Cascading Failure Analysis Method of Avionics Based on Operational Process State

YUQIAN WU, (Graduate Student Member, IEEE), GANG XIAO, AND MIAO WANG
School of Aeronautics and Astronautics, Shanghai Jiao Tong University, Shanghai 200240, China

Corresponding author: Miao Wang (miaowang@sjtu.edu.cn)

This paper was sponsored by National Program on Key Basic Research Project (2014CB744903), National Natural Science Foundation of China (61673270), Natural Science Foundation of Shanghai (20ZR1427800), New Young Teachers Launch Program of Shanghai Jiaotong University (20X100040036), Shanghai Pujiang Program (16PJD028), Shanghai Industrial Strengthening Project (GYQJ-2017-5-08), Shanghai Science and Technology Committee Research Project (17DZ1204304), and Shanghai Engineering Research Center of Civil Aircraft Flight Testing.

ABSTRACT In the context of functional integration of avionics, the complex operational process interaction has increased the complexity of the cascading failure analysis, which is significant to evaluate the overall safety status and design rationality. In order to achieve a dynamic and dimensional evaluation of cascading failures, this paper proposes a safety analysis method based on operational process state, with an operational process-orientated hierarchical system functional framework established by means of the state machine. Then, the cascading failure causation dynamic search algorithms and the cascading failure causation tree structure are designed respectively to describe the failure propagation in combination. Consequently, cascading failure propagation paths and minimum cut sets are generated automatically based on the search result. By using the aircraft integrated surveillance system as a research object, the effect and cause of cascading failure assessed in various failure scenarios have validated the effectiveness of the proposed method, and the comparison against the existing model based safety analysis methods demonstrates the higher flexibility and efficiency of the proposed method. The proposed method enables the dynamic and overall assessment of system safety status concentrating on the cascading failure utilizing the operational process state, and further enhances the systemization and automation level of the safety evaluation process in the early development phase.

INDEX TERMS Aerospace safety, aerospace and electronic systems, air safety, avionics, cascading failure, model-based safety analysis, operational process modeling, state machine, system engineering and theory.

I. INTRODUCTION

With the increasing complexity of avionics, cascading failure [1] has been used as a significant aspect in safety analysis, which provides the global effect assessment of a local failure, thus facilitating estimation of design rationality and optimization of the system configuration in the early development stage.

Traditionally, the safety models are constructed manually by the fault tree analysis (FTA) [2], failure modes and effects analysis (FMEA) based on engineering experience [3]. However, the safety analysis process, especially the cascading failure propagation, is not trackable by the existing static analysis methods, because of the gap between the safety model and system design mechanism. Therefore, the accuracy of failure causality analysis needs to be further improved by combining the system design process with the safety analysis process.

Accordingly, the model-based safety analysis (MBSA) [4] has been developed and adopted successfully in many fields [5]–[9]. During the MBSA process, design engineers and safety analysts share the extended system model considering failure situations, which provides versatile measures to analyze the relationships between cascading failure. In specific, FTPN [10], HiP-HOPS [11], [12], and FPTC [13] are commonly used methods that focus on the logical relationships between failure modes to demonstrate failure propagation. However, the operational modes and dynamic behaviors cannot be demonstrated flexibly under various functional integration mechanisms since the causal flows among the components and failure modes are inherent within the methods.
With the aim to strengthen the significance of dynamic behaviors in cascading failure analysis, the degradation-related methods have been introduced to address the mentioned problem. The dynamic-Bayesian-network (DBN) is one of the efficient tools for fault diagnosis and propagation analysis in the time domain. Cai et al. proposed a fault diagnosis method to detect the transient and intermittent faults [14], as well as the remaining useful life of structure systems [15]. Schneider [16] proposed a modeling framework based on DBN to study the reliability of deteriorating structural systems, Rebello [17] integrated the DBN with Hidden Markov Model to assess the functional reliability. In addition, the Petri net can be used to simulate the fault evolution considering the concurrency, sequence, and conflict relationships of various events. Zhou [18] presented a fault analysis approach based on Petri net for fault detection. Wu [19] proposed a quantitative safety analysis method by timed coloured Petri nets based on formal model. Lajmi [20] applied an interval fuzzy Petri net to satisfy the characteristics of uncertain systems. However, the above-mentioned methods confront the difficulty of expressional complexity in large systems such as avionics.

