ICAS 2021 SHANGHAI
International Council of the Aeronautical Sciences
32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

DESIGN AND RESEARCH ON SAFETY ANALYSIS TOOL FOR AVIONICS SYSTEM

LIU Yu

Shanghai Aircraft Design and Research Institute, Shanghai, China201210

Abstract

With the continuous improvement of avionics system, the traditional method of safety analysis is difficult to guarantee the completeness of failure mode as it is too dependent on engineering experience. And in the process of system iterative design, due to the complexity of the system, the workload of safety analysis is too large, which increases the time and cost. Aiming at the above problems, an automatic safety analysis tool was designed, the safety data model was established based on SysML, automatic fault tree was built by using route tracing method, and the common mode analysis and zone safety analysis were carried out on the generated fault tree. The experimental results of a system show that the tool can realize the automatic modeling and analysis of the fault tree and improve the efficiency and competeness of the safety analysis.

Keywords: avionics system; fault tree; systems modeling language; safety analysis

1. INTRODUCTION

The structure of avionics system has gone through the development process from the initial discrete avionics system, combined avionics system to integrated avionics system[1]. At present, the development of avionics is dominated by the third generation, with fewer and more centralized processing units replacing a large number of independ-ent processors and field replaceable units (LRUs). It plays an important role in reducing weight, saving maintenance costs, reducing resource allocation, improving resource efficiency and reducing crew workload on the new generation of civil aircraft[2]. While integration brings the above benefits, it also increases the complexity of the system, which makes the propagation and uncertainty of system faults in the synthesizing process have a great impact on sys-tem safety. Traditional safety analysis methods (such as FTA, FMEA)[3] mainly depend on engineering experience, and are not synchronized with system design. With the increase of system complexity, it is difficult to list all failure modes and effects of the system. At the same time, it is difficult to ensure that safety requirements can be timely fed back to the system design due to the iteration of system design. The root cause of these problems is that the data of system design and safety analysis can not be ex-pressed uniformly, which makes the system design unable to show its safety attributes, and the results of safety analysis can not be directly fed back to the system design model.

To solve the above problems, we need to adopt a consistent formalized model for system design and safety analysis, and implement automatic safety analysis based on this model. This paper establishes the data model of system architecture based on SysML Description Language[4], and studies the description method of the safety data model attached to the system architecture data model. Then, using the idea of model-driven safety analysis, it studies the automatic fault tree analysis method and the au-tomatic safety isolation requirement checking meth-od. Finally, based on the automatic analysis method, a rapid safety analysis tool is designed. The effec-tiveness of the tool is verified by experiments.

2. Description of Safety Data Model Based on SysML

SysML is an extension of UML in the field of system engineering application. It can model various problems of system engineering and effectively sup-port requirement description, system structure de-sign, function behavior and allocation.

This paper establishes a safety data model by adding safety attributes to the system architecture

data model based on SysML description language. In order to achieve automatic safety analysis, it is necessary to define the data needed for fault tree analysis and safety isolation analysis, as shown in Table 1.

Table 1– Requirements for safety analysis data

Number	Data	Type	Description
1	Id	String	Identification
2	Name	String	Name
3	Premise	EANode	Resident node
4	Parent	List<String>	Source links
5	Child	String	Target Link
6	Availability	Float	Availability
7	Integrity	Float	Integrity
8	Level	String	Module Hardware Hierarchy
9	Area	RectangleF3D	Location and volume information

Among them, architecture-related attributes (such as 1-5 items in Table 1) have been defined in architecture modeling, and safety data (6-9 items in Table 1) have not been independently modeled in architecture modeling, which needs further descrip-tion. In this paper, the tagged value is used to de-scribe the safety data model needed for analysis, and the relationship between the output abnormal state and the input stream is described by assertion. The added tagged value is shown in Table 2.

Table 2– Description of Safety Data Model

Number	Data	Type	Description	SysML Description Method
6	Availability	Float	Availability	Tagged value：LossOfFunction
7	Integrity	Float	Integrity	Tagged value：Malfunction
8	Level	String	Module Hardware Hierarchy	Tagged value：Level
9	Area	RectangleF3D	Location and volume information	Tagged value：StationLine Tagged value：WaterLine Tagged value：ButtockLine

3. Research on Automatic Fault Tree Analysis Method

3.1 Method of Automatic Fault Tree Generation

3.1.1 Principle of Automatic Fault Tree Generation

The principle of automatic fault tree generation is shown in Figure 1. While building SysML system architecture model, the safety data model is estab-lished. Based on the XML files derived from the two models, the fault tree is generated automatically, and the analysis results are selected and displayed to support the fault tree analysis.

