applied sciences

MDPI

*Article*

# An Improved FFIP Method Based on Mathematical Logic and SysML

**Jian Jiao *, Shujie Pang, Jiayun Chu, Yongfeng Jing and Tingdi Zhao**

School of Reliability and Systems Engineering, Beihang University, No. 37 Xueyuan Road, Beijing 100191, China; Shujiepang@buaa.edu.cn (S.P.); bhcjy@buaa.edu.cn (J.C.); jyfjiayou@buaa.edu.cn (Y.J.); ztd@buaa.edu.cn (T.Z.)
* Correspondence: jiaojian@buaa.edu.cn; Tel.: +86-186-0051-3132

**Abstract:** In recent years, the model-based safety analysis (MBSA) has been developing continuously. The Functional Failure Identification and Propagation (FFIP) method is a graphics processing technology which supports the analysis of fault propagation paths before making costly design commitments. However, the traditional FFIP has some deficiencies. In this paper, we extend the functional failure logic (FFL) in the FFIP and introduce the concept of deviation. So, FFIP can be used to analyze the failure process of the systems and make the logical analysis of functional failure easier. Based on the extended FFL, we present a new overview of the FFIP. The FFIP is improved by using mathematical logic and Systems Modeling Language (SysML). The standard expression of FFL is realized, which is conducive to the subsequent modeling and modification. Additionally, we use the failure logic analysis in the FFIP to improve the state machine diagram (SMD) in SysML. Finally, the improved FFIP method is used to analyze the fault propagation paths of the system and Simulink is used for simulation. The fault tree is generated according to the simulation results, the minimum cut set is calculated, and the key failure parts of the system are obtained.

**Keywords:** FFIP method; failure logic; mathematical logic; SysML; Simulink

check for updates

**Citation:** Jiao, J.; Pang, S.; Chu, J.; Jing, Y.; Zhao, T. An Improved FFIP Method Based on Mathematical Logic and SysML. *Appl. Sci.* **2021**, *11*, 3534. https://doi.org/10.3390/app11083534

Academic Editor: Wen-Hsiang Hsieh

Received: 14 March 2021
Accepted: 2 April 2021
Published: 15 April 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

With the development of new technology, the demands for system safety, especially in complex industrial systems, are constantly increasing. Currently, there are several safety analysis methods that have been proposed, such as Fault Tree Analysis, Event Tree Analysis, Failure Mode Effect Analysis, Hazard and Operational Analysis, etc. [1]. However, the above methods rely too much on the analyzers' experience and subjective judgment. With increasing complexity of industrial systems, it is difficult to use these traditional safety analysis methods to achieve accurate and complete analysis results.

In the past few years, some new safety analysis methods have emerged, for example, the Functional Resonance Analysis Method (FRAM) and the Systems-Theoretic Process Analysis (STPA) method. FRAM [2] is a systematic analysis method, which can be used to model complex social-technology system. Although it is powerful for visualizing and understanding the functions of system, it does not do well in quantization. It is also difficult to obtain safety constraints. STPA [3] is based on the system theory and cybernetics, it treats the research object as a hierarchical control structure. STPA carries out safety analysis by determining analysis objectives, constructing hierarchical control structure, identifying unsafe control actions, and identifying loss scenarios. However, all these analyses methods need to rely on the subjective experience of personnel, which leads to the deviation of the results of different personnel analysis.

With the advancement of the whole life cycle of the system, relevant safety analysis is constantly updated, and the MBSA was gradually developed [4]. The purpose of MBSA is to study the modeling of complex industrial systems, implement automated or semi-automated safety analysis and perform verification based on system models [5]. MBSA is executed as follows:

1. Establish a general and unified model according to the purpose of system design and analysis.
2. Use formal tools to describe the model.
3. Use simulation or model checking to evaluate the safety of the system based on the model.
4. Generate the analysis results, such as the FTA minimum cut set, FMEA form (Failure Mode Effect Analysis), etc.

At present, some typical MBSA methods include Failure Propagation and Transformation Notation (FPTN), Hierarchical Performed Hazard Origin and Propagation Studies (HIP-HOPS), Failure Propagation and Transformation Calculus (FPTC), AltaRica 3.0, etc. FPTN [6,7] is aimed at software safety analysis. FPTN limits artificial logic reasoning to the module level, reduces the capability requirements for analyzer and increases the accuracy of reasoning. However, this method is not supported by other tools and its scope of application is limited. FPTC [8] is a further development based on FPTN, in which system model is linked with failure model and requires a repeated analysis after injection failure. HIP-HOPS [9,10] is a hierarchical and modular analysis method, it can be used for system reliability modeling and analysis. In this method, the input failure and output failure are linked, and the SAM tool is used to generate the fault tree and calculate the minimum cut to find all propagation paths that cause the output faults. AltaRica [11,12] has been one of the most popular modeling languages in the MBSA field, it is advanced description language based on pattern automata. The modeling languages of MBSA also include Systems Modeling Language (SysML), Architecture Analysis & Design Language (AADL), east-adl2, Security Assertion Markup Language (SAML), etc. [13]. When it comes to model checking, MBSA also has a lot of tools. The most representative of them are Symbolic Model Verifier (SMV) and Simple Promela Interpreter (SPIN) [14–16].

In this paper, we study the Functional Failure Identification and Propagation (FFIP), a graphical evaluation method, which is composed of structural models, functional models, behavior rules, functional failure logic (FFL) analysis and failure simulation. By using this method, a designer can analyze the system function effectively, realize the path of failure propagation, and reduce the influence of human subjectivity through simulation [17]. Simulation and reasoning methods in FFIP are derived from qualitative physics and qualitative reasoning. In other words, the finite state of system behavior is represented, and then the fault propagation path is deduced based on the qualitative relationship between function and behavior [18]. In recent years, FFIP has been constantly expanding and improving. General speaking, FFIP has the following advantages:

1. It does not require the users to make the hypothesis of causality in advance [19], and it avoids the subjective influence of human.
2. It can identify the functional failures caused by global interaction, and simulate any number of failures.
3. It can help engineers identify the key components of the system and quantify the related risks of specific components.

Since FFIP was proposed [17], it has been applied in many fields and developed continuously. In reference [20], a method of automatically generating an event tree based on the FFIP framework was formed, and an automatic tool was developed. In reference [21], the FFIP method is extended to generate training and test data sets for the development of failure detection systems based on a data-driven machine learning method. Reference [22] shows that the FFIP method can effectively evaluate software and hardware. Additionally, in reference [23], human factors are considered, and the impact of human error on system failure is analyzed by using FFIP. In reference [24], external events are considered, the Time-Based Failure Flow Evaluator (TBFFE) is developed based on FFIP method. In reference [25], the Function Failure Propagation Potential Method (FFPPM) is proposed by extending FFIP with graph theory and matrix theory.

Of course, the FFIP still has its own shortcomings. First, the FFIP method is suitable for the early design stage of the system, but MBSA pursues a unified model in the whole

life cycle of the system, from design to use to inspection. Secondly, the FFL of FFIP is not specific and clear for the description and lacks the standardized description. On top of that, the FFIP method uses graphics to describe the model, missing a standardized and unified modeling language.

SysML is a multipurpose standard modeling language, and can support the detailed description, analysis, design, validation, and verification of various complex industrial systems, such as hardware, software, information, process, personnel and facilities and other systems. A SysML diagram contains many elements that can be combined with each other, which can represent the structure, function and behavior of the system in a standardized way [26–28].

In recent years, researchers have combined some parts of FFIP with SysML. In reference [29], the FFL of FFIP is integrated into the Block Definition Diagram (BDD) to evaluate and compare the input and output streams. In reference [30], StateCharts are used to further expand the FFL. The framework of FFIP is expanded by clearly listing the hierarchical relationship among system state, function state, component state, command signal and sensor signal. In the above literatures, FFIP has been combined with SysML, but they only used SysML to expand FFL, and did not combine with the whole modeling process of FFIP.

