Research on software safety growth testing method based on FTPM model

Zhongxiao Ji[1, a], Guohua Jiang[2, b]

[1]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[2]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

[a]email:Jacknuaa@163.com, [b]email:2019151095@qq.com

Keywords: Safety Testing; Fault Tree Analysis; Markov Model; Running; Scenario

Abstract. In the key areas of safety, the safety requirements of the software are becoming higher and higher. However, there are all kinds of safety testing methods, or the presence of state space explosion, or the test case is huge, and the safety testing efficiency is low. In order to solve this problem, this paper proposes a safety testing method, which is based on FAPM model. Getting danger of hazardous events which is through using dynamic fault tree analysis. And it is based on the call graph of the program to build the mapping relationship between basic events and operation, operation and code. Then, build a scene using a Markov model. According to the model generation of safety test case, and according to the test stop criterion to judge the adequacy of the testing.

Introduction

Along with the rapidly development and progress of the information, the application of software has been extended to various field of our social life, including the financial industry, the power industry, the newly aerospace and the medical and health services. Besides, software play an increasingly essential role in the control of the safety critical system, in the field which required much higher security. Once these fields which are involved in safe functions lose control, it may cause inestimable safety accidents; this software is called safety critical software. In 1983, the Soviet missile early warning system failure occurs, which nearly leads to the third world war; In 2009, one reason of the network incident of six provinces in the south of China may be caused by storm player software mass user requests. Therefore, it is very necessary to test the safety of the critical software security. At present, for safety critical software test, there are a lot of researches in theory, including the study of the test framework and test methods; For example, the article[1] proposes a new security analysis framework, and software security key test; Article[2] reviews the safety testing methods and analysis of the safety testing standards. However, the methods in practical engineering are not abundant, or there are many serious problems in the state space explosion or the great amount of test cases. This paper mainly puts forward a model based on FAPM security testing methods, to improve the efficiency of safety testing and reduce the number of test cases.

The current methods and the disadvantages of the software safety testing

Software security means that the software running does not cause the ability of system hazards [3]. Although software cannot lead to severe danger directly; But the software failure can result in system failure directly, thus leading to the damage of life property and environment. The first time of Software security (software safety) appeared was in 1979, when the American released the military standard MIL-STD-1574A[4]. As the application software is becoming more and more widely used, the safety of software is also got more attentions in our daily life. Depending on the different concerns, software Security Testing can be in divided into the following aspects: software failure Safety Testing and software confidential Security Testing. This article mainly focuses on the software failure safety testing. For software failure safety tests part, the commonly used test methods is safety testing fault tree analysis methods and the Petri net safety testing method.

The safety testing method is based on fault tree analysis. This method is basis of the conditions that may lead to the danger and failure of minimum cut sets, to generate the safety cases test, and ensure the adequacy of safety testing; but it will take a lot of energy and larger workload. Besides, this method did not cover the risk of danger or fault degree, but analysis every risk which will bring a lot of repetitive work.

Safety testing method based on Petri net, which mainly use the advantages of Petri net i.e. intuitionistic and visual, building test cases of software security testing. This method is searching out the hazardous states in the state table, constructing a state transition sequence which transfer from the initial state to a hazardous state, and the state transition sequence is a safety test cases; next, generate software safety test suite according to this method. This method needs to spend a lot of energy in the process of generating the reachability graph, and it may appear the problem of state explosion.