In addition, a few MBSA methods [21], [22] have introduced the conception of the state into component characterization. Specifically, the state machine [23], [24] is one of the powerful methods for failure propagation expression, where the state block is used to distinguish the normal conditions from abnormal conditions, and the failure is presented by an event to cause the state transition. For instance, AltaRica [25], [26], Simulink [27], [28] based methods have intensified the combination of the physical component behaviors and system configurations, facilitating the FTA and FMEA.

However, the mentioned state-based cascading failure analysis methods are largely focused on the interactions between concrete physical components, which limits comprehensiveness and flexibility of the cascading failure analysis, attributed to neglecting the interactions between various operational processes. With the complexity increase and functional integration of the avionics design, the operational processes have obtained increasingly significant roles in cascading failure analysis, especially in the early phase when the component details are still obscure.

Therefore, in order to realize the accurate and integrated evaluation of cascading failure towards the functionally integrated avionics, this paper proposes an operational process state-orientated safety analysis method considering cascading failures. The proposed method performs dynamic and dimensional safety analysis using the system design language SysML [29] through further integration of design mechanisms and cascading failure propagation process, contributing to the overall and accurate safety status evaluation and design mechanism optimization in the early development stage.

The rest of the paper is organized as follows. Section II introduces the system operational model framework of the proposed method, as well as the expressive means of functional integration mechanisms and safety handling mechanisms. Section III presents the dynamic analysis method of cascading failure effect and root cause, as well as the visualization method of the failure propagation path. Section IV analyzes the cascading failure propagation relationships in several failure scenarios, and conducts the comparison tests with the existing MBSA methods, by using the aircraft integrated surveillance system as a research object. Section V summarizes the paper.

II. OPERATIONAL PROCESS MODEL ESTABLISHMENT

A. STATE MACHINE CONCEPT

A system can be characterized by states as a combination of conditions and behaviors. When the system properties are described as the response to the external conditions, a state machine diagram denoted as \( M \) can be expressed as [30]:

\[
M = \left( \sum_i, S, S_0, O, T, \Delta, \partial \right)
\]

where

1) \( \sum_i = (X_1, X_2, \cdots, X_n) \) contains the various input parameters.
2) \( S \) indicates the current state collection.
3) \( S_0 \) denotes the initial state collection.
4) \( O \) describes the state machine output parameters, which are obtained by the combined function of input parameters and the states, and serves as the input parameters of other state machines.
5) \( T = (\cap TR, \cap GR, \cap EF) \) refers to the transition factor specification, where \( TR \) stands for the trigger that leads to the transition, \( GR \) describes the guard that constrains the transition, and \( EF \) indicates the effect to be executed during the transition. These factors can be assembled in the grammar of the state machine in the following way [31].

\[
\langle TR \rangle \langle GR \rangle / \langle EF \rangle
\]

6) \( \Delta (x, s, t) \) reflects the state transition function

\[
\Delta : S \times \sum_T \times T \rightarrow S
\]

7) \( \partial (x, s) \) appears as the output function of the state

\[
\partial : S \times \sum \rightarrow O
\]

Apart from the direct transition relationship between adjacent states, there are also indirect cascading relationships between states belonging to different process models.

**Definition 1**: Cascading relationship means that the abnormal state of one or a few system parts can lead to the failure of the other system parts.

Specifically, the cascading relationships between non-adjacent states can be divided into two categories as shown in Fig. 1.