In this paper, we introduce mathematical logic and SysML to extend the FFIP method from two aspects: the applicable stage of the method and the formation of standardized expression. Firstly, we introduce the concept of failure deviation and extend the FFL. Then, we use mathematical logic to normalize the failure logic. In addition, we combined FFIP with SysML. On one hand, we combine SysML with the whole modeling process of FFIP and propose an integral modeling method under the MBSA framework. It includes building the structure model, function model and behavior model in FFIP with SysML. This solves the problem of lacking standardized language description of FFIP, and makes it easier for the modification and analysis. On the other hand, we improve the state machine diagrams (SMD) in SysML to better represent the FFL of the system. In the end, the S18 brake system in SAE ARP4761 is taken as an example to demonstrate the feasibility of the extended FFIP method, in which we use Simulink to simulate failure and generate a fault tree based on the output of simulation. By calculating the minimum cut set, the key failure location is obtained, and a quantitative probability-based analysis of the FFIP method can be achieved.

## 2. Functional Failure Logic and Propagation Model

### *2.1. Functional Architecture of the System*

The MBSA emphasizes the model as the basis and core of safety analysis [31]. Building the unified model is one of the major differences between MBSA and traditional safety analysis. Based on the model, it carries out more efficient safety analysis than the traditional methods. Generally, designers tend to start with the functional requirements of the system and build the model from top to bottom. On the other hand, analysts prefer from bottom to top based on the functional state of the system. However, in order to build a unique system model, we need to unify the modeling process and standards [32,33]. Therefore, a modeling framework based on function, behavior and structure is formed, as shown in Figure 1.

In the function–behavior–structure modeling framework, defining the overall function is the beginning of system design. The overall function describes the tasks that the system needs to complete. Based on the understanding of the overall function, the framework breaks it down to different levels of functions, builds the relationship between the functions, and constitutes the functional model. As the realization of function depends on the implementation of behavior, we need to extend the function model to behavior and the behavior relationship, and then constitute the behavior model. Behavior needs specific structure to perform, so we need to obtain structure and structural relationship in the components of the system, using them to constitute the structural model. In a system

design, function is the king, while behavior is the foundation of function realization. Structure is the basis of building the whole system. It is the foundation of the existence of the system.

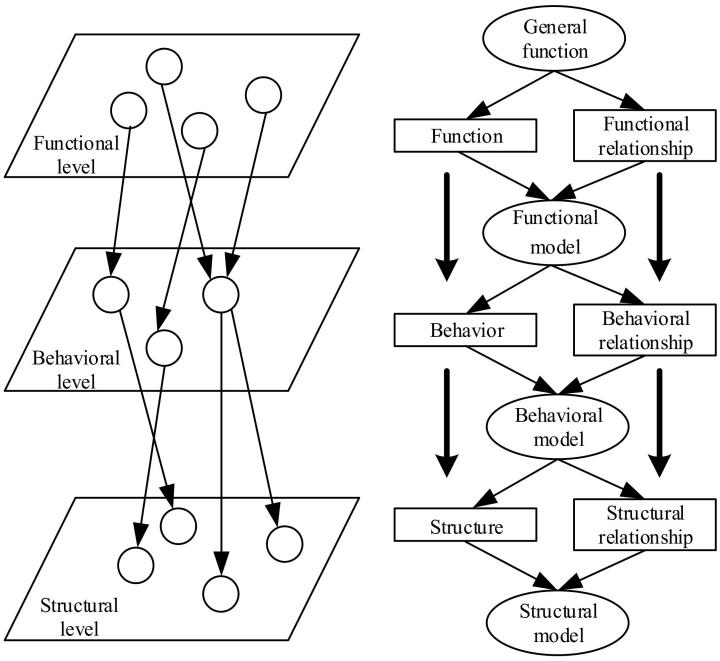



**Figure 1.** System design based on function–behavior–structure model [34].

FFIP is a modeling method based on the mapping of function, behavior and structure [17]. FFIP is composed of three parts. The first part is the structure and function modeling: to draw the structure flow chart and function model diagram according to the system schematic diagram. The second part is behavior modeling, including setting behavior rules and analyzing the FFL. The third part is functional failure propagation simulation, to analyze the path of functional failure propagation. The FFIP is based on the function–behavior–structure relationship mapping, which realizes the systematic safety analysis of failure simulation and qualitative reasoning. In specific failure scenarios, FFIP can simulate the potential function failure mode and determine the fault propagation paths using a qualitative method, to evaluate the impact of the functional failure.

The purpose of FFIP is to find the path of failure propagation. Therefore, FFL is the core of FFIP. This paper focuses on the component failures, which is manifested by the deviation of the key process variables. These variables include flow, pressure, signal, amplitude, etc. In the process of system operation, human factors and external events could also cause system fault, these are all valuable things we need to spend some time on. The common result of these unexpected events is the changes of the key process variables of the components. However, the changes of these variables do not always cause system failure. In other words, component failures do not necessarily cause system fault. Therefore, it is meaningful to analyze the component failures and carry out failure simulation.

### *2.2. The Extension of Functional Failure Logic*

In the modeling process, we need to identify the functions of products and components first, and then complete the system design according to the mapping relationship of function–behavior–structure. However, the function and structure of components are not always a correspondence of one-to-one. As shown in Figure 1, one function might correspond to multiple structures, and one structure might participate in the implementation of multiple functions. Under the design idea of V-model [35], in the design phase, the personnel need to allocate the equipment according to the function corresponding to the

structure. In the verification phase, the personnel need to verify and integrate equipment for the target function. Therefore, the functional requirements of the single component should be considered in the early design stage of the system. One of the purposes of MBSA is to use a unified modeling method in the whole life cycle of equipment [36]. It is incomplete to consider only FFL in the safety system modeling.

Failure logic describes the relationship among the causes, processes, and consequences of failure. FFL is the functional highlights of failure logic. To improve the deficiency that the FFIP method has, namely, that it is mainly suitable for the early design stage of the system, this paper extends the FFL in FFIP and introduces the concept of deviation. The deviation is the degree to which the real situation deviates from the normal situation. Based on the magnitude of the deviation, the state of the components can be classified to infer the extent of the component exception. We can also classify failure deviations and determine failure types based on observed values. In this way, the FFIP method not only stays on the function analysis, but also extends to the specific failure. By analyzing the specific failure of the system, the FFIP method can be applied to the design stage and other stages of the system operation, including the decision-making and formation stage, and the use and maintenance stage [37].

The extension of the concept will lead to a change of the FFIP framework. Firstly, the extended FFIP method needs more detailed modeling of the behavior and function of the system to cover the whole life cycle of the system. It is because the typical FFIP method is suitable mainly for the early design stage, and the function description of the system and components is unspecific. Secondly, the extended FFIP method is changed from analyzing FFL to analyzing failure logic, which makes the level of analysis higher. Additionally, it is necessary to classify failure logic to facilitate subsequent failure logic analysis.

## 3. Failure Logic Based on Mathematical Logic

### *3.1. Classification of Failure*

Researchers always classify the failure modes of a particular system. For example, in reference [38], the faults of an automobile are divided into high oil temperature, abnormal fan speed, abnormal oil pressure, etc. In reference [39], the faults of the hydraulic cylinder are divided into leakage of the telescopic cylinder oil pipe, bending of the oil pipe after extending out of the hydraulic cylinder, and corrosion of the guide sleeve. The purpose of classifying these faults is to identify the fault category more accurately in the process of safety analysis for specific equipment, to determine better improvements. However, there is a lack of a universal classification method without considering the research object. References [40,41] creates a hierarchical fault mechanism taxonomy, using level 1, level 2 and level 3 terminology to describe the potential fault mechanisms. However, this classification is still for those products that have already failed.

In paper [42], the classification of HAZOP deviation is referred, and the failure deviation is divided into NONE, MORE, LESS, and REVERSE. In this paper, through the study of fault classification of different equipment, the failure deviation can be analyzed more comprehensively. Deviation is divided into three categories: Logic, Time, and Value. Each category has its own secondary classification, as shown in Table 1.

The advantage of the above classification is that the type of deviation only needs to be judged according to the observed value directly, not the failure mode of the system or component. The classification of failure deviation can help us to understand the failure cause of a system or component systematically. Then, we can define the transition conditions of the component state and analyze the failure logic. The classification of failure deviation can be applied to different kinds of equipment systems to form a standardized analysis process.