Safety Testing Process Based On the FTPM Model

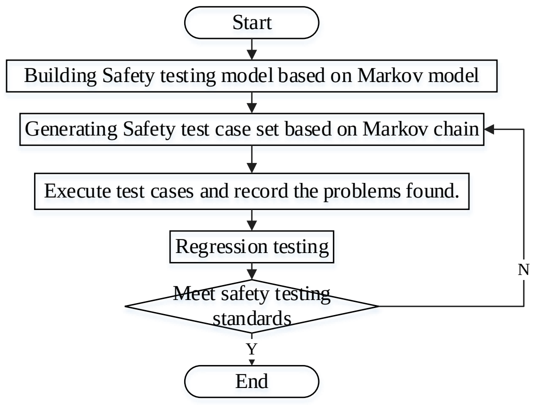
Through analyzing the common security testing methods, we can find that each type of safety testing methods have some problems, such as security testing method based on Petri net has the problem of state explosion. However, these methods have their unique advantages, e.g. the fault tree analysis method analysis can fully meet the testing requirement, the Petri net is intuitionistic and visual; As a result, integrating a variety of methods, generating reasonable safety testing strategy is worth considering. The safety testing method (Fault Tree Profile Markov, FTPM) proposed in this paper combines the DFTA, safety association profile and Markov using model. Combining with the adequacy of FTA, test purpose of safety association profile and the advantages of the correlation of Markov model. This method is mainly based on improve the efficiency of safety testing and reduce system risk quickly, find out the higher safety risk system path through building safety test model, execute safety testing aim at those paths can reduce the risk of the system quickly. It is purposeful and high-efficiency. The basic framework of the method, as shown in figure 1.

Constructed FTPM Safety Testing Model

FTPM model is based on risk, not only focus on the risk occurrence probability, and focus on the criticality caused by the risk. FTPM model building process, as shown below:

(1) Use the method of DFTA to obtain hazard and basic events cause the risk;
(2) Analyze of the basic events to obtain the corresponding software running;
(3) Build Program call graph, on this basis to build the software running transfer graph;
(4) According to the risk of running, determine the key running and key scenes;
(5) Build Markov using model based on scenario;
(6) According to the risk of each scenario chain, determine the test order of Markov chains.

Obtain the Hazard Events. Hazard event(HE) is made up of one or more of the basic event according to certain order, consisting of a collection of basic events which may lead to hazard; A subset of the collection of the basic event system. Several basic events occurred in a certain way and order, the system may be dangerous. At present, FTA is a safety analysis techniques, widely applied in the system safety analysis. FTA provides an intuitive and simple way to descript of the system risk. However, the traditional FTA used in the analysis of dynamic system, being unable to describe the system behaviors which depend on time and sequence order. To solve this problem, the scholars in the field extend the fault tree analysis, get the dynamic fault tree (DFT). DFT is mainly on the basis of fault tree, add some logic gate which can represent the order of sequence, mainly PAND, function-correlated gate and so on. Literature [6] by introducing dynamic logical gate such as PAND and function-correlated gate, extending traditional fault tree to the dynamic fault tree, so that the traditional fault tree has the ability of safety analysis was carried out on the dynamic system. Using the common software fault identification method, such as FMEA, PHA to obtain a list of dangerous or malfunction of the system, Then, each item in the list was executed the safety analysis using the dynamic fault tree analysis method, get the hazardous and basic events cause risks. Figure 2 is a dynamic fault tree model, in the fault tree, H is risk events, G1 to G4 are intermediate events, A1 to A6 are basic events; between the basic events are mutually independent and between intermediate events G3 and intermediate events G4 exist a logical order.


Start
Building Safety testing model based on Markov model
Generating Safety test case set based on Markov chain
Execute test cases and record the problems found.
Regression testing
Meet safety testing standards
N
Y
End
Fig.1 Safety testing process based on FTRM model

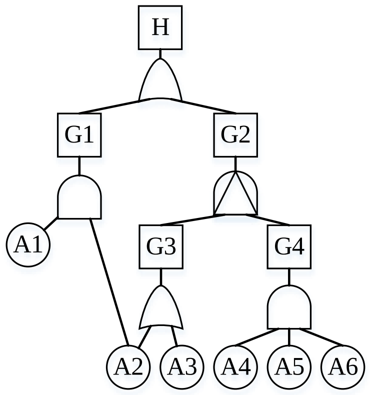


Fig.2 Dynamic fault tree model

Through the analysis of dynamic fault tree model, we can get the hazard events cause hazard H, i.e. minimum cut set. The PAND in the dynamic fault tree have an order relation, namely only G3 and G4 happened first, then G2 can happen; Using the top-down method of fault tree search algorithm, i.e. Fussell algorithm to generate the minimum cut set. And the minimum cut set generated as shown in table 1.