1) Specific state transition in one process model induces multiple transitions in different processes (PROC) simultaneously:

\[
EF_0 \rightarrow \{ TR_1, TR_2, \cdots, TR_n \} TR_i \in PROC_i
\]
2) Specific state transition is accompanied by cascading transition of subsequent states in adjacent processes:

$$EF_0 \rightarrow TR_1 \rightarrow EF_1 \rightarrow \cdots TR_n \rightarrow EF_n \ (TR_i, EF_i) \in PROC_i$$

(6)

In order to express the cascading relationships between the states belonging to different process models, this paper utilizes the event broadcast mechanism to transmit the effect of state transition to the other objects in the BROADCAST_EVENT grammar, which is embedded in the effect statement of the transition. For instance, the event “FUNC1LOSS” can be transmitted to all the processes in the environment through the following semantics:

%BROADCAST_EVENT(“FUNC1LOSS”)% (7)

The predominance of the mechanism lies in performing real-time propagation among multiple models within the operational process framework. By further updating and synchronization of each model, the global association of various operational logics can be obtained.

B. CASCADING FAILURE ANALYSIS FRAMEWORK

The structure of the cascading failure orientated safety analysis framework is illustrated in Fig. 2. As presented in Fig.2, this framework mainly involves three processes, which are operational process state modeling, state relationships extraction, as well as cascading failure analysis and visualization.

Launched with the system operational model establishment, the logical cross-links among various functional processes are constructed under the functional logic framework, as well as the operational state connection under the various functional integration mechanisms and safety handling mechanisms. Then the state connections are extracted by the matrix assembly, which is followed by the analysis and visualization of cascading failure propagation by the propagation searching algorithms of cascading failure effect and cause in section III.

C. OPERATIONAL PROCESS STATE MODELING

1) OPERATIONAL PROCESS MODELING

According to the system operational modes, a hierarchical operational model framework is designed, which consists of the functional model layer, process model layer, behavior model layer, and physical model layer.

**Definition 2**: Function presents the goals that a system is expected to achieve.

**Definition 3**: Process involves the measures connected in the form of chains, to be taken to perform the function. The specific executor of a process is exempt from restriction, and it is implemented by multiple behavior combinations.

**Definition 4**: Behavior is the specific action bound to the physical components, which is capable of being reused by different processes and functions.

**Definition 5**: Physical component specifically refers to the physical reliability status of the component that conducts the behavior.

**Definition 6**: Failure mode reflects one of the various failure modes of a component, resulting in the change of component state.

In the system design language SysML, elements of each layer are represented by the Block Definition Diagram (BDD). Among them, functions and processes represent abstract concepts with no practical executor attached, while behaviors and physical components denote objective physical concepts. In order to ensure the independence of the process logic chains, the granularity of the behavior model is abstracted to the physical components with clear directivity.

The entire framework centralizes the process interaction structure. By bridging functions upward and mapping physical components downward, the reuse and flexible combination of behaviors can be used to describe various integration mechanisms and safety mechanisms of the system. Compared to the direct coupling between physical components, the loose coupling between behavioral states provides greater freedom for cascading failure propagation, and serves as a basis of cascading failure causation analysis from a global perspective.

2) INTEGRATION MECHANISM AND SAFETY MECHANISM MODELING

Once the operational process framework has been established, the operational states of each process can be represented by state diagrams, focusing on the integration mechanism and safety mechanism of a system.

In the context of the functionally complex system, integration mechanisms include the sharing and reuse of resources, energy, physical components, input parameters, and so forth,
illustrated within various processes. In order to express the joint function of the above-mentioned integration mechanism, all the mechanisms are disassembled and constrained by the substates of composite state relationships in the state diagram of each process model, with a compartment assigned for each dimension of integration, as depicted in Fig. 3. The nested sub-states in each compartment are used to determine whether the process status is normal or not. Among the above-mentioned integration factors, if one of those mechanisms violate the normal state boundary, the overall state of the process will be implicated accordingly.

As for the safety mechanism, the mechanisms can be split into two dimensions, namely, failure prevention mechanism and failure mitigation mechanism. Particularly, the commonly used mechanisms such as the system operational status monitoring mechanism, system redundant switching mechanism, and multiple input/output parameter voting mechanism can be modeled utilizing the composite states that are synchronized by various operational processes, or further depicted by an independent state block.

**D. OPERATIONAL STATE RELATIONSHIP ESTABLISHMENT**

Based on the system operational process model and the operational state model, the operational process interaction relationships and the state transition relationships are extracted by transition matrix assembly. The functional logic matrix indicates the logical relationship between the function modules, process models, and behavior models. In comparison, the state transition matrix reveals the transition relationships between various states in a state diagram, which is regarded as an attachment of the process model.

