Automatic analysis technology for aviation equipment software requirements

ZHOUHan-Qing, LI Hai-Feng,HUANG Yan-Bing

Quality Engineering Technology Center, China Aeronautics Polytechnology Establishment, Beijing 100028, China

email:zhouhanqing_hk@sina.com

Keywords: aviation equipment software, requirement safety, requirement modeling, safety analysis rule, safety analysis automatically

Abstract: Quality of software requirement is an important factor to safety of aviation equipment software. With the increase of software complexity, Artificial software requirement analysis is difficult to find requirement defect caused by multi state combination, multi fault concurrent, multi condition conflict and multi path migration. This paper presents an automatic analysis technology for aviation equipment software requirement safety. First of all, formal modeling of software requirement and extracting safety analysis rules from the failure data are introduced. After that, how to analyze the interfaces, functions and states information automatically in requirement model based on analysis rules and requirement model is discussed. A platform based on this technology is developed and applied to a certain type of aviation engine control software safety analysis project successfully.

Introduction

In recent years, over 60% of the failures in the verification test and the field application of the Chinese aeronautic equipment are resulted from the problems in the software requirement phase, so the software requirement analysis [1][2] is the key link of the whole software security analysis work. In European and American developed countries, the software requirement security is analyzed in the equipment software research and development process in order to improve the software requirement quality, and many methods are accordingly generated for such analysis[3-4]. At present, the aeronautic airworthiness certification standard system proposed by the Radio Technical Commission for Aeronautics is widely applied in the aeronautic field. Specifically, the model-driven formal method is clearly determined in the newest RTCA/DO-178C[5] on the basis of DO-178B[6]as the recommendable measure for analyzing and verifying the airborne software. Additionally, the European and American mature aeronautic equipment software security engineering experience shows that the effective approach for improving the software security is to take the software failure data as the engineering experience to guide the aeronautic equipment software requirement formulation process. Compared with the airborne software security analysis work in foreign countries, although relatively complete standard and normative systems are formed in China, a large gap still exists in the aspects of standard understanding, technical support, tool & method, engineering application, etc. The prominent problems in the following two aspects are necessitated to be solved: firstly, the traditional security analysis methods excessively depend on personal experiences and have low analysis efficiency and low analysis effect for the aeronautic equipment software requirement with complex logics. Secondly, the aeronautic equipment software security analysis work is still in the initial stage in China, without any engineering experience database, and the past achievements cannot be applied in the security analysis of the aeronautic equipment software with similar type or function, thus causing the past requirement defects to appear in the new software requirement again. Therefore, it is important and urgent to research and realize the automatic analysis technique for the software requirement security.

Embedded Software Security Analysis Oriented Software Requirement Modeling Technique

At present, GJB-438B is universally adopted as the requirement document standard for the military software development in China. The requirement model elements in GJB-438B are combined with the requirements for the software requirement security analysis to research and realize the embedded software security analysis oriented software requirement modeling technique.