Table 1 Fussell algorithm for solving the minimum cut set table

1	2	3	4	minimum cut set
H	G1	(A1,A2)	(A1,A2)	(A1,A2)
	G2	$\overrightarrow{(G3,G4)}$	$\overrightarrow{(A2,G4)}$	$\overrightarrow{(A2,(A4,A5,A6))}$
			$\overrightarrow{(A3,G4)}$	$\overrightarrow{(A3,(A4,A5,A6))}$

According to the definition of hazard events, each of the minimum cut sets is a hazard event. In table1, $\overrightarrow{(G3,G4)}$ means G3 happened before G4 happen; $\overrightarrow{(A2,(A4,A5,A6))}$ means A2 happened before A4, A5 and A6, but between A4, A5, A6 does not exist an order.

Mapping of Code to Running Based Procedure Call Graph. Procedure call graph is a kind of graph which describes the software system structure. It can describe the call relationship between functions clearly and intuitively. It also can provides a lot of help to analysis and test the software. At present, there are a lot of researches on method of procedure call graph building. Literature [7] proposes a process-oriented program call graph generation algorithm and implement. In the process of building a FTPM model, based on procedure call graph to build the transfer graph of software running. Using software test tools LDRA_Testbed to generate procedure call graph in process. Transfer diagram process which transfer from code to software running, as shown below:

(1) Generate software system call graph through the software test tools LDRA_Testbed;

(2) Traverse call graph, instrument the function, generating function execution path graph;

(3) Traversal function execution path graph, combined with the software requirements and design, generate software running transfer diagram.

Next, use a simplified civil aviation aircraft landing gear system [8] as an example, introduced the transfer process which transfer from code to software running. First of all, using LDRA_Testbed analysis software code, generate procedure call graph, as shown in figure 3; Secondly, traverse gear detection system function call graph, do syntax analysis for the node which out-degree is greater than or equal to 2. Plug code into the beginning, ending and branch location. Generate specific function execution path diagram, as shown in figure 4(---►Grammar analysis but not a logical branch,→ No grammar analysis,⟶Grammar analysis and a logical branch). Finally, traverse function execution path diagram of the system, combining with the system requirements and design documents, generate software running transfer diagram, as shown in figure 5.

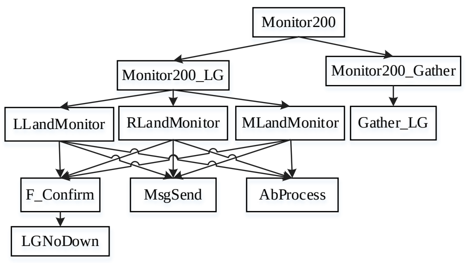


Fig.3 Function call graph

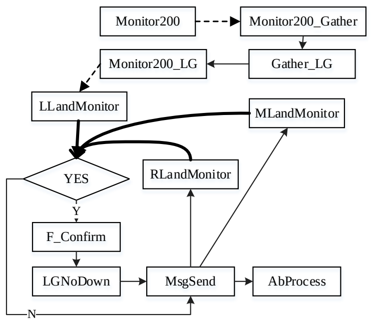


Fig.4 Function implementation path graph

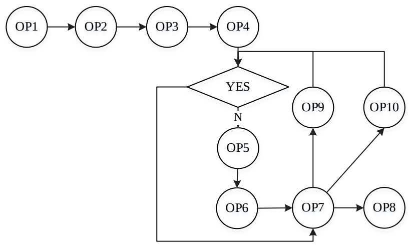


Fig.5 Software running graph

Mapping Relationship between Software Running and Basic Events. Definition one: Basic Event (BE) is the atomic event which cause danger happening. It's inseparable in the event level. The basic event can be expressed in a triple, i.e. $\boldsymbol{EB = (EB\_des, pre, post)}$. $\boldsymbol{EB\_des}$ is the description of the basic event; $\boldsymbol{pre}$ is the necessary condition of basic event trigger; $\boldsymbol{post}$ is the result of the basic event happened.