**Table 1.** Classification of failure deviation.

| Primary Classification | Secondary Classification | Description |
|---|---|---|
| Logic | Reverse | Contrary to normal operation logic |
| | Wrong material | Input/output/use the wrong material |
| Time | Start too early | Earlier before normal beginning time |
| | Start too late | Later than normal beginning time |
| | End too early | Earlier before normal ending time |
| | End too late | Later than normal ending time |
| | Out of sequence | A sequential error between two actions |
| Value | Too high | The current value is greater than normal |
| | Too low | The current value is less than normal |
| | None | Numerical is zero |
| | Intermittent | Numerical fluctuation |

### *3.2. Description Method of Failure Logic*

Failure logic analysis is the core of the extended FFIP method. It is important to analyze and express the failure logic of systems and components. Mathematical logic is a logical reasoning that uses mathematical symbols to describe objective objects [43]. In mathematical logic, generally, "1" and "0" are used to represent "true" and "false". Symbols such as "and", "or" and "not" are used to represent the operation rules, to realize the transformation from complex logical operation to simple numerical calculation. In this paper, mathematical logic is adopted to describe the failure logic of systems and components. Thus, the complex functions of a system can be expressed in a specific way, and the readability of information is increased with mathematical logic. Through the guideline on the specific mathematical expression in failure logic, the designer and the user can unify their expression for further communication and modification. Common connectives in mathematical logic are $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$. The method of using mathematical logic to express failure logic is shown as follows.

First, the failure deviations are classified according to the failure modes that may occur in the system. It is described with the symbol of mathematical logic, as shown in Table 2.

**Table 2.** Mathematical description of failure deviation.

| Primary Classification | Secondary Classification | Mathematical Description |
|---|---|---|
| Normal | / | $\{P, (t_1, t_2), n_0\}$ |
| Logic | Reverse | $\{\neg P, (t_1, t_2), n_0\}$ |
| | Wrong material | $\{Error, (t_1, t_2), n_0\}$ |
| Time | Start too early | $\{P, (t_0 < t_1, t_2), n_0\}$ |
| | Start too late | $\{P, (t_0 > t_1, t_2), n_0\}$ |
| | End too early | $\{P, (t_1, t_n < t_2), n_0\}$ |
| | End too late | $\{P, (t_1, t_n > t_2), n_0\}$ |
| | Out of sequence | $\{P, (t_0 = variable, t_n = variable), n_0\}$ |
| Value | Too high | $\{P, (t_1, t_2), n > n_0\}$ |
| | Too low | $\{P, (t_1, t_2), n < n_0\}$ |
| | None | $\{P, (t_1, t_2), n = 0\}$ |
| | Intermittent | $\{P, (t_1, t_2), n = variable\}$ |

Where P: the component behavior under normal conditions; $t_1$: the time when the behavior starts under normal conditions; $t_2$: the time when the behavior ends under normal conditions; $t_0$: the time when the behavior starts in the case of failure; $t_n$: the time when the behavior ends in the case of failure; $n_0$: the size of the quantity under normal conditions; n: the size of the quantity in the case of failure.

Then, according to the formal graphical representation method [44,45], the conversion method of failure logic is expressed. As shown in Figure 2, we use circles to indicate the state of the component and arrows to indicate the transition relationship between states. The content on the arrow is the mathematical description of the failure logic using the failure deviation, and the conversion condition of the state is expressed in simple and clear language.

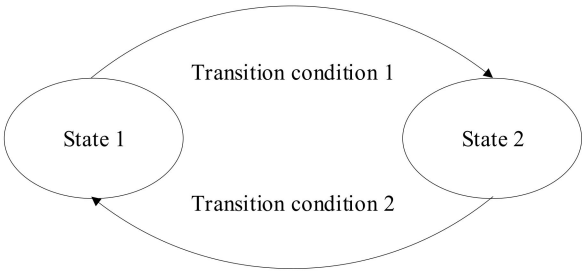



**Figure 2.** Failure logic diagram.

## 4. Improvement of SysML Based on Failure Logic

### *4.1. Conversion Method from FFIP to SysML*

In this paper, we combine the FFIP method with SysML and use different graphs of SysML to demonstrate the FFIP model. Using SysML to represent the structure and function model of FFIP can make the information flow and function flow more closely related. Additionally, the use of SysML can facilitate communication and make design easier between system users, maintenance personnel, and those who may debug, integrate, extend, modify and change the system [46]. SysML is mainly composed of nine types of diagrams: activity diagrams, sequence diagrams, state machine diagrams, use case diagrams, requirement diagrams, block definition diagrams, internal block diagrams, parameter diagrams and package diagrams [47]. The mapping relationship between FFIP and SysML elements is shown in Table 3.

**Table 3.** Mapping relationship between FFIP elements and SysML elements.

| FFIP Elements | SysML Elements | SysML Symbol | Explanation of Meaning |
|---|---|---|---|
| Component | Block | «Block» **Block_1** | It represents the events in the process dimension and the people, machines, and environment in the object dimension. |
| Information | Attributes | «valueType» **Block_1** *values* | It describes events, as well as the value attributes of people, machines, and environments. |
| Function | Operation | «valueType» **Block_1** *Operation* | It describes events, as well as the functional attributes of people, machines, and environments. |
| Behavior | Message | → | It describes the relationship between basic events and people, machines, and environment factors. |
| Reply | Reply message | ⇠ | It describes the feedback of behavior. |
| Structure connection | Directed Association | ◆— | It describes the relationship between the events and their processes. |
| State transfer | Transition | ⤷ | It describes the transition conditions between different states. |

The block definition diagrams (BDD) in SysML define the attributes of modules and the relationships among them, which includes association, generalization, and dependency. The block attributes of BDD can be divided into two types: structure attributes and behavior attributes. The value attribute of a structure attribute can be used to represent the output signal and flow value of the module. Moreover, the operation attribute of the behavior attribute can be used to represent the functional behavior of the module. Therefore, we can use BDD to describe the structure and function model of FFIP. In this way, the structure model and function model of FFIP can be represented in the same BDD of SysML. The advantage of using BDD is that the structure, function, and data flow can be

expressed at the same time, which makes the functional design of components clearer and easier to communicate and modify by different users.

Sequence diagrams (SD) in SysML can describe the sequence of actions and information flow transmission. There are two advantages of using sequence diagrams. They can describe the behavior rules in the behavior models of FFIP and can also describe the information transfer relationship between modules. Using the SD of SysML presents the complex task process in a more organized way, which makes up for the disadvantages of using tables to list behavior rules in the FFIP method.

### 4.2. Improvement of SysML State Machine Diagram

In the previous section, we use the BDD and SD in SysML to make a standardized modeling of the structure model, function model and part of the behavior model in FFIP. What has not been built in the behavior model is failure logic. Failure logic is the core of the extended FFIP method. However, none of the nine diagrams in SysML can describe failure logic. To combine SysML with FFIP, this paper extends the elements of SysML to describe failure logic.

State machine diagrams (SMD) in SysML are mainly used to describe the life cycles of objects, subsystems and systems [48]. Through the SMD, we can know all the states that an object can reach and the impact of events on the state. Therefore, the SMD can represent the state transition of components and the transformation relationship between different states.

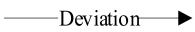
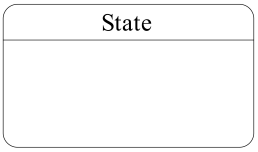
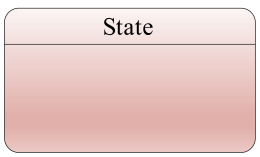
In this paper, we divided the functional states of components into three categories: nominal state, failure state and degradation state. The nominal state is the normal operation state of the system. Failure state refers to the state in which the system produces some failure when the deviation exceeds a certain value. Degradation state refers to a state between nominal and failure state, which has not had a significant impact but deviates from normal performance. Failure logic can be represented using key process variables. The logical relationship between the input, output, and deviation of variables can explain the current state of the component, and the deviation can express the internal failure of component behavior and the effect of other components.