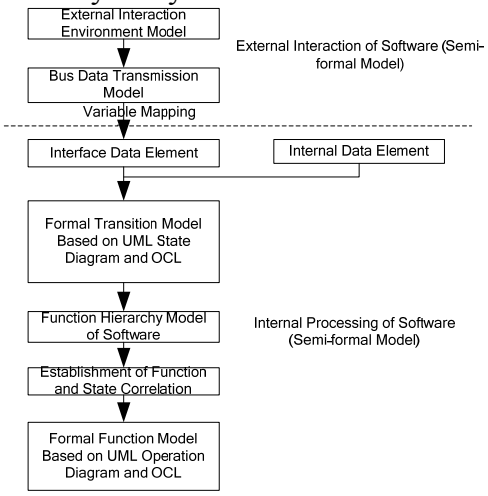

Figure 1 Requirement Modeling Process

The general requirement modeling process is as shown in Figure 1. Firstly, the software system is taken as the starting point to establish the external interaction environment model in order to clearly define the data interaction and the execution & control as well as other static and dynamic characteristics between the software (and the system thereof) and the external interaction equipment, thus to define the universal external input and output interface information for subsequent state transition model, function model and function hierarchy model, wherein the interface information includes two parts: the first part is the interface communication information, namely: the communication formats and contents of various common interfaces configured in the external interaction model for the communication between the software and the external interaction equipment, and this part mainly includes interface name, interface type, Baud rate for bus communication, routing and addressing, priority, transmission rate, etc.; the second part is the input and output interface constraint conditions, namely: the logical and temporal constraint conditions of/among the interfaces defined according to the software requirement documents, wherein the logical conditions include AND, OR, NOT, Mutex, etc., and the temporal constraint conditions include preorder, postorder, concurrence, delay, timing, etc., and such constraint condition information is recorded in the model in order to finally obtain the external interaction environment model of the software.

On the basis of the external interaction environment model, the data transmitted in the bus are organized in a form of data frame according to the interface data documents for the software development to establish the bus data transmission model. Specifically, the data in the bus are transmitted in a form of data frame and the same bus usually includes one or more data frames, wherein each data frame includes its own transmission direction, transmission period, data frame length, multiple frame variables each containing variable type, length and other information. Through the above method, the data transmitted in the bus are organized in forms of data frame and frame variable, thus to map the byte-code transmitted in the bus to the data frame.

As the identifiable variables with physical significance in the software, the interface data elements and the internal data elements are the operands of the subsequent function model and state transition model, wherein the variable attributes include the physical significance description information, such as data unit, resolution ratio, error, significant interval, etc. Specifically, the interface data elements used for describing the external interaction behaviors of the software are obtained through the mapping of the bus data transmission model. The example for the implementation process of the variable mapping module is as shown in Figure 2, the first two short type frame variables in the data frame are mapped into Height interface data elements through the

mapping expression: Height=Var1<<8+Var2. In this example, the mapping refers to the value mapping, the mapping relation is described through a mathematical expression. Besides the value mapping, the interface data elements can also obtain such logical and temporal information as data frame period and meanwhile introduce such information into the internal model of the software. Additionally, the internal data elements composed of the intermediate results and the temporary variables generated during the internal software process are obtained from the requirement document.

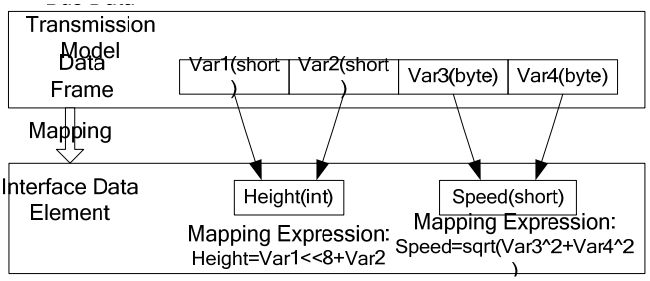

Figure 2 Example for Implementation of Variable Mapping Model

After the interface data elements and the internal data elements are obtained, the state diagram in the standard UML (Unified Modeling Language) can be used to model the software running state & way so as to establish the state transition model. For security analysis, OCL (Object Constraint Language) shall be adopted to formally describe the state transition in the state diagram, and the state transition syntax format in the standard UML is as follows: event name [control condition]/action expression ^ sending clause, and OCL formalexpansion for the state diagram mainly includes the following aspects: 1) Change event: it means that if the variable in a Boolean expression is changed and the value of the expression is correspondingly changed, then some conditions can be met; if there is any change event, then the control condition may be blocked and the state transition is realized after the control condition is met. 2) Control condition: such symbols as AND, OR, NOT, =, < and > can be used in OCL expression to describe the control condition for condition judgment. 3) Actual parameter of action expression: the actions in the state diagram include operation call and sending event, and they usually need to take along the parameters; in order to meet the security analysis requirement, OCL expression shall be used to clearly designate the actual parameter rather than the formal parameter. 4) Target object of the sending clause: the transition action expression in the state diagram points out the action executed by the object itself when the state transition is activated (namely: when the object state is transferred). Sometimes, an object needs to send message to another object in order to get the assistance from that object during the function execution process. At this moment, OCL is adopted to describe the target object sending the message.