Definition two: Software Action (SA) is the function completed in accordance with the rules.

Definition three: Software Operation (OP) is the independent action completed according to certain specifications. OP can be expressed in a triple, i.e. $OP = (p, \varepsilon, F)$. p is the probability of a software running occurrence; ε is the criticality when a running failed; F is a software action this OP belong to.

SA is the function of the software. Each SA may be the results of the multiple software running. SA, therefore, is a collection of the running, can be represented as: $SA = (SA\_des, OP\_set, PreS, PostS)$, SA_des is The description of the SA; OP_set is a collection of multiple software running, i.e. $(op_1, op_2, \dots\dots, op_n)$; $PreS$ is a collection of software states which the SA need to meet, any condition in the collection met, can perform the action; $PostS$ a collection of software states after the SA performed.

The basic events and software actions are gained through the system requirements and design documents, descriptions of both are used in natural language. So, there is no way to automate matching. We need to implement by artificial matching. The matching rules between them, is as follows:

(1) When the description of the basic event and the description of software action is consistent in the semantic level; pre and $post$ in basic event respectively belong to $PreS$ and $PostS$ in software action. Then, we can consider that there is a corresponding relationship between basic events and software action, expressed as (EB, SA).

(2) When the description of the basic event and the description of software action is inconsistent in the semantic level, we can say that there is not a corresponding relationship between basic events and software action.

Because of the natural language exists ambiguity, the corresponding relationship between basic events and software action is a one-to-many or many-to-many relationship instead of a simple one-to-one relationship. Therefore, between the basic events and software running is also a one-to-many or many-to-many relationship. A ground safety management system as an example. There is a basic event in the system, i.e. $A = (des, PreA, PostA)$; two software action: $B_1 = (B_1\_des, OP_1\_set, PreS_1, PostS_1$ and $B_2 = (B_2\_des, OP_2\_set, PreS_2, PostS_2$, among them, the corresponding two collection of software running: $OP_1\_set = \{b_{11}, b_{12}, \dots\dots, b_{1n}\}$ and $OP_2\_set = \{b_{21}, b_{22}, \dots\dots, b_{2n}\}$. Two assumptions: B_1_des,B_2_des and des are consistent in the semantic level; $PreA \in PreS_1$, $PreA \in PreS_2$, $PostA \in PostS_1$, $PostA \in PostS_2$. There is a corresponding relationship between a basic event and software action, i.e. (A, B_1)、(A, B_2). SA is a implementation of a software running, consists of software running. There is a corresponding relationship basic event and software running. As A result, the corresponding relationship between the basic event A and running as: (A, b_{11}), (A, b_{12}), ⋯⋯, (A, b_{1n}), (A, b_{21}), (A, b_{22}), ⋯⋯, (A, b_{2n}); ; According to the different risk of each running, we can determine the key running which have higher risk and correspond to the basic event, i.e. $(A, b_{12})$and(A, b_{21}).

Building Scenarios Used Markov Model. Through the analysis of the Operation transfer diagram, you can get the operation of the whole module or system. However, because of the different risk of each operation, the safety testing priority also not the same. Operation which the risk coefficient is larger, is called a safety key running. Due to software safety testing, focus on the risk of the whole system; If only focus on the risk of single operation, requires a large number of test cases, spend a lot of time and energy, bring huge workload for the assessment of the whole system, and also have a little help for the risk assessment of the system. Therefore, you need to order these key operations according to certain order to constitute an execution path from initial state to the end state of the system, and then do the security testing. In this way, not only is there a purpose for safety testing, and also can meet the goal of security testing quickly. So you need to make as much as possible the key operation concentrated on a security testing path.