However, we need more information in the state diagrams to describe the failure logic:

- the representation of deviation;
- the representation of different state types of components.

Table 4 shows the extension of SysML elements to meet the above two requirements.

**Table 4.** The extended elements of SysML.

| SysML Elements | SysML Symbol | Explanation of Meaning |
| --- | --- | --- |
| Numerical deviation |   | The deviation from the current value |
| Normal state |   | The state of the component when it is running normally. |
| Abnormal state |   | A block turns red after a component that has deviated indicates an exception. |

When we use the SMD to represent the FFIP behavior model, the state blocks are used to represent different states of the components. The arrows are used to show the factors that cause the state transition. The text on the arrow shows the mathematical logic to describe

the conditions for triggering failure. The SMD of SysML can better show the transformation relationship of failure logic, making the models more coherent and readable.

## 5. Improved FFIP Method

According to the above improvements, the proposed FFIP method is mainly divided into the following five steps:

Step 1: Establish structure model and function model

Use the BDD of SysML to build the structure model and function model in FFIP. These two models describe the structural configuration and functional relationship between system components. In addition to the two models, information flow and function flow of a component could be built into one graph more explicitly, making the model more coherent.

Step 2: Introduce deviation of key process variables

Select the key process variables that can reflect the behavior state of components. Then, calculate the deviation with formula (1). The deviation value can reflect the change of component state.

$$D = \left| \frac{RV - DV}{DV} \right| \times 100\% \quad (1)$$

where $D$ is the deviation; $RV$ is the present value; $DV$ is the design value.

Step 3: Build behavior model

The behavior model includes behavior rules and failure logic. Behavior rules represent the activities of system components, as well as the inputs and outputs that lead to the activities. The SD in SysML is used to build the behavior rules in FFIP, which can better emphasize the occurrence order of behaviors and help to clarify the logic. Failure logic describes the relationship between the cause, process, and consequences of component failures. The state and state transformation conditions of the component also need to be analyzed. According to Table 2, using mathematical logic, we express the transformation conditions of states. Then, we describe the state and transformation relationships of the components, in combination with the improved state machine diagrams in SysML.

Step 4: Establish the simulation model and simulate the failure

According to the established FFIP models, the nominal model and failure model are established in Simulink. The failure injection method is used to simulate the failure.

Step 5: Analyze the simulation results and output the fault tree.

Using the fault propagation paths obtained by simulation, the fault tree is generated, and the minimum cut set is calculated to obtain the key failure components.

## 6. Case Study

In this paper, we take the wheel brake system in the appendix of ARP 4761 as a case to apply the above method.

### *6.1. Structural and Functional Model*

In the traditional FFIP method, the construction of the structure and function model is represented in the form of a flow chart, as shown in Figures 3 and 4.

In the improved FFIP method, SysML is used to describe the structure and function model of FFIP, as shown in Figure 5.

The sequence diagram of SysML can explain the behavior rules in FFIP, as shown in Figure 6. This model represents the following behavior process. After the operator presses the pedal, the BSCU subsystem receives the work instruction and transmits the signal outwardly. The selector valve receives an S1 signal and selects the green pump subsystem. The green pump subsystem returns error data to the selector valve, making it select the blue pump subsystem. The blue pump subsystem returns the correct data. Then, the blue pump subsystem performs the task of controlling the wheel rotation. Finally, the

correct operation of the wheel is generated and fed back to the operator. Thus, this task has been completed.

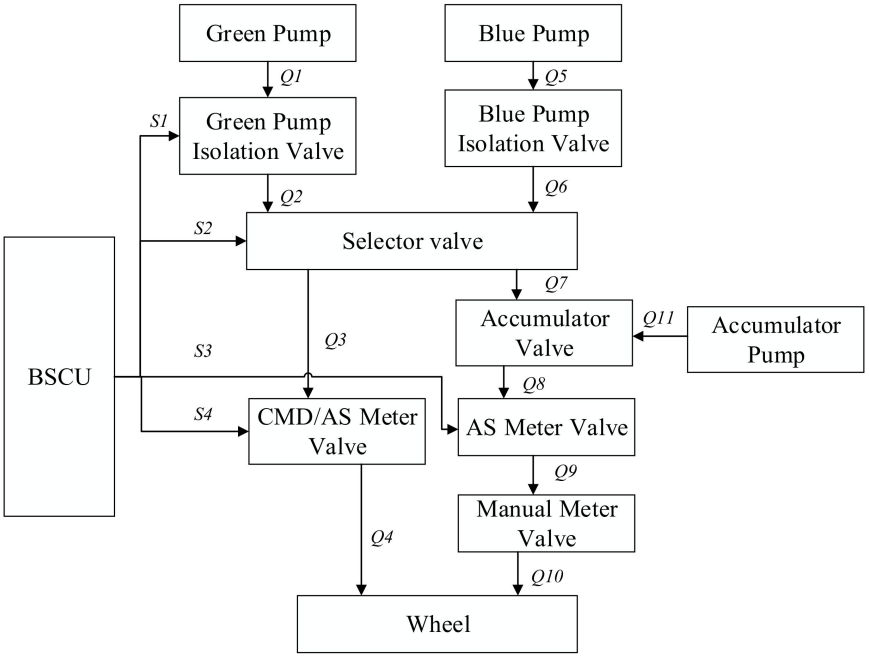



**Figure 3.** Structure Model of Brake System. ($Q_1$–$Q_{11}$: the flow of the components; $S_1$–$S_4$: the signal of the components.)

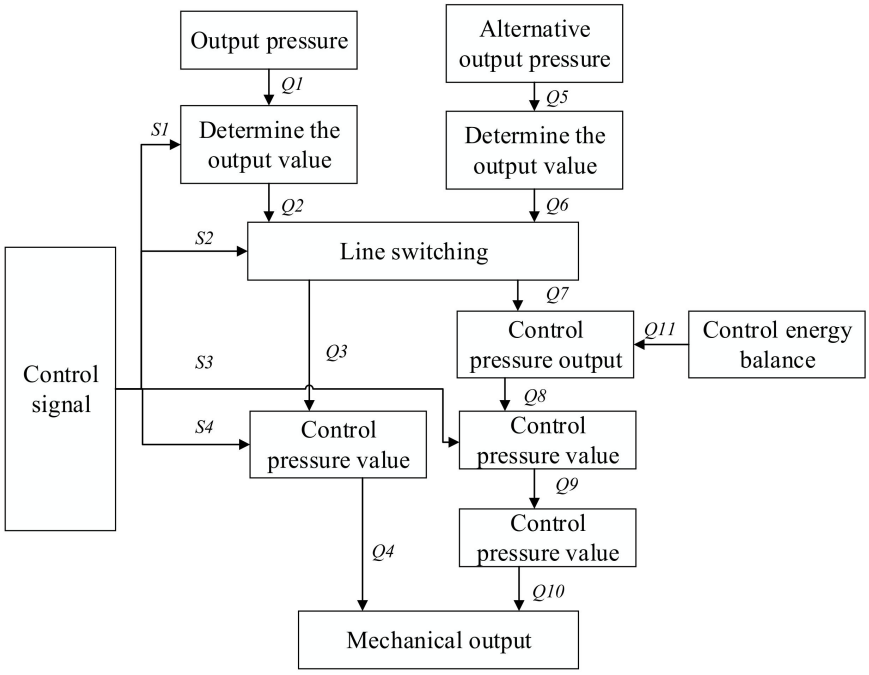



**Figure 4.** Function Model of Brake System. ($Q_1$–$Q_{11}$: the flow of the components; $S_1$–$S_4$: the signal of the components.)