After functional decomposition, it is necessary to establish the function and state correlationat the bottom layer of the function hierarchy model. Multiple functions may be concurrently executed in the same state, and the same function may appear in different states to execute different processing logics. As shown in Table 1, all states include fuel oil control (RC) function. Specifically, the processing logic of the fuel oil control function in the initial state is expressed by RC1 which is called operation sequence; the processing logics in cold runningstate, false runningstateand ground start state are the same and are expressed by RC2;After the function and stateare associated with each other, the function can be further decomposed into operation sequence, and the software failure caused by such complex logics as function and state combination is allowed to be inspected.

Table 1 Function and State Correlation Example

	Fuel Oil Control (RC)	Guide Vane Control (DK)	Switch Control (KK)	Signal Self-inspection (XZ)	Circuit Monitoring (HJ)	Surge Elimination (XC)
Initial State	RC1	DK1	KK1	XZ1	HJ1	XC1
Cold Running State	RC2	DK1	KK2	XZ1	HJ1	XC2
False Running State	RC2	DK1	KK3	XZ1	HJ1	XC2
Ground Start State	RC2	DK1	KK4	XZ1	HJ1	XC2

After function and state are associated with each other, it is necessary to establish the model for the operation sequence (such as RC1) of the function. In order to meet the security analysis

requirement, the function model is divided into the following three parts: external input interface (Input) - operation sequence process (Process) - external output interface (Output) (abbreviated as IPO), and the above three requirement elements form the dynamic failure link in the software running state. Specifically, the external input interface and the external output interface of the function are selected from the interface data elements and the internal data elements for association, and after association, the function can obtain input and output objects and meanwhile obtain such information as period and time sequence from the interface data elements and the internal data elements. Next, UML and OCL languages are adopted to establish the model for the operation sequence process in order to formally describe the processing procedures of the software function in special states. Specifically, the processing procedures are expressed by the activity diagram in UML, and OCL is adopted to formally expand the activity diagram: 1) Object instance: OCL expression is adopted to designate the object instance which executes a certain activity, and the use method thereof is similar to that of the target object in the sending clause of the state diagram in the state transition model. 2) Decision-making condition and synchronization condition: the use method thereof is similar to that of the control condition of the state diagram. 3) Actual parameter: the use method of the actual parameter is similar to that of the action expression parameter of the state diagram.

Automatic Analysis of Software Security

How to establish the software security analysis oriented requirement model and how to obtain the software security analysis rules are described in the above paragraph. The implementation process of the automatic analysis of the software security is as follows: the computer program is adopted to compile the corresponding failure detection algorithm for each security analysis rule, wherein the failure detection algorithm aims at traversing the requirement model according to the semantics of the security analysis rule. Specifically, according to the temporal relation, the judgment condition, the transition condition, the significant value interval and other information, the failure detection algorithm statically scans the complex software requirement model in order to find all failure modes designated in the security analysis rules and probably causing system dangers.

Next, this analysis rule “During the whole system running process, multiple functions output for the same variable, thus causing the value conflict” is taken as an example to describe the implementation method of the failure detection algorithm. The failure detection algorithm is as follows: step 1, traverse the state transition model to find the concurrent states in the system which may have several groups of concurrent states, and implement the following algorithm for each group, wherein one group of the concurrent states are as shown in Figure 3and include state 1, state 2 and state 3. Step 2, traverse the operation sequence of the functions running in each state at the same time according to the function and state correlation in the requirement model, wherein function 1 and function 2 run in state 1, function 3 runs in state 2, and function 4 runs in state 3. Step 3, find the input and output interface data elements and the internal data elements of the operation sequence of each function through the function model.Step 4, check whether multiple functions have the same output interface data element, wherein function 1, function 3 and function 4 have the same output interface data element O1, function 3 and function 4 have the same output interface data element O3. Step 5, for the functions with the same output interface data element, check whether the value assignment operations are concurrently executed to the output interface data element. Specifically, in the example of function 1, function 3 and function 4 with the same output interface data element O1: in function 1, the decision-making condition on the path from the starting point to O1:=1 is I2<0; in function 3, the decision-making condition is I2>0 and I2<3; in function 4, the decision-making condition is I4>0. Obviously, I2<0 in function 1 and I2>0 and I2<3 in function 3 cannot be true at the same time. If I2 and I4 are assigned in thesignificant intervals, I2<0 in function 1 and I4>0 in function 4 may be met at the same time,I2>0 and I2<3 in function 3 and I4>0 in function 4 may be met at the same time.According to the above analysis, two failure modes can be obtained: function 1 and function 4 execute the value assignment operation to the interface data element O1 at the same time; function 3 and function 4 execute the value assignment

operation to the interface data element O2 at the same time.