We can summarize software action each running belongs to according to the transfer diagram of running and the execution graph of functions. Obviously, the software action is a collection of the running. At the same time, the running of software action, combined in different ways, made up the different scenarios. And the scenarios is the specific implementation of software actions. Among them, the states which execute the action need to meet is the prepositive states of the scenarios; after the action execution, the software states reached is the postpositive states of the scenarios. Assume that each action execution result is unique, so the software reaches the state is also unique.

Definition 4: more than one run in accordance with certain rules, the external performance of a clear, known as the scene [9]. The software state is called as scene, which is constituted by some running in accordance with certain rules and has explicit external performance. The scene can be represented as a four tuple $SCEN = (OP, Pre, Post, Trans)$;In which, OP: represents a sequence of operations in a certain rule; Pre : represents the pre-condition of the scene, that is, the condition for the execution of the scene; $Post$: represents the post state of the scene, that is, the state of the system after carrying on the scene; $Trans$: represents a mapping, $Pre \times Post \to [0,1]$,and $trans(s,t)=p$represents that the possibility that execute the scene and satisfy the system's post condition" t" after satisfying the pre-condition" s" is p.

Nature 1: the various operations constituting the scene are independent to each other.

Nature 2: the probability of a transfer of each scene is smaller than or equal to 1 and the sum of all the probabilities of transfer to the next scene is 1, as $\sum_{i=1}^{n} p_i = 1$; and p_i represents the probability of the scene transferring to the scenario i.

Because the scene is a sequence of operations, the operation sequence from the initial state of the system to the termination, without an explicit external performance, can be transformed into a scene chain from the initial scene to the termination scene. The link methods of the various scenes are mainly refer to Markov model commonly used in the software reliability testing [10].The generation process of the Markov which is scene level will be introduced through the Markov formation process of the landing gear detection system described above. Assuming that the plane is currently in the air, the scenes are satisfied with the table 2.

Table 2 The pre-condition and post condition that each scene meets

Scene	Pre-condition	Post condition
SCEN1	PreL	L_OK
SCEN2	PreL	No
SCEN3	L_OK	R_OK
SCEN4	L_OK	No
SCEN5	R_OK	Yes
SCEN6	R_OK	No

Note: SCEN1 to SCEN6 represents six different states of the landing gear detection system.

According to the conditions of the scene and the basic nature of the Markov process, we can generate such a scene level Markov model, shown in figure 6.

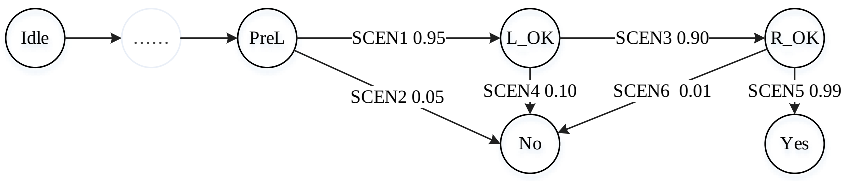


Figure 6 The Markov model of landing gear detection system

Generate safety test cases. According to the difference of the probability the operation occurs and the criticality the failure causes, the operation can be divided into two kinds of sets: safety critical operation and common operation. Safety critical operation is the focus of the safety testing. But a single key operation can not necessarily lead to the occurrence of dangerous. Only a hazardous event can lead to the danger happening. Those scenes including critical operations are called as critical scene. Because of the difference of the number of critical operation, the criticality of critical scene and the degree of safety testing will be different. The steps for safety testing are as follow:

(1) According to the probability that each critical operation in the scene occurs, the critical probability that the scene occurs is calculated: $p = \prod_{i=1}^{n} p_i$, So, we can get the risk of each scene: $\text{Risk} = p * \varepsilon = \prod_{i=1}^{n} p_i * \max\{\varepsilon_1, \varepsilon_2, \dots\dots, \varepsilon_n\}$.