The state transition matrix shown in Fig. 4 shows the current state in the vertical direction and the updated state in the horizontal direction. The transferability between any two states is indicated by the solid line. Simultaneously, the transition relationship contains the trigger and effect semantics, which are above the line and below the line, respectively, representing the implicit connection relationship between the operational states. In addition, the global broadcast syntax structure of certain events is recorded through the effect event in the transitions.

Furthermore, when the process status is shifted from normal to abnormal due to one of the functional integration factors mentioned above, the state transition process will be broadcasted to all the process models in the global scope through the event. The implicit state connections further trigger cascading failures in other processes, where the output parameters of the upstream process are taken as an input, so global propagation of the effects along the state transition chains is achieved.

In the mathematical expression of the state transition relationships, \( \text{tmp} \) denotes the relation function of states, and it is expressed as:

\[
\text{tmp}(x_1, \ldots, x_m, y_1, \ldots, y_n) = \prod_{i=1}^{n} (y_i \equiv \delta_i (x_1, \ldots, x_m, w_1, \ldots, w_k))
\]

where

1) \( x \) depicts the current state collection, \( y \) describes the state collection to be updated.
2) \( W = (w_1, w_2, \ldots, w_k) \) represents the collection of applicable connections within the state matrix.
3) \( \delta_i (\cdot) \) is the transferability judgment of state.

Accordingly, the original state transition matrix \( \text{tmpG}^{(0)} \) can be defined as follows:

\[
\text{tmpG}^{(0)}[i][j] = \begin{cases} 
1 & \text{if } i = j \cup ((i, j) \in W \cap t \in T) \\
0 & \text{if } i \neq j \cup (i, j) \notin W \cup t \notin T
\end{cases}
\]

where \( t \) represents the current transition condition, and \( T \) is the constraints of applicable transition. Then the matrix \( \text{tmpG}^{(k)} \) of the order of \( k \) is calculated by:

\[
\text{tmpG}^{(k)}[i][j] = \text{tmpG}^{(0)}[i][j] \lor \left( \text{tmpG}^{(k-1)}[i][j] \land \text{tmpG}^{(k-1)}[j][i] \right)
\]

Moreover, the reachability matrix \( \text{rechG} \) is expressed as follows:

\[
\text{rechG} = G + G^2 + G^3 + \cdots G^m
\]
III. CASCADING FAILURE PROPAGATION ANALYSIS METHODS

After the operational process interaction relationship and state transition relationship are extracted as transition matrices, all system models, as well as the relationships, are retained in a database and extracted into the C# dataset as the original input for further cascading failure analysis of the effect and rational factor.

A. CASCADING FAILURE EFFECT ANALYSIS METHOD

The cascading failure effect propagation searching algorithm is designed considering the implicit relationships between state transition matrix sets. The algorithm is premised on the definition of the Discrete Event Dynamic System (DEDS) [32], which develops an expended three-level nesting on the definition of the Discrete Event Dynamic System state transition matrix sets. The algorithm is designed considering the implicit relationships between the original input for further cascading failure analysis of the effect and rational factor.

1) ALGORITHM LOGICAL IMPLEMENTATION

With regard to the logical implementation of the cascading failure effect propagation searching algorithm, the algorithm (ALG) is constituted by the definition of the following elements:

\[ \text{ALG} = \{\alpha, S, D_\alpha(S), DEQ, FPPROC\} \] (12)

where

1) \( \alpha \) represents the searching object.
2) \( S \) indicates the state of the object.
3) \( D_\alpha(S) \) denotes the activated states of the searching object.
4) \( DEQ \) refers to the dynamic event queue to reserve and update the cascading events (E) at each searching step, and it is expressed as:

\[ DEQ_x = \{E_x, step_x\} \] (13)

5) \( FPPROC \) denotes the failure propagation process based on the activated state and event of a certain step, and it is expressed as:

\[ FPPROC(S_\alpha, E_x, step_x) = \{D_\alpha(S), E_{x+1}\} \] (14)

Driven by the failure event in the dynamic event queue (DEQ) at a certain step, the current activated states of each process are combined with the existing cascading event \( (E_x) \) to determine the following propagating path illustrated by the cascading event \( (E_{x+1}) \) and estimate the states to be activated of each process \( (D_\alpha(S)) \).

