areas software implementation has reached 80% of the functionality of the entire system. Therefore, the quality of the software system often determines the quality of the system, and sometimes software defects can cause very serious consequences. Induced failure of software systems often result in very serious consequences. For example, [1]: radiation therapy machine software error occurred in the United States in the 1980s, leading to a serious accident which caused five patients to death by ultra measurement of radiation. With the deepening and a wide range of the software applications, especially software applied to the high-speed railway, banking, medical, military and aerospace, it need more concern on the quality of software. The consequences will be unpredictable if the accident caused by software defects that occur in these systems!

Currently, software testing is still one of the important means to ensure software quality, and one of the most critical aspects of software testing is test case design, test case design fully determine the Efficiency of the test. It is the constantly explore research topics to determinate which test case design method can meet the effectiveness and adequacy requirements of the field of software testing.

Software fault tree analysis [2] (SFTA) is a top-down software reliability and safety analysis methods. Start from the event which we do not want it to occurrence (top event), especially for the safety of personnel and equipment produce a material impact, starts down gradually to trace the reasons leading to the top event, until the basic event (the end of the event). The SFTA analysis results can determine the focus and content of the software testing.

Equivalence class method is a black box testing method [3], it divide the software input into a number of data classes to get the input of the testcase. Equivalence class test case design is based on the assessment of the equivalence classes of the input conditions, and is one of the most commonly used method in the software black box testing.

We presented an effective method of test case design based on the combine of SFTA and equivalence class method, and achieve the automatic generation of test cases.

II. SOFTWARE FAULT TREE AND MINIMAL CUT SETS

The SFTA technology is widely used software safety analysis method, it is mainly used for software black box test case design in software testing. For those highly reliable software, we design the test cases based on the safety analysis to achieve effective coverage of the test cases.

A. Establish a software fault tree

The establishment of the software fault tree is the most basic and critical task in software fault tree analysis. In simple terms, software fault tree is built by some logic and event symbols.

Because the accuracy of the software fault tree directly affects the analysis of software, we need to carry out the necessary preparatory work in building software fault tree software. The establishment of the software fault tree usually include: the collection and analysis of relevant technical information, select the top events and build a fault tree analysis.

1) the collection and analysis of relevant technical information

The degree of perfection of the software fault tree directly impact on the Efficiency of the set of test cases based on the minimal cut sets, and thus requires a wide range of relevant knowledge and experience. These knowledge and experience rely mainly on the learning of the relevant information and familiarity with the software, the content need to master should include:

a) the architecture of the software system design, the functionality of the software, the scope of the system, interface between the software and operating environment, etc;

b) to identify the impact of human factors on software system;

c) To identify the state of the software in different operation modes, as well as mutual conversion relationship between the different modes.

In addition, we should take counsel with experienced designers and users when we establish the fault tree, in order to ensure the correctness of the software fault tree.

Published by Atlantis Press, Paris, France.

For higher level softwar, in general, the software developer has completed the software security analysis, and further analysis can be carried out on the basis of it.

2) selecting the top event to analysis

The problem often encountered in the use of fault tree is the fault tree is too complex to be used effectively in test case design. To avoid this problem, we can build the fault tree in layers according with single function unit, then gradually extended to the top events.

This method also meets the actual requirements in the confirmation test for all functions. Top events can be determined with the completion of the test items of confirmation testing. When a test item contains multiple sub-test items, sub-test items can be used as the top event.

This method not only avoids the problem of the over complexity of the software fault tree, but also according to the needs and processes of the test work. Software fault tree analysis and decomposition of the test items, test case design is closely integrated, more conducive to the automatic generation of test cases.

3) build a fault tree

Software fault tree symbols include event symbols and logic gate symbols. The event symbol is used to indicate the failure event, and the logic gate symbols is used to indicate the logical relationship between the failure event.

Software fault tree usually uses deductive method. The so-called deductive method firstly select top event to be analysed (ie, the fault event which is not wanted to happen) as the "root" of the fault tree. And then analyze the direct cause of the top events (including all of the events or conditions), and connected to the appropriate logic gates with top event, as Fault Tree node(the middle of the event). According to this method, until traced back all reasons which caused the top events (underlying event). The underlying event known as the end of the event, constitute the leaves of fault tree.

The end event in the bottom of the fault tree is the root cause of the top events. Some end event can lead to the top event independently, some end events according to certain logical relationship to lead to the top event. In the causes of analysing of failure, the experience of the analyst, software development group and software testing institutions should be fully exerted.