(2) According to the determined critical event and the risk of each scene, it may be possible to link as many hazardous events as possible to a Markov chain;

(3) Confirm input and predict output for the critical operation of the determined Markov chain.

Test Termination Criteria. Before safety testing, the risk of each Markov chain in the Markov model has been calculated, and the test has been sorted according to the priority. The higher the risk value, the higher the priority.

Guideline 1: Choose the Markov chain with the highest risk value to carry on testing each time.

Guideline 2: After each test, modify the key operation with the highest risk in the Markov chain; Then set the probability that the operation can also lead to the risk as p; Calculate the risk of each Markov chain again $Risk_i$.

Guideline 3: If the risk value of the whole system,$Risk_{all}$, is smaller than the safety testing standard, $Risk$, the safety testing will stop. Among them, $Risk_{all}$ is the sum of the risk of each Markov chain, $Risk_{all} = \sum_{i=1}^{n} Risk_i$.

Conclusion

The purpose of safety testing is to find and solve the security flaws whose risk is bigger, and this shows safety testing has strong pertinence; But a single safety testing method, for example the safety testing method based on fault tree, has some problems, such as the undefined purpose, the space explosion state and so on. Therefore, we need to combine a variety of methods to enhance the advantages and avoid the disadvantages. In this paper, the testing method based on the safety testing model, FTPM, has a strong purpose, can find the security flaws existing in the software quickly, and need few safety testing cases; It is efficient and saves the test time and other resources. A single safety testing method has various shortcomings, so, the safety testing combined with a variety of methods will be one of the focus of the future research.

References

[1] Medikonda BS, Panchumarthy SR. A framework for software safety in safety-critical systems. ACM SIGSOFT Software Engineering Notes, 2009, 34(2):1−9. [doi: 10.1145/1507195.1507207]

[2] Huang ZQ, Xu BF, Kan SL, Hu J, Chen Z. Survey on embedded software safety analysis standards, methods and tools for airborne system. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):200−218 (in Chinese). http://www.jos.org.cn/1000-9825/4530.htm

[3] Athalye P, Maksimovic D, Erickson R. High-Performance front-end converter for avionics applications. IEEE Trans. On Aerospace and Electronic Systems, 2003,39(2):462−470. [doi:

10.1109/TAES.2003.1207258]

[4] USAF. MIL-STD-1574A. System safety program for space and missile system. Arlington: Department of Defence, 1979.

[5] Oded Tal, Scott Knight, Tom Dean: Syntax-based Vulnerability Testing of Frame-based Network Protocols. In proc. Second Annual Conference on Privacy, Security and Trust, October 13-15, 2004, Wu Centre, University of New Brunswick, Fredericton, New Brunswick, Canada (PST 2004).pp.155-160.

[6] Dugan J B, Bavuso S J, Boyd M A. Dynamic fault-tree models for fault-tolerant computer systems. IEEE Transactions on Reliability, 1992,41(3):363-377.

[7] XIE Gang. Design and implementation of C program call graph construction algorithms [J]. Journal of Guizhou Normal University (Natural Sciences),2009,(4):77-80.

[8] Wei Xiaohui. Dynamic Analysis of Aircraft Landing Impact and Vibration Attenuating Techniques [D].Nanjing: Nanjing University of Aeronautics and Astronautics，2005

[9] Johannes Ryser, Martin Glinz. A Scenario-Based Approach to Validating and Testing Software Systems Using Statecharts. Presented at the 12th International Conference on Software and Systems Engineering and their Applications ICSSEA'99. Proceedings: CNAM, Paris, France.

[10] Yan J, Wang J, Chen HW. Deriving software Markov chain usage model from UML models. Journal of Software, 2005,16(8):1386−1394. DOI: 10.1360/jos161386