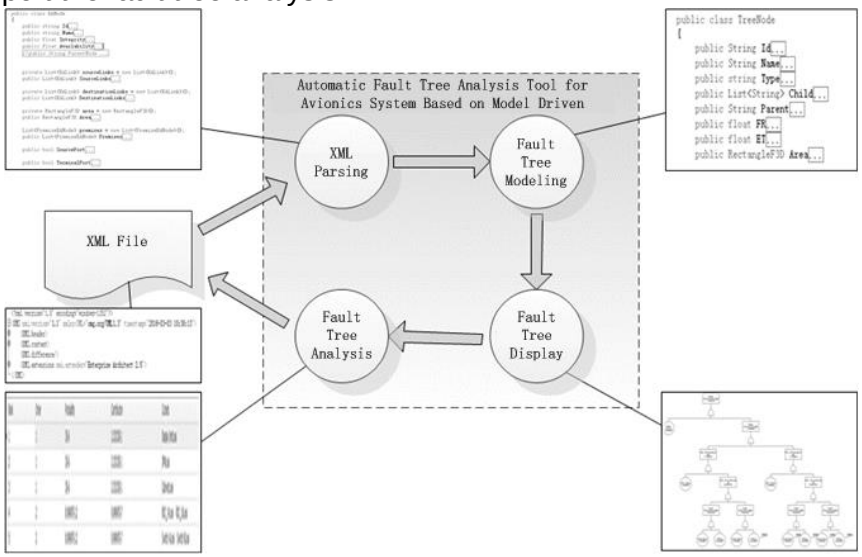

Figure 1 – Principle of Automatic Fault Tree Generation.

When using SysML model data to automatical-ly generate fault tree, the first step is to expand the hierarchical structure to get a planar system architec-ture model; secondly, to find the failure mode of the terminal object, determine it as the top event, and connect it with the failure events of other objects; then, the fault source is searched by path tracing method, and the possible combination of failure causes is obtained according to the data flow rela-tionship with other objects. Based on this, the top event fault tree is automatically established. The process of automatically generating fault tree from the data of avionics system architecture model based on SysML is to find out the cause of failure on de-mand and only care about other failure events relat-ed to top events.

Based on the above basic idea of automatic fault tree generation, the process of constructing fault tree can be understood as the process of trans-forming SysML structure into fault tree structure. In the process of fault tree modeling, we need to inject the safety data model into SysML model firstly. The process of injection is based on these two models to determine the failure mode, failure probability and failure transfer mode of elements. Among them, failure modes are classified into Loss and Error. Fi-nally, fault tree modeling can be completed by sort-ing out the injected results into the form of fault tree.

The method described in this paper uses the fault tree modeling method of path tracing, that is, starting from the fault mode of the terminal object, and inferring the possible causes of the fault accord-ing to the path of data transmission, thus completing the injection process. The basic unit of fault tree modeling can be described as: for a module, its func-tional fault must be caused by its own fault or data flow input fault, and for SysML basic unit as shown in Figure 2, the basic unit of fault tree as shown in Figure 3 is established.

Figure 2 – Basic Unit in SysML Model.

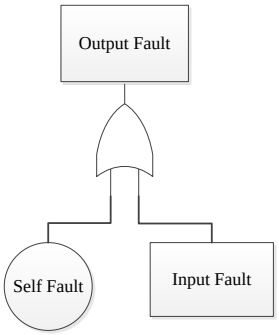

Figure 3 – Basic Unit of Fault Tree.

3.1.2 Algorithm of Automatic Fault Tree Generation

Based on the principle of automatic fault tree generation, the algorithm of fault tree generation is as follows:

Step 1: If the system architecture model is a hi-erarchical model, expand it so that it does not con-tain elements that can be further divided into other elements.

Step 2: Determine the top event of the fault tree: Look at the relevant description in the Package ele-ment and determine the top event according to the failure mode tags LossOfFunction and MalFunction.

Step 3: For LossOfFunction, do the following:

1) For an object A, the basic unit of fault tree is constructed as shown in Figure 4. Its output loss is connected by OR gate. The cause of failure is the loss of A itself, that is, the corresponding loss of A function (if LossOfFunction has been defined), and the input loss of A.