B. mathematical description of the fault tree

Suppose a fault tree constituted of N end events, and the top event of the fault tree is T, xi is the end of the event state variables, the value of xi is 1 or 0, Φ indicates the state of the top event, defined as follows

$$x_i = \begin{cases} 1 & \text{End event i happens} \\ 0 & \text{End event i not happens} \end{cases}$$

$$\Phi = \begin{cases} 1 & \text{top event happens} \\ 0 & \text{top event not happens} \end{cases}$$

Φ is determined by the state of the bottom events, i.e., $\Phi = \Phi(X)$, wherein $X = \{x_1, x_2, \ldots\ldots, x_i\}$. $\Phi = \Phi(X)$is the fault tree structure function of the mathematical formulation [4].

a) for the fault tree consisting of all the elements of logical AND gate between the top event and the end event, the structure function can be expressed as follows:

$$\Phi(x) = \bigcap_{i=1}^{n} x_i \ (i = 1, 2, \cdots, n)$$

b) for the fault tree consisting of all the elements of logical OR gate between the top event and the end event, the structure function can be expressed as follows::

$$\Phi(x) = \bigcup_{i=1}^{n} x_i \ (i = 1, 2, \cdots, n)$$

The structure function of a fault tree can be represent by the above definition. The structure function of the software fault tree shown in Figure 1 can be expressed as:

$$\Phi = \left[x_1 \bigcup (x_2 \bigcap x_3) \right] \bigcap \left[x_1 \bigcup x_4 \bigcup (x_3 \bigcap x_5) \right]$$

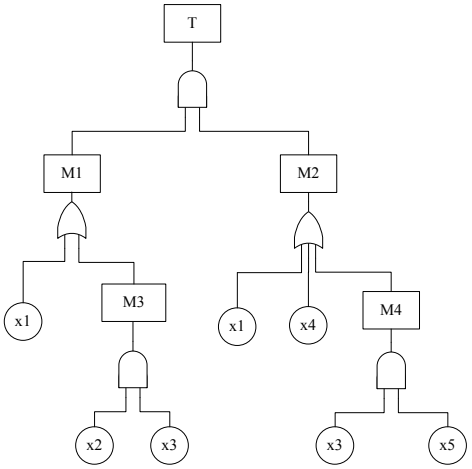

Figure 1. software fault tree

C. mathematical description of the minimal cut sets

Cutset: a set of end events which can cause the top event.

Minimal cut sets: the cut set which does not contain any redundant factors. If you get rid of any event of the minimum cutset, it is no longer a cut set.

According to the above definition, in the software fault tree,when the minimal cut set occurs, the top event must occur. Therefore, all minimum cut sets of a fault tree represent all the possibilities of the top event. Therefore, if the software fault tree Contains m minimum cutset C=(C1，C2，……，Cm), the top event of the fault tree must have occurred when all end events of the minimum cut set occurs, minimal cut sets can be represented as follows:

$$C_j = \bigcap_{i=1} x_i$$

When any of the minimal cut set in the Collection of m minimal cut set occurs, the top event occurs, so the software fault tree can be expressed as

$$\Phi = \bigcup_{j=1}^{m} \bigcap_{i=1}^{n} x_i$$

According to the definition of the minimal cut sets of the fault tree, using the algorithm of Fuseell-Vesely, minimal cut sets of Figure 1 can be obtained: $\{x_1\}$、$\{x_2$、x_3、$x_4\}$、$\{x_2$、x_3、$x_5\}$, its software fault tree can be expressed as:

$$\Phi = x_1 \bigcup (x_2 \bigcap x_3 \bigcap x_4) \bigcup (x_2 \bigcap x_3 \bigcap x_5)$$

According to the above expression, the equivalence software fault tree of Figure 1 is shown in Figure 2.

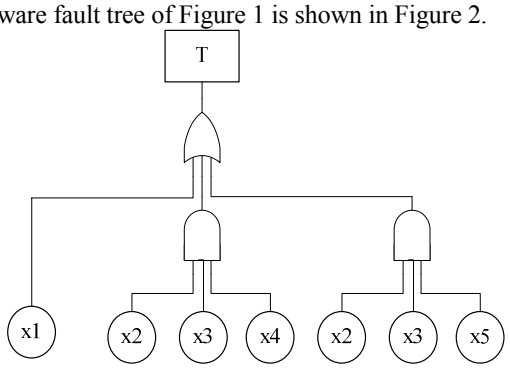

Figure 2. minimal cut set represents the equivalent software fault tree

III. GENERATES A SET OF TEST CASES

The test case design model is built by minimal cut sets of the software fault tree, each end event is the input of the test case, and each input's value should be based on the typical value by the principle of equivalence classes.

A. Determine the equivalence classes of input conditions

When the typical value of the input is selected, it should be determined according to the method of equivalence class. Equivalence classes should be determined by the following principles [3]:

1) If the input condition specifies a range, you can define a valid equivalence class and two invalid equivalence class;

2) If the input condition requires a specific value, you can define a valid equivalence class and two invalid equivalence classes;

3) If the input conditions specified an element of a set, you can define a valid equivalence class and one invalid equivalence class;

4) If the input condition is a Boolean value, you can define a valid equivalence class and one invalid equivalence class.

For the example of a minimum cut set $\{x_2,x_3,x_4\}$, the typical value is: x_2:a_1,a_2,a_3; x_3:b_1,b_2,b_3; x_4:c_1,c_2.