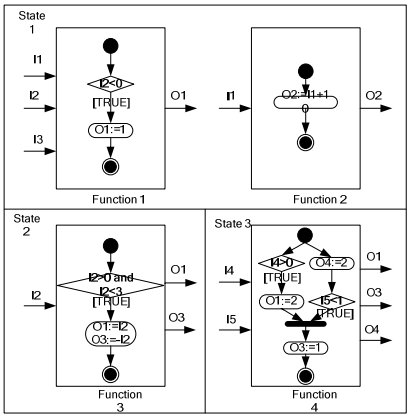

Figure 3 Internal Concurrent State of Software and Function Example

Engineering Practice

In an engineer software security analysis project, the automatic analysis platform researched and developed according to the technique proposed in this article is adopted for the security analysis of the engine software. Meanwhile, the traditional manual analysis method is also compared with the automatic analysis platform in the aspects of workload and analysis effect.

Table 2 Comparison of Time Consumption of Security Analysis

	Automatic Analysis Platform (Hour/Person)	Manual Analysis (Hour/Person)
Modeling Time	348	293
Analysis Time	0.1	265
Total Time	348.1	558

According to Table 2, compared with the manual analysis, the automatic analysis platform needs a longer modeling time due to the formalmodeling. Meanwhile, the automatic analysis platform can achieve 1,162 security analysis rules and complete the security analysis of the whole software within 5min, but the manual analysis takes 265h. In the aspect of the total time, the automatic analysis platform takes less time to complete the security analysis. In the aspect of the analysis result, according to Table 3, compared with the manual analysis, the automatic analysis platform can discover more failures and more verified severe failures.

Table 3 Comparison of Security Analysis Result

	Automatic Analysis Platform (Items)	Manual Analysis (Items)
Input Interface Failures	297	263
Output Interface Failures	106	74
Independent Function Failures	157	138
Combined Function Failures	51	26
State Failures	119	31
Total Discovered Failures	730	532
Verified Severe Failures	69	21

Conclusion and Expectation

An automatic software requirement security analysis technique based on requirement model and security analysis rules is proposed in this article, and this method can achieve the requirement security analysis of the aeronautic embedded software. The software failures in the software requirement security analysis results are constantly increased along with the continuous improvement of the software complexity, and it takes too much time and too much energy to verify whether the software loses efficiency andto reasonably and effectively process the software failures. Therefore, it is significantly necessary to research the automatic verification technique for the software requirement security, and such project has been currently researched in new scientific and industrial subjects.

Acknowledgements

Technology basic research project from National Defense Science and Technology Industry Bureau(Z0520138009)

Reference

[1]Xu XJ, Bao XH, Lu MY, Chang W.A study and application on airborne software safety requirements elicitation[C]. In: Proc. of the2011 9th Int'l Conf. on Reliability, Maintainability and Safety (ICRMS). 2011. 710-716. [doi: 10.1109/ICRMS.2011.5979357]

[2]Walia GS, Carver JC. A systematic literature review to identify and classify software requirements errors[J]. Information andSoftware Technology, 2009,51(7):1087-1109. [doi:10.1016/j.infsof.2009.01.004]

[3] Rausand M, Høyland A. System Reliability Theory: Models, Statistical Methods and Applications[M]. 2nd ed., Wiley, 2004.[doi:10.1002/9780470316900]

[4] IEC 60812: Analysis techniques for system reliability. In: Proc. of the Failure Mode and Effect Analysis (FMEA)[S]. Int'lElectrotechnical Commission, 1991.