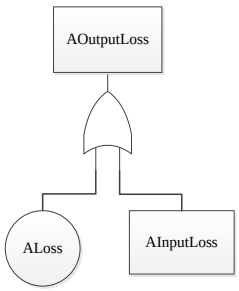

Figure 4 – Loss Type Fault Tree Unit.

2) For the input loss of an object A, if the data stream only comes from object B, the event is connected by or gate, and one of the failure reasons is the loss of B and the other is the loss of B input.

3) For the input loss of an object A, if its data stream originates from multiple objects, look at the data stream relationship between A and other objects, find the expression of the loss pattern of an object pair, and map it to a fault tree. If the corresponding expression can not be found, the default is that these objects are redundant backup relationships, and connect their output loss events with AND gates.

4) For object A without data stream source, the loss of A itself replaces the loss of A's input and output.

Step 4: The operation for MalFuntion is similar to Step 3, except that when the data stream of an element comes from multiple elements, if the corre-sponding Assertion expression can not be found, the output loss events of the element are connected with OR gates, because even if the redundant backup rela-tionship, any input module error will cause the module error.

3.2 Method of Automatic Fault Tree Analysis

This paper completes the establishment of fault tree on the premise of fault independence hypothesis. In order to ensure the establishment of this hypothe-sis, in addition to the basic qualitative and quantita-tive analysis of fault tree, common mode analysis and regional safety analysis are also needed.

3.2.1 Fault Tree Analysis

In this paper, the open source fault tree analysis tool XFTA is used to analyze the fault tree. The gen-erated fault tree file is transformed into an XML file which conforms to OPENPSA standard. Then the XML file is used as the input of XFTA, and the XFTA algorithm is used to calculate the minimum cut-set and probability of each intermediate event. At the same time, the structural importance, proba-bilistic importance and critical importance are ob-tained.

3.2.2 Common Mode Analysis

The purpose of automatic common mode checking is to support common mode analysis, which is further divided into three parts: minimum cut-set, common mode checklist and common mode event check.

1) Minimum Cut-set

Based on the algorithm of fault tree cut-set cal-culation, the cut-set of an intermediate event can be calculated, so the minimum cut-set and the first-order minimum cut-set can be displayed through the visual interface by selecting an intermediate event in the fault tree.

2) Common Mode Checklist

Based on the classification of common mode sources in ARP 4761, the common mode analysis checklist and record table are displayed through vis-ual interface to check common mode sources and errors of two hardware events in order to support automatic common mode analysis.

3) Common Mode Event Check

Based on the result of fault tree generation, we can give a same hardware bottom event which appears repeatedly in different branches of AND gate event, and display it through visual interface, and realize automatic common mode analysis by com-bining minimum cut-set and common mode checklist.

3.2.3 Zone Safety Analysis

The purpose of automatic safety isolation in-spection is to support zone safety analysis, which includes intersection check, hazardous area check and separation distance check.

1) Intersection Check

Intersection check is used to determine whether there are overlapping areas between two basic hard-ware events in the fault tree. Based on the result of fault tree generation, the basic events of hardware type are identified. Combining the location and vol-ume information in the safety data model, the three-dimensional model of two basic events is generated. Through the spatial location of the two three-dimensional models, the overlap area of the two basic events of hardware type is judged.

2) Hazardous Area Check

Hazardous area check is used to determine whether two hardware type cut-set events under the same AND gate fall into the same dangerous area in the fault tree. Based on the result of fault tree gen-eration, two hardware-type cut-set events under the same AND gate are identified. Combining with the location and volume information in the safety data model, the three-dimensional model of two basic events is generated. Then, the three-dimensional model of hazardous areas is generated based on the input of hazardous area information (location and edge length) manually. Through the spatial location of two cut-set event stereo models for the spatial location of the dangerous area stereo model, we can judge whether the two hardware type cut-set events fall into the same hazardous area defined.

3) Separation Distance Check

Separation distance check is used to determine whether two hardware type cut-set events under the same AND gate in the fault tree satisfy the distance requirement. Based on the result of fault tree genera-tion, two hardware type cut-set events under the same AND gate are identified. Combining with the location and volume information in the safety data model, the three-dimensional model of two basic events is generated, and the distance between the spatial positions of the two cut-set event stereo mod-els is calculated. Then, based on the artificial input distance, a three-dimensional model of two cut-set events is constructed. Then, based on the manual input distance requirement information, it judges whether the two hardware type cut-set events meet the specified distance requirement.