B. generate a set of test cases

In test case design, a minimum cut sets for a set of test cases. A case in point is a minimum cut set {x2，x3，x4}, based by the each equivalence class of the x2，x3，x4 which determined in a),the test case are: {a1，b1，c1}，{a1，b1，c2}，{a1，b2，c1}，{a1，b2，c2}，{a1，b3，c1}，{a1，b3，c2}，{a2，b1，c1}，{a2，b1，c2}，{a2，b2，c1}，{a2，b2，c2}，{a2，b3，c1}，{a2，b3，c2}，{a3，b1，c1}，{a3，b1，c2}，{a3，b2，c1}，{a3，b2，c2}，{a3，b3，c1}，{a3，b3，c2}. So that the number of the test case you get is $C_3^1 \times C_3^1 \times C_2^1 = 3 \times 3 \times 2 = 18$.

IV. APPLICATION EXAMPLES

(1) Get the software fault tree. In practical applications, we got the software fault tree shown in Figure 3, wherein the occurrence of the intermediate event M3 occurs by any two or more of the x3,x4,x5 resulting. Under such circumstances, the following diagram transformation equivalent fault tree is shown in Figure 4.

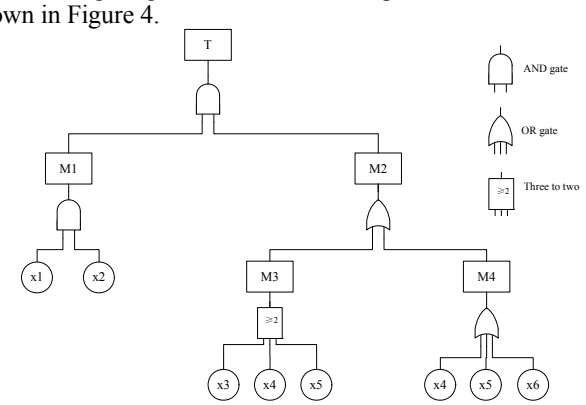

Figure 3. software fault tree

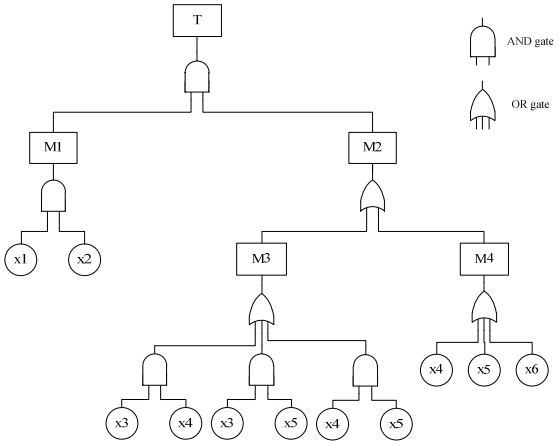

Figure 4. improved software fault tree

(2) Get the cut sets: {x1, x2, x3, x4 },{x1, x2, x3, x5},{x1,x2, x4, x5},{x1, x2, x4 },{x1, x2, , x5},{x1,x2,x6}.

(3) Get the minimal cut sets: {x1, x2, x4 },{x1, x2, , x5},{x1，x2，x6}.

(4) Set the typical value. A typical value is set according to the principle of equivalence classes of input conditions. Shown as Table I.

TABLE I. TYPICAL VALUE SET

x1	valid,invalid
x2	valid,invalid
x3	valid,invalid
x4	valid,invalid
x5	valid,invalid
x6	valid,invalid1,invalid2

(5) generate test cases.

The number of test case based on the cut sets is 78, and the number of test case based on the minimal cut sets is 28.

A case in point based on the minimum cut set $\{x_1, x_2, x_4\}$ is shown in Figure 5.

1	x1、x2、x4	valid(x1)、valid(x2)、valid(x4)
2		valid(x1)、valid(x2)、invalid(x4)
3		valid(x1)、invalid(x2)、valid(x4)
4		valid(x1)、invalid(x2)、invalid(x4)
5		invalid(x1)、valid(x2)、valid(x4)
6		invalid(x1)、valid(x2)、invalid(x4)
7		invalid(x1)、invalid(x2)、valid(x4)
8		invalid(x1)、invalid(x2)、invalid(x4)

Figure 5. test cases list based on one cut set

V. CONCLUSION

In this paper, based on the method and principle of fault tree, we use the minimum cut sets as the test case generation model, select the input of the test case by the method of the equivalence classes, present a method for generating test case based on the minimal cut sets of fault trees and equivalence class, and develop a automated test case generation tool.

The method has been applied in the escape software testing projects, the actual results show that it is effective to improve the adequacy degree of automation and high reliability software testing test case design.

REFERENCES

[1] Huang Xizi,2002, software reliability, security and quality assurance, Electronics Industry Press, Beijing 2002.10

[2] Lu Tingxiao, Zheng Pengzhou, He Guowei, Zheng Shengkui, 1995 reliability design and analysis, Beijing: National Defense Industry Press, 1995

[3] Roger S. Pressman.Software Engineering A Practitioner's Approach Seventh Edition, The McGraw-Hill Companies, Inc ..

[4] Sun Zhian, Pei Xiaoli, Song Xin, Dai Zhongjian, 2009, Software Reliability Engineering, Beijing: Beijing University of Aeronautics and Astronautics Press, 2009.3