The core of the cascading failure effect analysis method is the global propagation of the failure, which relies on the combination of jump search logic of the event and the state synchronization in real-time, as explained in the following, respectively.

1) According to the jump search logic, every effect event \( (e) \) induced the reactions \( (FPPROC(S, E_x, step_x)) \) of the state \( (S) \) with the event assembly \( (E_x) \) at \( step_x \) will constitute the cascading event assembly \( (E_{x+1}) \) at \( step_{x+1} \) in (15). After all the cascading events have been collected, the event assembly will be fed back to the dynamic event queue \( (DEQ_{x+1}) \) in (16), which will be broadcasted in further through the event broadcast semantics, and assembled with the other effect events and the current states of each process to conduct the continuous extension of the cascading effect, thus realizing the effect coupling between the global processes.

\[ E_{x+1} = \{e^1_{x+1}, e^2_{x+1}, \ldots, e^k_{x+1}\} \in FPPROC(S, E_x, step_x) \] (15)

\[ DEQ_{x+1} \ni \forall e \in E_x : FPPROC(S, E_x, step_x) \text{ has completed} \] (16)

2) With respect to the state synchronization, the searching step is compartmentalized according to the derived event collection at the previous step, rendering each event to be traversed for a complete interaction with the current states. Under the premise that all the possible cascading relationships at the current step have been coupled and deduced, the current activated conditions of each state will be updated collectively and recorded according to \( ACTIVATABLE() \) function in (17). The certain state will remain activated either at least one of the activation conditions is satisfied at the current step, or no activation conditions is violated if the state has already been activated previously, which serves as the input for the searching process of the next searching step.

\[ ACTIVATABLE(S_\alpha(step_x)) = \begin{cases} \exists e \in E_x : DEQ_x \nsubseteq \emptyset \cap FPPROC(S, E_x, step_x) \subset D_\alpha(S) \\ S_\alpha(step_{x-1}) = \text{activated} \land \neg e \in E_x : FPPROC(S, E_x, step_x) \nsubseteq D_\alpha(S) \end{cases} \] (17)

2) ALGORITHM CODE FRAMEWORK

With regard to the code execution of the cascading failure effect propagation searching algorithm, the cascading failure effect propagation searching function possesses a nested three-layer sub-function structure, where the three subfunctions perform the time control, update activated states, and perform specific searches from outside to inside layer respectively. The outermost subfunction (OutmostSearch()) initializes the activation state of all the state nodes within each process. Then, the dynamic event queue at each step is automatically traversed as long as the derived cascading event persists, searching for the state transition relationships that may be affected by the middle-layer subfunction (Middle-Search()). The middle-layer subfunction is responsible for updating the activation status of each process state according to the search results at a certain step, as well as for calling the innermost subfunction (InnermostSearch()) to perform the detailed search. The innermost subfunction conducts a series of recursive searches launching from the specific state passed by the middle subfunction. The subsequent node of a state, the nested trigger, and the effect of the transition are analyzed to determine the extensibility of the search. To be specific, first,
The downward adjacency state is used to measure and identify the searchable direction, and then the trigger and effect are combined with the dynamic event queue to determine the jump relationship and termination of subsequent search paths.

More specifically, the transition trigger will be matched with the event set of the dynamic event queue at the current step. In case there is no matching event, the current state node will be set as the endpoint of the search at this step. In contrast, the innermost subfunction is further called to conduct recursive searches. Besides, the event broadcast syntax structure contained in the effect statement is parsed by the trigger-effect couple mechanism, with the included event inserted into the dynamic event queue for the subsequent cascading failure search of the next step.

After all the downstream branches of the selected state have been searched, all the identified nodes in the propagation paths are combined with the current searching node and further passed to the uncompleted upper layer search, with the failure propagation chain updated on a continuous basis during the searching process.