4. Design of Automatic Safety Analysis Tool

4.1 Software Architecture of Automatic Safety Analysis Tool

The architecture of automatic safety analysis tool is shown in Figure 5. Based on the XML file of EA architecture model, automatic safety analysis is realized.

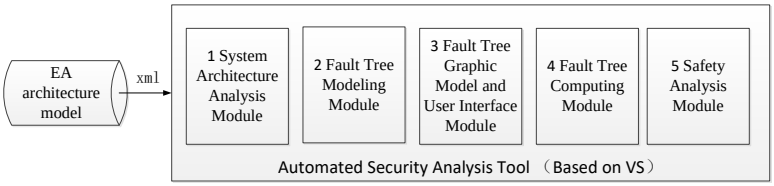

Figure 5 – Automatic Security Analysis Tool Architecture.

4.2 System Architecture Analysis Module

The function of the system architecture analysis module is to extract information from the XML for-mat file of EA architecture model and transform it into "EANode" class as shown in Table 3. The fault tree modeling module directly uses it to build the fault tree. This module uses XML programming in-terface to analyze the XML file of EA architecture model.

Table 3– EANode Attribute Description

EANode Attribute	Type	Description	XML-related Information
Id	String	Unique identifier	
Name	String	Name	

Availability	Float	Probability calculation for function loss	Tagged value: LossOfFunction
Integrity	Float	Probability calculation for function error	Tagged value: Malfunction
Level	String	Unit level	Tagged value: Level
Premise	EANode	Resident node	Architecture diagram attribute
SourceLinks	List<EALink>	Linking Data Stream Input List	Architecture diagram attribute
DestinationLinks	List<EALink>	Linking Data Stream Output List	Architecture diagram attribute
SourcePort	Bool	Source node	Architecture diagram attribute
TerminalPort	Bool	Target node	Architecture diagram attribute
Area	RectangleF3D	Location and volume information	Tagged value: StationLine, WaterLine, ButtockLine

4.3 Fault Tree Modeling Module

The function of the fault tree modeling module is to transform the "EANode" class attributes extracted from the system architecture analysis module into the "TreeNode" class attributes of the fault tree structure. The definition of the "TreeNode" class attributes is shown in Table 4. In the process of transformation, the structure data of fault tree are generated by analyzing the correlation among the objects in the "EANode" class attributes, and the information needed for probability calculation is extracted at the same time.

Table 4– Description of TreeNode attributes

TreeNode attribute	Type	Description
Id	string	Unique identifier
Name	string	Name of Fault Tree Node (Gate/Event)
Type	string	Types of fault tree nodes (Gate/Event), such as intermediate events, basic events, and gates, or gates
Child	List<string>	Child node Id List
Parent	string	Parent node Id list
FR	float	failure rate
ET	float	Exposure time
Area	RectangleF3D	Location and volume information
Level	string	Hardware level, such as LRU or LRM

4.4 Fault Tree Graphic Model and User Interface Mod-ule

Fault tree graphics model and user interface module mainly realize the following two functions:

1) Graphic representation of fault tree: Accord-ing to the fault tree modeling module, the fault tree is represented and displayed by standard fault tree graphic symbols.

2) User interface: mainly includes three parts: a) users can modify and edit the fault tree manually; b) users can select and display the results of fault tree calculation by clicking on graphics and labels; c) users can get the results of safety requirements checking by inputting safety requirements.

The graphical representation of fault tree is re-alized by associating the attributes of "TreeNode" with the "Northwoods. go", "Northwoods. Xml" and "Northwoods. go. Layout" libraries.

The output function of fault tree calculation re-sults is realized by reading the result file and dis-played directly in the fault tree graphical interface.

4.5 Fault Tree Computing Module

Fault tree calculation module is mainly based on XFTA open source engine to calculate the probability of fault tree. The probability of top event of fault tree is calculated by calling XFTA open source fault tree analysis software. XFTA is a fault tree evaluation engine, which includes efficient algorithms for calculating probability, cut set and so on. Among them, the "depth first call" function is used to obtain the probabilistic values of the intermediate events and the minimum cut set of the fault tree.

4.6 Safety Analysis Module

According to the results of fault tree calculation module, the safety analysis module checks common mode and safety isolation requirements, and displays them in the form of lists.