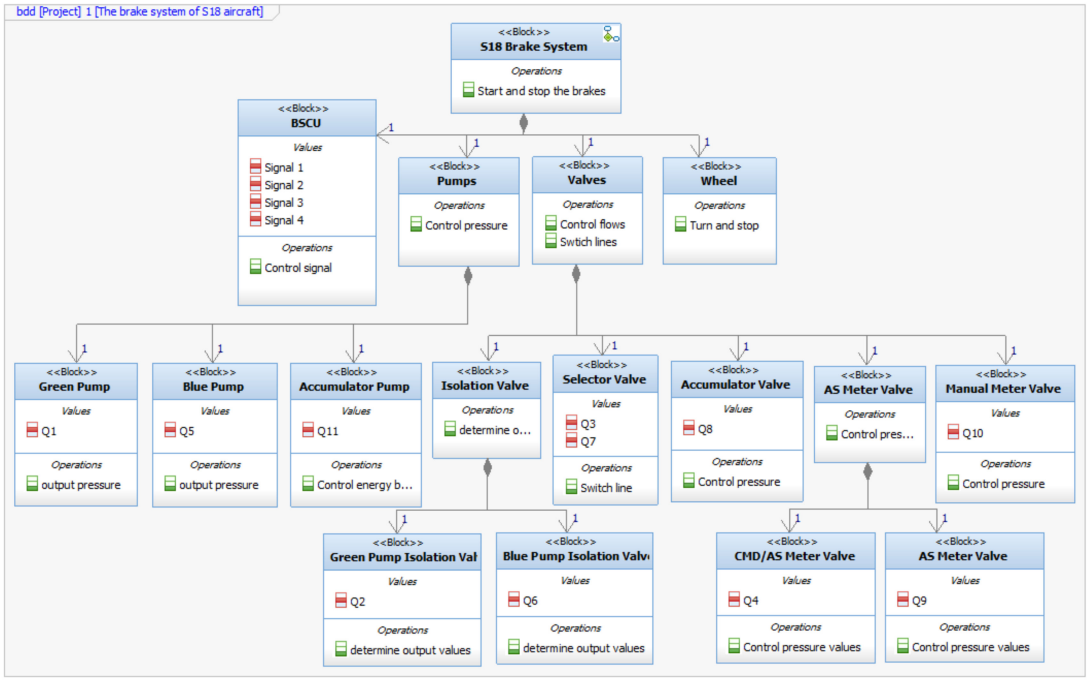



**Figure 5.** BDD of brake system.

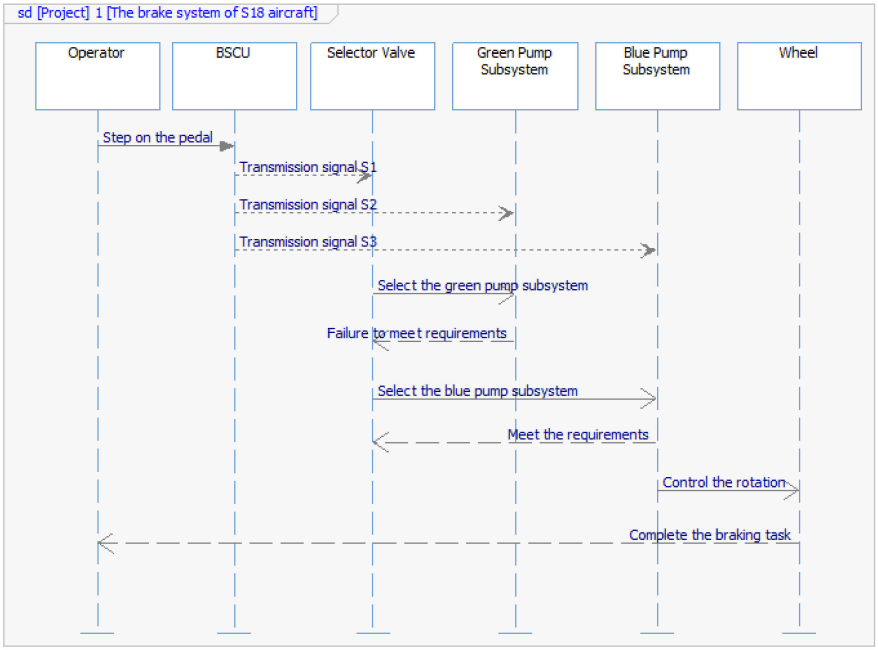



**Figure 6.** Sequence diagram of brake system.

The SMD of SysML can show the different states of the components, and the transformation relationship between them. The failure logic of the brake system described by the SMD is shown in Figure 7. In this case, the key process variables are quantity of flow: Q, pressure: P and time: T. The arrows represent transitions between different states, and the text above them is a mathematical description of the failure deviation.

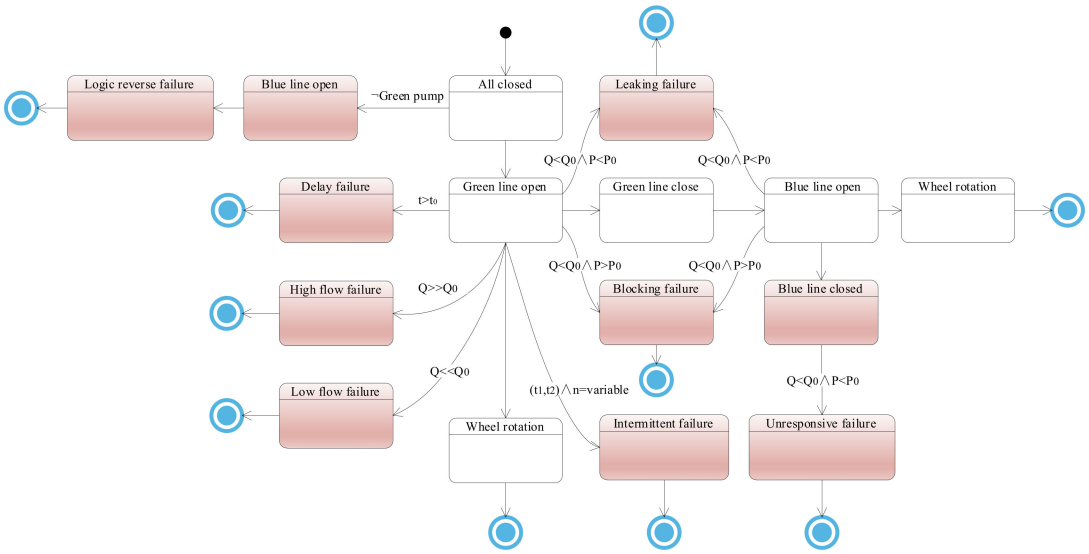



**Figure 7.** State machine diagram of selector valve module.

### *6.2. Simulation Model and Failure Simulation*

According to the structure model, function model and behavior model, the simulation model of the S18 brake system is established by using the Simulink. Firstly, the nominal model, representing the system under normal operation, is established according to the structure model and the function model. Then, according to the failure logic in the behavior model, the fault injection method is used to establish the failure extended model, as shown in Figure 8. The system can simulate 39 single failures and 741 double failures.