As the searching process is based on the parallel function of multiple operational processes, all the state activations are globally shared and reacted to the failure in real-time regardless of positional relationship and centrality, then the failure propagation will be extended in a distributed manner according to the objective of different process chains, thus making the method adept at reflecting the combined impact of a failure event on multiple functions in the context of interactions of multiple processes.

### B. CASCADING FAILURE CAUSE ANALYSIS METHOD

The cascading failure cause propagation searching process consists of three steps. First, the cause of the appointed top failure state is searched to construct the cascading failure cause propagation chains. Then, the chains are parsed into the cascading failure causation tree structure. Finally, the root causes and minimum cut sets are obtained according to the tree structure.

1) CASCADING FAILURE CAUSE ANALYSIS ALGORITHM

The first step is accomplished by the cascading failure cause propagation searching algorithm, which analyzes all the causes that can lead to the specific unexpected effect. Starting from the top unexpected event, the reverse searching process is conducted in line with the upstream effect events corresponding to the trigger event recursively, until all of the potential root causes are identified. Then, the reverse failure propagation chains are generated, and the fault propagation process is further visualized.

Similar to the cascading failure effect propagation searching algorithm, the cascading failure cause propagation searching algorithm is composed by three sub-functions. The outermost one (OutermostSearch()) is a top-level calling function that appoints a specific top-level failure as a starting point according to the user input. Then the innermost subfunction (InnermostSearch()) is called to generate a set of root failure propagation chains. The innermost function further extracts all the upstream states of the starting point passed by the outermost subfunction before analyzing the trigger in the state transition between the searched node and its upstream

| TABLE 1. Code framework of cascading failure effect propagation searching algorithm. |
| --- |
| 1. //the activated conditions of state at step |
| 2. activated(state, step)==true/false |
| 3. //dynamic event queue, reserve events at every step |
| 4. deq=[<event, step>] |
| 5. //reserve the couples of trigger-effect relationships |
| 6. couple<<effect, transition, trigger, downstream>> |
| 7. //record the current and updated state of the transition |
| 8. transition StateFrom, StateTo |
| 9. // conduct search automation and time control |
| 10. Function OutermostSearch() |
| 11. State activated condition initialization |
| 12. set the certain bottom failure as startevent, set x=0 |
| 13. while deq(step+1) != empty then |
| 14. for each event in deq(step+1) |
| 15. find the activated transition of each process containing the event |
| 16. conduct MiddleSearch(process, transition, state_from) |
| 17. end for |
| 18. x++ |
| 19. end while |
| 20. end Function |
| 21. //update activated states |
| 22. Function MiddleSearch(process, step) |
| 23. for each state in process |
| 24. if activated(state, step) then |
| 25. Conduct InnermostSearch(state, step) |
| 26. end if |
| 27. update activated state by searching results of step |
| 28. end for |
| 29. Return propagation chains |
| 30. end Function |
| 31. //perform specific searches |
| 32. Function InnermostSearch(node, step) |
| 33. for each node.state downstream |
| 34. if transition(node, node.state downstream) has trigger but does not match the event collection of deq(step) then |
| 35. set the node as the terminal state of the search at step |
| 36. end if |
| 37. else |
| 38. conduct recursive InnermostSearch(node, state downstream, step) |
| 39. end else |
| 40. extract the event from effect, add to the deq(step+1) |
| 41. end for |
| 42. add the node to the head of each propagation chain |
| 43. return propagation chains |
| 44. end Function |
state node in turn. If a trigger exists, a further recursive search is conducted for all the upstream effects of the trigger.

During the searching process, if the current searched node shows no upstream state, or there is no matched upstream effect on the premise that the node is a trigger, the state/trigger is treated as the search endpoint, i.e., the root cause of a failure. Moreover, the search results are integrated into the upper layer subfunctions by failure reverse propagation chains, which include the root failure node and failure propagation nodes. Each node in the chain is arranged according to the grammatical structure, which can be further analyzed to generate the cascading failure causation tree structure.