1) ErrorList：The errors in the structure of the fault tree model are listed, such as that OR gate must have two or more child nodes;

2) IntersecCheck：The interaction between two basic event entities in the fault tree is listed;

3) MiniCutSet：List all minimum cut-sets of the event based on the selected top or intermediate event;

4) 1-MiniCutSet：List all first-order minimal cut-set of the event based on the selected top event or intermediate event;

5) CMAChecklist：List cut-set events under the same AND gate, record the common-mode checking results and requirements between two events by checklist, and inspect them from eight aspects: conception and design, manufacturing, in-stallation/integration and experiment, operation, maintenance, testing, calibration and environment. Each checking result includes four aspects: N/A (not applicable), to be analyzed, derivative mitigation requirement, and independence self-evident;

6) CMEventList：List the basic events that re-cur in different branches below the same AND door;

7) HazardAreaCheck：Defining hazardous are-as under the "Zones" tag in the attribute area, includ-ing central coordinates, length, width and height, and recalculating, lists whether the cut-set event en-tities in the same AND gate fall into the defined area;

8) DistanceHazard：List the hazardous area distances defined by a hierarchical entity distance of the cut set event in the same AND gate;

9) SeparationCheck：List the distances be-tween a hierarchical entity of cut set events in the same AND gate.

5. Case Analysis

In order to verify the effectiveness of the safety analysis tools designed, the above tools are used to automatically model and analyze the fault tree for the system EA model shown in Figure 6.

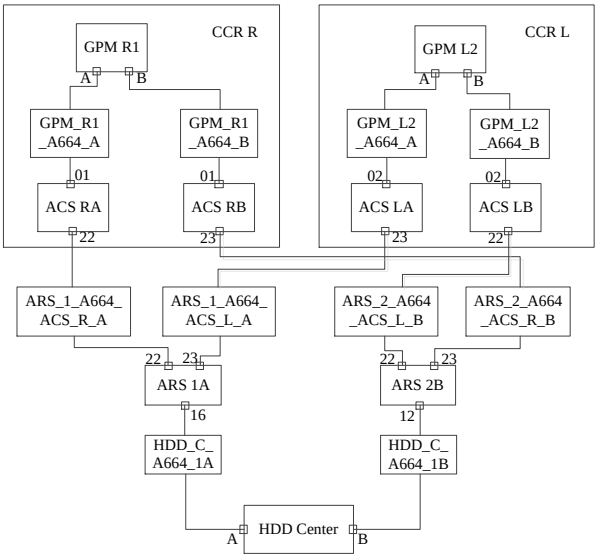

Figure 6 – EA Model.

5.1 Safety Analysis Module

Generate the XML file of the EA model in the automatic safety analysis tool and import it. After background calculation, the safety analysis tool converts the XML file into fault tree directly, as shown in Figure 7.

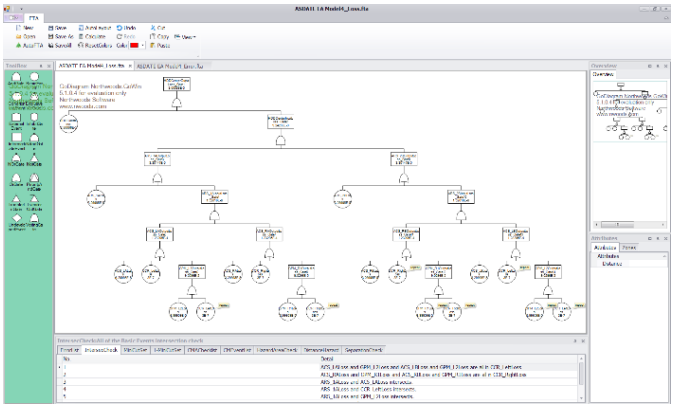

Figure 7 – Automatically Generated Fault Tree.

5.2 Results of fault tree analysis

For each model, the tool generates a "Loss" type and a "Error" type failure state fault tree, which are displayed in two labels, and the whole fault tree is displayed in the "profile area" on the right side, as shown in Figure 7. In the displayed fault tree, failure probability is also displayed under the names of top events, intermediate events and basic events.

The checking results of common mode and safety isolation requirements are as follows:

1) Result of ErrorList

There is no model error in this example, so the ErrorList is empty, as shown in Figure 8.