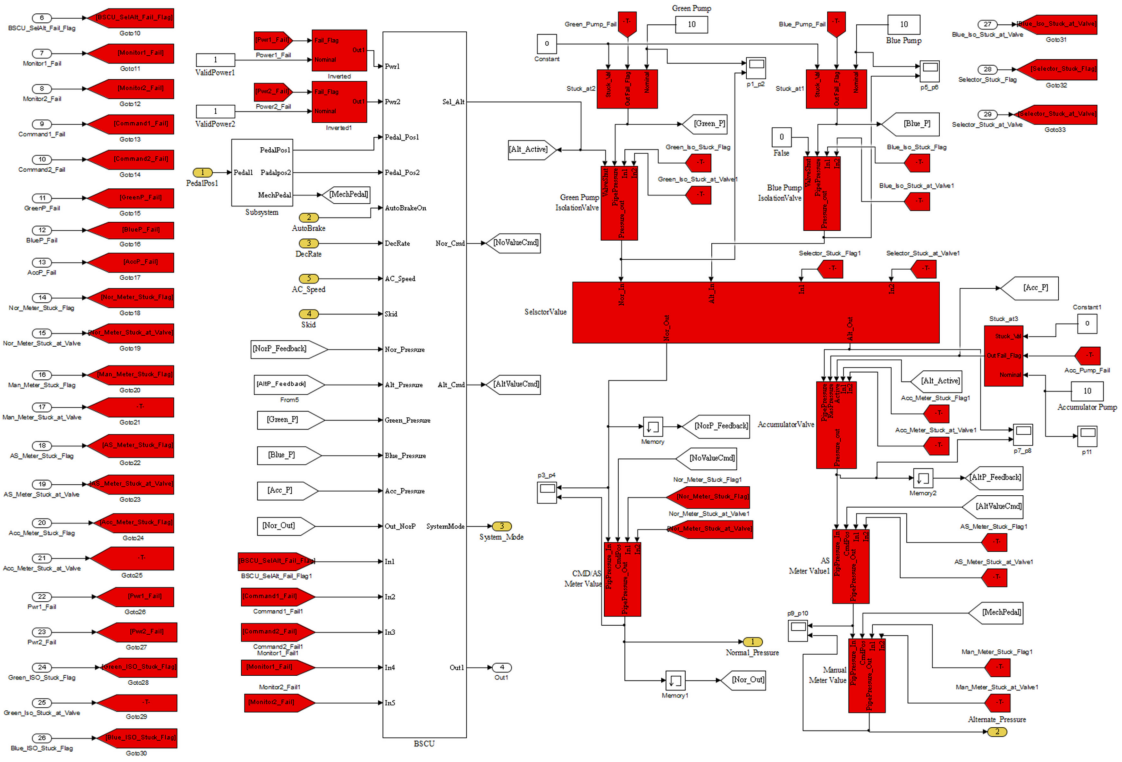


**Figure 8.** Failure extended simulation model of S18 brake system.

### *6.3. Result*

- Single failure simulation

Single failure simulation is to simulate the system situation when one failure module is set to "1" and other failure modules are set to "0" each time. Limited by the length of the paper, only 9 typical single failures are selected for description in this section, as shown in Table 5. The single failures shown in the table are as follows.

**Table 5.** Component States under single fault simulation.

| | | Failure 1 | Failure 2 | Failure 3 | Failure 4 | Failure 5 | Failure 6 | Failure 7 | Failure 8 | Failure 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| BSCU | Command | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| | Monitor | $Z_1$ | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| | SystemModel_SelCmd | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| | AutoBraking_PressureOut | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Green Pump | | $Z_0$ | $Z_0$ | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Green Pump Isolation Valve | | $Z_0$ | $Z_0$ | $Z_2$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Selector Valve | | $Z_0$ | $Z_0$ | $Z_0$ | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| CMD/AS Meter Valve | | $Z_1$ | $Z_1$ | $Z_0$ | $Z_2$ | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Blue Pump | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_1$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Blue Pump Isolation Valve | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_1$ | $Z_0$ | × | $Z_0$ | $Z_0$ | $Z_0$ |
| Accumulator Valve | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_2$ | $Z_0$ | $Z_1$ | × | $Z_0$ | $Z_1$ |
| AS Meter Valve | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_2$ | $Z_0$ | $Z_0$ | $Z_1$ | × | $Z_0$ |
| Manual Meter Valve | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_2$ | $Z_0$ | $Z_0$ | $Z_1$ | $Z_1$ | $Z_0$ |
| Accumulator Pump | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_1$ | $Z_0$ | $Z_1$ | $Z_0$ | $Z_1$ | × |
| System | | $Z_1$ | $Z_1$ | $Z_0$ | $Z_2$ | $Z_1$ | $Z_1$ | $Z_1$ | $Z_1$ | $Z_1$ |

Where: × indicates the failure location, $Z_0$ indicates normal function, $Z_1$ indicates degradation function, and $Z_2$ indicates failure functional.

- Failure 1: Command does not respond,
- Failure 2: Monitor's logic is opposite,
- Failure 3: Green pump is blocked,
- Failure 4: Selector valve's logic is opposite,
- Failure 5: CMD/AS meter valve opened too late,
- Failure 6: Blue pump isolation valve leakage,
- Failure 7: Accumulator valve opened early,
- Failure 8: AS meter valve does not respond,
- Failure 9: Accumulator pump leakage.

It should be noted that the example here only lists one failure mode for each component. There are still other failure modes for the components. Other failure deviations that may be involved in components are shown in Table 1.

- Double failure simulation

Double failure simulation is to simulate the system situation when two failure modules are set to "1" and other failure modules are set to "0" each time. Also due to space limits, this paper selects 8 typical double failures to explain, as shown in Table 6.

### *6.4. Further Analysis*

According to the results of the single-failure and double-failure simulations, the fault propagation paths of the brake system can be deduced. Through analyzing the fault propagation paths, the fault tree of the brake system can be obtained, as shown in Figure 9. The propagation of single failure is represented by an OR gate in the fault tree, while the propagation of double failure is represented by an AND gate.

**Table 6.** Component States under double fault simulation.

| | | Failure 10 | Failure 11 | Failure 12 | Failure 13 | Failure 14 | Failure 15 | Failure 16 | Failure 17 |
|---|---|---|---|---|---|---|---|---|---|
| BSCU | Command | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| | Monitor | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| | SystemModel_SelCmd | $Z_2$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| | AutoBraking_PressureOut | $Z_2$ | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Green Pump | | $Z_0$ | × | × | × | × | × | × | $Z_0$ |
| Green Pump Isolation Valve | | $Z_2$ | $Z_2$ | × | $Z_2$ | $Z_2$ | $Z_2$ | $Z_2$ | $Z_0$ |
| Selector Valve | | $Z_2$ | $Z_1$ | $Z_0$ | $Z_2$ | $Z_1$ | $Z_0$ | $Z_0$ | $Z_0$ |
| CMD/AS Meter Valve | | $Z_2$ | $Z_1$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Blue Pump | | $Z_0$ | $Z_0$ | $Z_0$ | × | $Z_0$ | $Z_0$ | $Z_0$ | $Z_0$ |
| Blue Pump Isolation Valve | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_2$ | × | $Z_0$ | $Z_0$ | $Z_0$ |
| Accumulator Valve | | $Z_0$ | $Z_0$ | $Z_0$ | $Z_2$ | $Z_1$ | × | $Z_0$ | × |
| AS Meter Valve | | $Z_2$ | $Z_1$ | $Z_0$ | $Z_2$ | $Z_1$ | $Z_1$ | × | $Z_1$ |
| Manual Meter Valve | | $Z_1$ | $Z_1$ | $Z_0$ | $Z_2$ | $Z_1$ | $Z_1$ | $Z_1$ | $Z_1$ |
| Accumulator Pump | | $Z_1$ | $Z_0$ | $Z_0$ | $Z_2$ | $Z_1$ | $Z_1$ | $Z_0$ | × |
| System | | $Z_2$ | $Z_1$ | $Z_0$ | $Z_2$ | $Z_1$ | $Z_1$ | $Z_1$ | $Z_1$ |

Where: × indicates the failure location, $Z_0$ indicates normal function, $Z_1$ indicates degradation function, and $Z_2$ indicates failure functional.

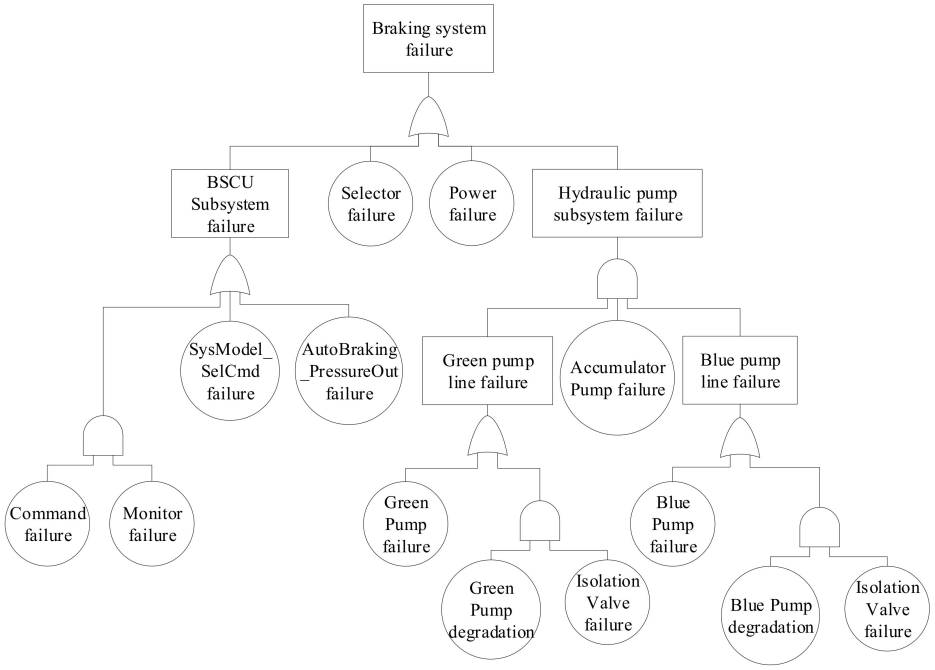



**Figure 9.** Fault tree of brake system.

Next, we use letters and numbers to present the failure events, as shown in Figure 10. The purpose is to facilitate the calculation of the minimum cut set.