Meanwhile, there are remaining state transition branches that could not be traversed attributing to the jump midway action of the trigger-effect cascading relationship. Therefore, the forward causes of the corresponding state of the trigger are further searched by the middle-layer subfunction (MiddleSearch()), and integrated with the existing search results, thus obtaining a complete chain collection that covers all causative events and failure propagation paths, as aggregated by the outermost subfunction.

2) CASCADING FAILURE REVERSE PROPAGATION CHAIN GENERATION

As for the cascading failure propagation chains of the causes for the top failure, all of the nodes checked during the search are categorized as either a state or a trigger event, with the propagation path recorded in the order of node arrangement. The relationship between adjacent nodes includes three types of grammatical structures: state-state, state-trigger, and trigger-state, which will be combined into the following two syntaxes.

**Syntax 1:** If two state nodes are interconnected, and the downstream node of the latter state node is treated as a state, it is considered that there is a direct transition between the two nodes.

**Syntax 2:** If two state nodes are interconnected, and the downstream node of the latter state node is categorized as a trigger event, then the latter state node is regarded as the current state, and the former state node is the state to be updated, regarded as the reaction of the current state on the trigger event, which is the above-mentioned the downstream node of the latter state node. Then these three nodes constitute an AND gate that demonstrates the state transition mechanism, which will be parsed into graphited form in the next subsection.

For instance, in Fig. 5, the two shown chains infer the failure propagation structure from the bottom cause to the top failure event; the trigger event is capitalized. In Fig. 5, state15 is the final failure state, while PRO2FAIL is the event that caused this state to be triggered by state14. In addition, the state that caused the event in another process is denoted as state13, and it can be further searched until the root causes are identified as TRIGG1 and PHYFAIL1 at the end of the two chains.

### TABLE 2. Code framework of cascading failure cause propagation searching algorithm.

| Line | Code |
|------|------|
| 1. | //conduct overall search and merge the search result |
| 2. | Function OutmostSearch() |
| 3. | set certain state as topfailurestate |
| 4. | <searchresult,unsearchedstate>=MiddleSearch(topfailurestate) |
| 5. | for unsearchedstate in the searchresult |
| 6. | searchresult2=InnermostSearch(unsearchedstate) |
| 7. | end for |
| 8. | end Function |
| 9. | |
| 10. | //record the unsearched state nodes of the search |
| 11. | Function MiddleSearch(topfailure) |
| 12. | propagationchains=InnermostSearch(topfailure) |
| 13. | record the unsearched state nodes due to the jump relations of the trigger |
| 14. | return propagationchains and unsearched nodes |
| 15. | end Function |
| 16. | //perform specific searches |
| 17. | Function InnermostSearch(node) |
| 18. | if exist(node, state_current) then |
| 19. | for each node, state_current |
| 20. | if trigger of transition(node, state_current, node) has upstream effect then |
| 21. | find the upstream transition through couple(effect_current, trigger, downstream) |
| 22. | conduct recursive InnermostSearch(transition, state_updated) |
| 23. | end if |
| 24. | else |
| 25. | assign the trigger as the root cause |
| 26. | end else |
| 27. | end if |
| 28. | end if |
| 29. | assign the node as root state |
| 30. | end else |
| 31. | add the node to the head of each propagation chain |
| 32. | return propagation chains |
| 33. | end Function |

3) CASCADING FAILURE CAUSE VISUALIZATION

Based on the failure reverse propagation chain, a causation tree structure is designed for the purpose of visualizing the cascading failure. The structure consists of the state node, the event node, the OR gate, and the AND gate. The state node and the event node denote the activated state and the cascading failure event presented in the failure propagation paths, respectively. Semantics of OR gate depicts that the state will be updated with the satisfaction of at least one of the condition branches, which is similar to the semantics of the OR gate in the fault tree. Semantics of AND gate is
FIGURE 5. The syntactic structure of cascading failure propagation chains.

specially designed to demonstrate the fault propagation logic. Herein, the updated state is defined as a consequence of the interaction between the trigger event and the current state, where the updated state set is a top event, the current state and the trigger event are the sub-branches of the AND gate. Furthermore, the upstream effect in another process model that activates the trigger event is traced in steps until the root cause is identified. The structure of cascading failure causation tree based on the trigger event is presented in Fig. 6.