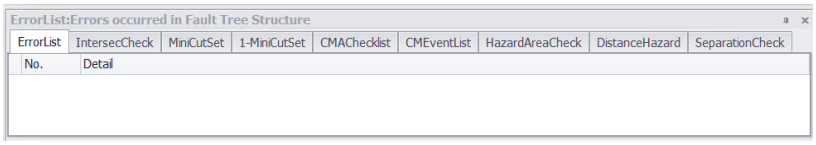

Figure 8 – Result of ErrorList.

2) Result of IntersecCheck

In this example, six intersection cases are checked out in the "Loss" type fault tree, as shown in Figure 9, including two types of inclusion (the first two) and intersection (the last four).

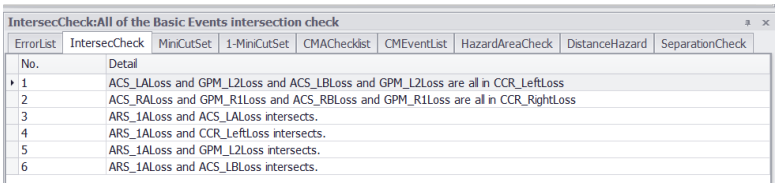

Figure 9 – Result of IntersecCheck.

3) Result of MiniCutset

Selecting the top event of the "Loss" type fault tree, the results of 21 minimum cut sets are checked. As shown in Figure 10, the results include one first-order cut set, five second-order cut sets, 14 third-order cut sets and one fourth-order cut set. The fail-ure probability of the cut set and its contribution to the failure probability of the top event are given.

MiniCutSet:Minimum Cut Sets list, shown after click the GateEvent

ErrorList | IntersecCheck | MiniCutSet | 1-MiniCutSet | CMAChecklist | CMEventList | HazardAreaCheck | DistanceHazard | SeparationCheck

Rank	Order	Probability	Contribution	Cutsets
1	1	2.99996E-5	9.99692E-1	HDDCenterLoss
2	2	8.09927E-9	2.69897E-4	GPM_L2Loss GPM_R1Loss
3	2	1.08896E-9	3.62882E-5	ARS_1ALoss ARS_2BLoss
4	2	2.69988E-11	8.99696E-7	CCR_RightLoss GPM_L2Loss
5	2	2.69988E-11	8.99696E-7	CCR_LeftLoss GPM_R1Loss
6	3	9.80024E-14	3.26579E-9	ACS_RBLoss ARS_1ALoss GPM_L2Loss
7	3	9.80024E-14	3.26579E-9	ACS_LALoss ARS_2BLoss GPM_R1Loss
8	3	9.80024E-14	3.26579E-9	ACS_RALoss ACS_RBLoss GPM_L2Loss
9	3	9.80024E-14	3.26579E-9	ACS_LBLoss ARS_1ALoss GPM_R1Loss
10	3	9.80024E-14	3.26579E-9	ACS_RALoss ARS_2BLoss GPM_L2Loss
11	3	9.80024E-14	3.26579E-9	ACS_LALoss ACS_LBLoss GPM_R1Loss
12	2	9E-14	2.99912E-9	CCR_LeftLoss CCR_RightLoss
13	3	3.59352E-14	1.19749E-9	ACS_LBLoss ACS_RBLoss ARS_1ALoss
14	3	3.59352E-14	1.19749E-9	ACS_LALoss ACS_RALoss ARS_2BLoss
15	3	3.26689E-16	1.08864E-11	ACS_RBLoss ARS_1ALoss CCR_LeftLoss
16	3	3.26689E-16	1.08864E-11	ACS_LALoss ARS_2BLoss CCR_RightLoss
17	3	3.26689E-16	1.08864E-11	ACS_RALoss ACS_RBLoss CCR_LeftLoss
18	3	3.26689E-16	1.08864E-11	ACS_LBLoss ARS_1ALoss CCR_RightLoss
19	3	3.26689E-16	1.08864E-11	ACS_RALoss ARS_2BLoss CCR_LeftLoss
20	3	3.26689E-16	1.08864E-11	ACS_LALoss ACS_LBLoss CCR_RightLoss
21	4	1.18584E-18	3.95165E-14	ACS_LALoss ACS_LBLoss ACS_RALoss ACS_RBLoss

Figure 10 – Result of MiniCutset.

4) Result of 1-MiniCutset

According to the result of MiniCutset, the top event of the "Loss" type fault tree has only one cut set, as shown in Figure 11.