Through calculation, the minimum cut set is obtained as follows: {A1}, {A2}, {A3}, {A4}, {A6, a7}, {A5, A8, A9}, {A5, A8, A12}, {A5, A8, A13}, {A5, A9, A10}, {A5, A10, A12}, {A5, A10, A13}, {A5, a11, a11}, {A5, a11, A13}.

Based on the minimum cut set, it can be concluded that the key events are A1, A2, A3, A4 and A5. They represent the failure of the selection valve, power, System Model_SelCmd module, Auto Braking_Pressure Out module and accumulator pump. SystemModel_SelCmd and AutoBraking_PressureOut are two modules in BSCU modeling, so it is not considered as the key failure position. In addition, power-off belongs to external factors of equipment, so it is not considered. On the other hand, since A8 and A10 are related to the green hydraulic pump, and A9 and A11 are related to the blue hydraulic pump, the failure of these two hydraulic pumps also plays a key role in the whole system.

Therefore, the key failure parts are the hydraulic pumps and the selector valve. Thus, in the process of equipment operation, more attention should be paid to the maintenance and repair of hydraulic pumps and the selector valve, to ensure the safety of the system.

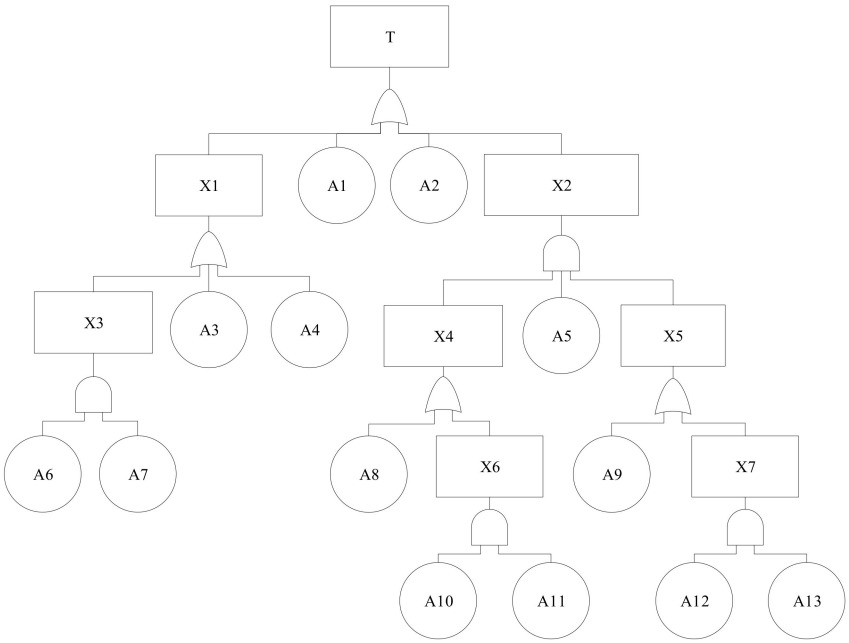



**Figure 10.** Simplified fault tree of brake system.

## 7. Conclusions

In this paper, an improved method of safety analysis is proposed by combining FFIP with mathematical logic and SysML. The following problems are solved.

1. The FFL in FFIP is extended, the concept of deviation is introduced. Now the FFIP method is suitable for failure analysis in every stage of the system.
2. Mathematical logic is used to describe the failure logic, solved the problem of expressing FFL in FFIP.
3. FFIP is combined with SysML. SysML is used to help the FFIP to establish the structure, function, and behavior model. At the same time, the SMD in SysML is improved by extending the elements in SysML to analyze and describe the failure logic.
4. A more detailed path of fault propagation is obtained by simulating the single failure and double failures of the brake system. Additionally, the fault tree is generated, and the key failure locations are calculated according to the fault propagation paths of FFIP.

In this paper, we have solved the problems of component failure and failure propagation of the system. Next, we will focus on the failure events that:

(1) do not follow nominal flow paths; and
(2) do not follow any established flow paths.

These researches will help people to predict more failure situations and propose improvement measures in the early design stage of the system.

**Author Contributions:** J.J. and T.Z. prepared the conception and the formal description of the proposed solution; they both performed the literature review; J.C. and Y.J. studied the case; S.P. and J.J. implemented the proposed solution and wrote the paper. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare that there is no conflict of interest.