1-MiniCutSet:First Order Cut Sets list, shown multiple times in different branches

ErrorList | IntersecCheck | MiniCutSet | 1-MiniCutSet | CMAChecklist | CMEventList | HazardAreaCheck | DistanceHazard | SeparationCheck

Rank	Order	Probability	Contribution	Cutsets
1	1	2.99996E-5	9.99692E-1	HDDCenterLoss

Figure 11 – Result of First-order Minimum Cut Set.

5) Result of CMAChecklist

According to the results of MiniCutset, cut-set events of the same AND gate are listed. As shown in Figure 12, common mode checks and records are performed for each two events to assess whether de-rivative requirements are needed.

CMAChecklist:Common Mode Analysis Checklist and record table

ErrorList | IntersecCheck | MiniCutSet | 1-MiniCutSet | CMAChecklist | CMEventList | HazardAreaCheck | DistanceHazard | SeparationCheck

No.	Item1	Item2	And Gate	Conception and Des...	Manufacturing	Installation/Integration and...	Operation	Maintenance	Test	Calibration	Environment
9	ACS_LALoss	GPM_R1Loss	And1,And2	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
8	ACS_LALoss	ARS_2BLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
7	ARS_1ALoss	GPM_L2Loss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
6	ACS_RBLoss	GPM_L2Loss	And1,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
5	ACS_RBLoss	ARS_1ALoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
4	CCR_LeftLoss	GPM_R1Loss	And1,And2,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
31	ACS_LBLoss	ACS_RALoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
30	ACS_LALoss	ACS_RBLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
3	CCR_RightLoss	GPM_L2Loss	And1,And2,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
29	ARS_2BLoss	CCR_LeftLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
28	ARS_1ALoss	CCR_RightLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
27	ACS_LBLoss	CCR_RightLoss	And1,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
26	ACS_RALoss	CCR_LeftLoss	And1,And2	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
25	ARS_2BLoss	CCR_RightLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
24	ACS_LALoss	CCR_RightLoss	And1,And2	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
23	ARS_1ALoss	CCR_LeftLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
22	ACS_RBLoss	CCR_LeftLoss	And1,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
21	ACS_LALoss	ACS_RALoss	And1,And2	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
20	ACS_LBLoss	ACS_RBLoss	And1,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
2	ARS_1ALoss	ARS_2BLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
19	CCR_LeftLoss	CCR_RightLoss	And1,And2,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
18	ACS_LALoss	ACS_LBLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
17	ARS_2BLoss	GPM_L2Loss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
16	ACS_RALoss	ARS_2BLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
15	ARS_1ALoss	GPM_R1Loss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
14	ACS_LBLoss	GPM_R1Loss	And1,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
13	ACS_LBLoss	ARS_1ALoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
12	ACS_RALoss	GPM_L2Loss	And1,And2	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
11	ACS_RALoss	ACS_RBLoss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
10	ARS_2BLoss	GPM_R1Loss	And1	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A
1	GPM_L2Loss	GPM_R1Loss	And1,And2,And3	To Be Analyzed	Mitigation Requiremen...	Independence Self-evident	N/A	N/A	N/A	N/A	N/A

Figure 12 – Result of CMAChecklist.

6) Result of CMEventList

In this example, there is no recurrence of events in different branches of the same AND gate, so the CMEventList is empty, as shown in Figure 13.

CMEventList:Event shown multiple times in different branches of AND Gate

ErrorList | IntersecCheck | MiniCutSet | 1-MiniCutSet | CMAChecklist | CMEventList | HazardAreaCheck | DistanceHazard | SeparationCheck

No.	Common Event	And Gate

Figure 13 – Result of CMEventList.

7) Result of HazardAreaCheck

Before checking the hazardous area, we need to define a hazardous area. In this example, we define a zone as shown in Figure 14, the center position is (0, 0, 0), the length, width and height are 1111. When the hazardous area is defined and recalculated, it will be displayed under the label "Hazard Area Check". As shown in Figure 15, there are 38 results. The "Cutsets" column lists all cut sets in the same haz-ardous area and in the same AND gate. The "Detail" column shows which hazardous area these cut sets are in together. The "LRUs" column summarizes which LRUs these cut sets belong to.

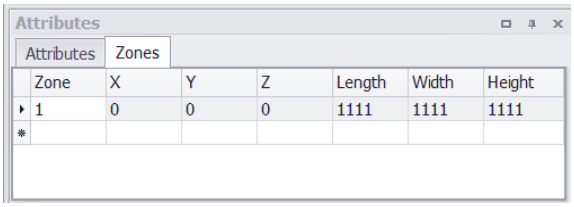

Figure 14 – Definition of Hazardous Areas.

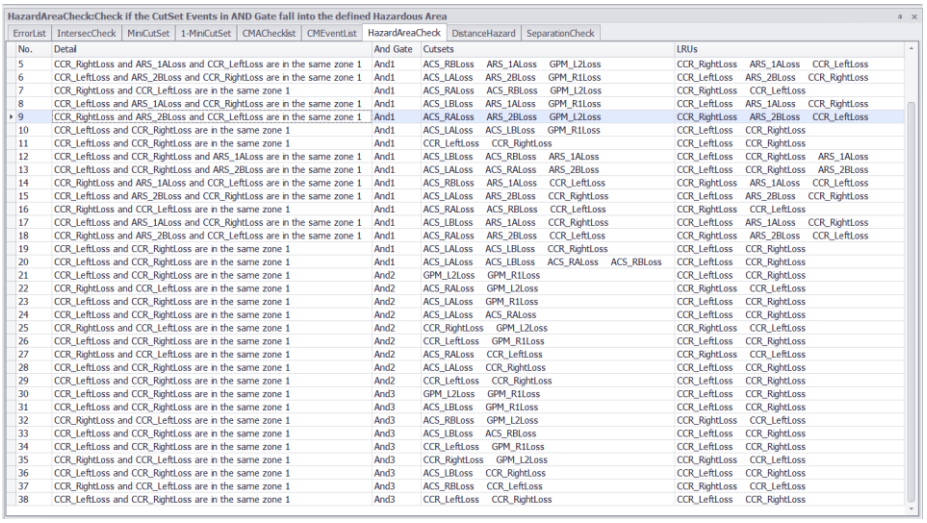

Figure 15 – Result of HazardAreaCheck.

8) Result of DistanceHazard

Based on the HazardAreaCheck results, all LRU events in the same hazardous area are listed and their distances from the defined hazardous area are given, as shown in Figure 16.

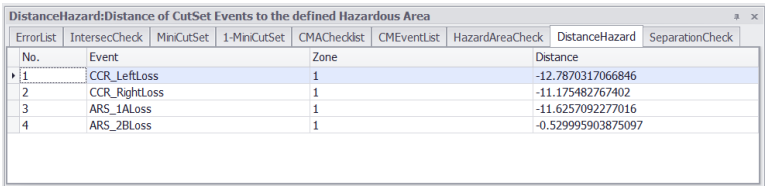

Figure 16 – Result of DistanceHazard.

9) Result of SeparationCheck

According to the results of MiniCutset, the LRU level cut set events under the same AND gate are listed, and the distance between each two events is calculated. The checking results of whether the rectangular areas of each LRU are intersected are given, as shown in Figure 17.

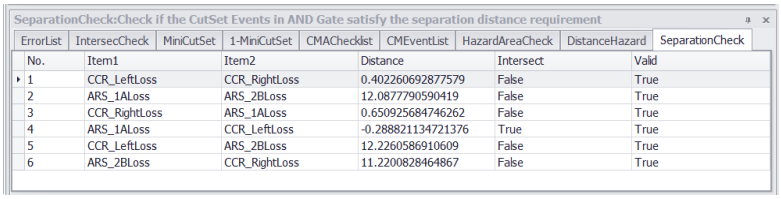

Figure 17 – Result of SeparationCheck.

6. Conclusion

1) The automatic fault tree modeling method designed in this paper is more efficient than the traditional manual tree building method.

2) Compared with traditional methods, the au-tomatic safety analysis method designed in this paper ensures the synchronization of safety analysis and system design.

Therefore, the method adopted by the tool is superior to the traditional manual tree-building method in terms of time and effectiveness.

References

[1] Vesely W E, Goldberg F F, Roberts N H, et al. Fault tree handbook[R]. Washington DC: Nuclear

Regulatory Commis-sion, 1981.
[2] Wang G. Integration technology for avionics sys-tem[C]//2012 IEEE/AIAA 31st Digital Avionics Systems Conference. IEEE, 2012:7C6-1-7C6-9
[3] Li Xiangming, Wu Xuejun. A software design method based on fault-tree analysis[J]. Ordnance Industry Automation, 2011, 30(8): 85-91.
[4] Kyle Hampson. Technical Evaluation of the Systems Model-ing Language (SysML)[J]. Procedia Computer Science, 2015, 44(1): 403-412.

Copyright Statement