## References


1. Zhao, T.D.; Jiao, J.; Bao, X.H. *Principles of Safety*; National Defense Industry Press: Beijing, China, 2018; pp. 146–179.
2. Salehi, V.; Veitch, B.; Smith, D. Modeling complex socio-technical systems using the FRAM: A literature review. *Hum. Factors Ergon. Manuf. Serv. Ind.* **2021**, *31*, 118–142. [CrossRef]
3. Longji, D.A.; Emilia, V. System safety assessment based on STPA and model checking. *Saf. Sci.* **2018**, *109*, 130–143.
4. Mažeika, D.; Butleris, R. MBSEsec: Model-Based Systems Engineering Method for Creating Secure Systems. *Appl. Sci.* **2020**, *10*, 2574. [CrossRef]
5. Chen, L.; Jiao, J.; Zhao, T.D. Review for model-based safety analysis of complex safety-critical system. *J. Syst. Eng. Electron.* **2017**, *39*, 1287–1291.
6. Grunske, L.; Han, J. A Comparative Study into Architecture-Based Safety Evaluation Methodologies Using AADL's Error Annex and Failure Propagation Models. In Proceedings of the 2008 11th IEEE High Assurance Systems Engineering Symposium, Nanjing, China, 3–5 December 2008.
7. Grunske, L.; Kaiser, B. Automatic generation of analyzable failure propagation models from component-level failure annotations. In Proceedings of the Fifth International Conference on Quality Software (QSIC′05), Melbourne, VIC, Australia, 19–20 September 2005.
8. Wallace, M. Modular Architectural Representation and Analysis of Fault Propagation and Transformation. *Electron. Notes Comput. Sci.* **2005**, *141*, 53–71. [CrossRef]
9. Sharvia, S.; Kabir, S.; Walker, M.; Papadopoulos, Y. Model-based dependability analysis. *Softw. Qual. Assur.* **2016**, *12*, 251–278.
10. Papadopoulos, Y.; Walker, M.; Parker, D.; Sharvia, S.; Bottaci, L.; Kabir, S.; Azevedo, L.; Sorokos, I. A synthesis of logic and bio-inspired techniques in the design of dependable systems. *Annu. Rev. Control* **2016**, *41*, 170–182. [CrossRef]
11. Dong, H.; Gu, Q.; Wang, G.; Zhai, Z.; Lu, Y.; Wang, M. Availability Assessment of IMA System Based on Model-Based Safety Analysis Using AltaRica 3.0. *Processes* **2019**, *7*, 117. [CrossRef]
12. Brameret, P.A.; Rauzy, A.; Roussel, J.M. Automated generation of partial Markov chain from high level descriptions. *Reliab. Eng. Syst. Safe* **2015**, *139*, 179–187. [CrossRef]
13. Chen, L.; Jiao, J.; Fan, J.; Ren, F. A fault propagation modeling and analysis method based on model checking. In Proceedings of the 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, USA, 25–28 January 2016.
14. Chen, L.; Jiao, J.; Wei, Q.X. Model-checking oriented unified modeling method based on NuSMV. *Syst. Eng. Electron.* **2018**, *40*, 1654–1659.
15. Wei, Q.; Jiao, J.; Zhao, T. Flight control system failure modeling and verification based on SPIN. *Eng. Fail. Anal.* **2017**, *82*, 501–513. [CrossRef]
16. Wei, Q.; Jiao, J.; Fan, J.; Zhao, T. An Optimized Method for Generating Fault Tree from a Counter-Example. In Proceedings of the 2016 Annual Reliability and Maintainability Symposium (RAMS), Tucson, AZ, USA, 25–28 January 2016.
17. Kurtoglu, T.; Tumer, I.Y. A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems. *J. Mech. Des.* **2008**, *130*, 051401. [CrossRef]
18. Kurtoglu, T.; Tumer, I.Y.; Jensen, D.C. A functional failure reasoning methodology for evaluation of conceptual system architectures. *Res. Eng. Des.* **2010**, *21*, 209–234. [CrossRef]
19. Bello, O.O. Developing Methods of Obtaining Quality Failure Information from Complex Systems. Ph.D. Thesis, University of Arkansas, Fayetteville, Arkansas, 2017.
20. Papakonstantinou, N.; Sierla, S.; O'Halloran, B.; Tumer, I.Y. A Simulation Based Approach to Automate Event Tree Generation for Early Complex System Designs. In Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, OR, USA, 4–7 August 2013.
21. Papakonstantinou, N.; Proper, S.; O'Halloran, B.; Tumer, I.Y. Simulation Based Machine Learning for Fault Detection in Complex Systems Using the Functional Failure Identification and Propagation Framework. In Proceedings of the International Design Engineering Technical Conferences, Buffalo, NY, USA, 17–20 August 2014.
22. Tumer, I.; Smidts, C. Integrated Design-Stage Failure Analysis of Software-Driven Hardware Systems. *IEEE Trans. Comput.* **2011**, *60*, 1072–1084. [CrossRef]
23. Irshad, L.; Ahmed, S.; Demirel, H.O.; Tumer, I.Y. Computational Functional Failure Analysis to Identify Human Errors during Early Design Stages. *J. Comput. Inf. Sci. Eng.* **2019**, *19*, 031005. [CrossRef]
24. Dempere, J.; Papakonstantinou, N.; O'Halloran, B. Risk Modeling of Variable Probability External Initiating Events in a Functional Modeling Paradigm. In Proceedings of the 2017 Annual Reliability and Maintainability Symposium (RAMS), Orlando, FL, USA, 23–26 January 2017.
25. O'Halloran, B.M.; Papakonstantinou, N.; Giammarco, K.; Van Bossuyt, D.L. A graph theory approach to predicting functional failure propagation during conceptual systems design. *Syst. Eng.* **2021**, *24*, 100–121. [CrossRef]
26. Delligatti, L.; Hou, B.W.; Zhu, Y.L. *SysML Distilled: A Brief Guide to the Systems Modeling Language*; China Machine Press: Beijing China, 2014; pp. 11–17.

27. Zhou, S.; Sun, Q.; Jiao, J. A safety modeling method based on SysML. In Proceedings of the 2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS), Guangzhou, China, 6–8 August 2014.
28. Baklouti, A.; Nguyen, N.; Mhenni, F.; Choley, J.Y.; Mlika, A. Improved Safety Analysis Integration in a Systems Engineering Approach. *Appl. Sci.* **2019**, *9*, 1246. [CrossRef]
29. Mehrpouyan, H.; Jensen, D.C.; Hoyle, C.; Tumer, I.Y.; Kurtoglu, T. A Model-Based Failure Identification and Propagation Framework for Conceptual Design of Complex Systems. In Proceedings of the ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, IL, USA, 12–15 August 2012.
30. Kramer, S.; Tumer, I.Y. Towards StateCharts Based Failure Propagation Analysis for Designing Embedded PHM Systems. In Proceedings of the The 2009 Prognostics and Health Management Conference, San Diego, CA, USA, January 2009.
31. Chen, L.; Jiao, J.; Wei, Q.; Zhao, T. An improved formal failure analysis approach for safety-critical system based on MBSA. *Eng. Fail. Anal.* **2017**, *82*, 713–725. [CrossRef]
32. Gero, J.S.; Kannengiesser, U. The Situated Function-Behaviour-Structure Framework. *Des. Stud.* **2004**, *25*, 373–391. [CrossRef]
33. Zhang, Y.; Wang, J.; Zhao, Y.; Huo, D. Element-based knowledge representation in intelligent design and its prototype for knowledge processing. In Proceedings of the 2014 20th International Conference on Automation and Computing, Cranfield, UK, 12–13 September 2014.
34. Zhu, Y.P. A Research on State-Oriented Safety Formal Modeling of Complex Equipment Fault. Master's Thesis, Beihang University, Beijing, China, 2018.
35. Friedenthal, S. *A Practical Guide to SysML: The Systems Modeling Language*; Morgan Kaufmann: Burlington, MA, USA, 2015; pp. 17–21.
36. Berriche, A.; Mhenni, F.; Mlika, A.; Choley, J.Y. Towards Model Synchronization for Consistency Management of Mechatronic Systems. *Appl. Ences.* **2020**, *10*, 3577. [CrossRef]
37. Zhou, M.; Wei, H.P.; Zhang, H. *Modern Equipment Engineering*; Metallurgical Industry Press: Beijing, China, 2011; p. 74.
38. Ma, L.L.; Guo, K.J.; Wang, J.Z. A Fault Diagnosis Method of Vehicle Transmission System Based on Improved SVM. *Trans. Beijing Inst. Technol.* **2020**, *8*, 856–860.
39. Chen, J.S.; Li, Y.Q.; Yan, K.; Wei, X. Classification and Prevention Improvement of Hydraulic Cylinder Fault Mode. *Sci. Technol. Vis.* **2020**, *11*, 16–17.
40. O'Halloran, B.M.; Jensen, D.C.; Tumer, I.Y.; Kurtoglu, T.; Stone, R.B. A framework to generate fault-based behavior models for complex systems design. In Proceedings of the 2013 Proceedings Annual Reliability and Maintainability Symposium (RAMS), Orlando, FL, USA, 28–31 January 2013.
41. Zurita, N.F.S.; Stone, R.B.; Demirel, O.; Tumer, I.Y. The Function-Human Error Design Method (FHEDM). In Proceedings of the ASME 2018 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Quebec, QC, Canada, 26–29 August 2018.
42. Pang, S.J.; Jiao, J. An Improved Model of Functional Failure Identification and Propagation Based on Mathematical Logic. In Proceedings of the 67th Annual Reliability and Maintainability Symposium (RAMS 2021), Orlando, FL, USA, 24–27 May 2021.
43. Natalie Ott, R.B.M.V. Multiple symbolic representations: The combination of formula and text supports problem solving in the mathematical field of propositional logic. *Learn Instr.* **2018**, *58*, 88–105.
44. Chen, X.; Jiao, J. A fault propagation modeling method based on a finite state machine. In Proceedings of the 2017 Annual Reliability and Maintainability Symposium (RAMS), Orlando, FL, USA, 23–26 January 2017.
45. Wang, J.; Zhan, N.J.; Feng, X.Y.; Liu, Z.M. Overview of Formal Methods. *J. Softw.* **2019**, *3*, 51–54.
46. Cao, Y.; Liu, Y.; Paredis, C.J.J. System-level model integration of design and simulation for mechatronic systems based on SysML. *Mechatronics* **2011**, *21*, 1063–1075. [CrossRef]
47. Salado, A.; Wach, P. Constructing True Model-Based Requirements in SysML. *Systems* **2019**, *7*, 19. [CrossRef]
48. Jacobs, J.; Simpson, A. On the formal interpretation and behavioural consistency checking of SysML blocks. *Softw. Syst. Modeling* **2017**, *16*, 1145–1178. [CrossRef]