FIGURE 6. Cascading failure causation tree structure.

There exist two means of the state updating process in the tree structure. First, the last state can be transited to the current state directly in case the activation of the last state. Second, the state transition can be constructed by a specific trigger event, which is regarded as the constraint for judging the feasibility of the transition. After that, the transition effect will be broadcasted to the remaining objects in the environment, resulting in the activation of other state transitions. Noted that each low-level effect may influence multiple relevant process models simultaneously, resulting in different high-level triggers, as shown in Fig. 6.

The proposed method and fault tree method considering the analysis of failure causes are compared in Table 3. The main advantage of the proposed method lies in the automated modeling of the cascading failure relationship between multiple processes, thus providing the intuitive and overall perception of the failure propagation paths as well as the root causes of the top failure. Moreover, each downstream effect appears only once in the model, which can be reused by multiple upstream processes, thus strengthening the uniqueness of elements and reducing the overall scale of the model compared to the fault tree.

| Advantage | Disadvantage |
|-----------|--------------|
| Fault tree | Widely applicable by the industry |
| Cascading failure causation tree | 1) Build the failure propagations among different processes automatically and globally. |
| | 2) Every event appears once, which depicts the root causes and model relations more clearly, effectively avoiding information duplication. |

TABLE 3. Comparisons between the cascading failure causation tree and fault tree.

In practice, the failure propagation graphical algorithm is designed to generate the cascading failure causation tree structure, with all the failure reverse propagation chains traversed. Particularly, the OR gate as well as the AND gate generation method developed to achieve the logical integration of the tree structure.

The failure propagation graphical algorithm is composed of the following steps. First, the graphical position of the first node of a chain to be searched is determined, and the location is inherited if the node overlaps with any other existing node in the tree structure; otherwise, a separate location is set to establish the causal structure. Next, each node of the chain is traversed and compared with the existing chains front-back. If a certain node falls into the state category, and the node along with its upstream nodes is consistent with the existing chain, while the downstream nodes are not, then the downstream nodes are parsed as a new branch of the OR gate of that node in the tree structure.

Second, it is determined whether an AND gate exists or not in the chain. The tree structure continues to extend downward until the “state + state + trigger event” structure is encountered. Following the afore-mentioned cascading failure causation tree structure, the first state is set as a top event of AND gate, and the second state and trigger event are set as the subordinate branches. Progressively, if the trigger
event contains an upstream effect event, the location of the updated state in the transition containing the upstream effect is appointed as the starting point of a new tree branch, and the further search is conducted until it encounters specific trigger event with no upstream element attached, which is the underlying root cause of the top failure.

Besides, the cascading failure mapping relationship and the root cause failure are labeled in the structure. On the one hand, the mapping relationship is obtained through the linkage from the upstream effect event to the downstream trigger event in two independent failure causation tree structures, which belong to different processes, to create a complete failure propagation path. On the other hand, the root cause failure is represented as the end node of the tree structure, which is presented in the form of the trigger event. Furthermore, the root cause failures are combined to generate the minimum cut sets.

For the failure reverse propagation chains presented in Fig. 5, the corresponding cascading failure causation trees are presented in Fig. 7, where the TRIGG1 event of PROCESS1 and the PHY2FAIL event of PROCESS0 denote the root failures, and they are highlighted by the orange color, which will cause the top-level failure state15 of the PROCESS3.

**FIGURE 7.** Cascading failure causation tree of the propagation chains presented in Fig.5.

4) CUT SETS GENERALIZATION

A minimum cut set search algorithm is designed to search for the minimum cut sets that cause failure events using the reverse failure propagation chain and causation tree. The searching algorithm of the minimum cut sets can be divided into two phases. First, all the root cause failures of the reverse cause chain are traversed and filtered. For every root cause to be searched, a Cartesian product with the existing minimum cut set is carried out to obtain all the cause event combinations. Second, the minimum cut sets covering all the root cause failures with the unique combination are obtained by removing duplicate events, out-of-order queue sorting, and excluding derived cut sets.

Compared with the cut sets generated from the fault tree by the static means, the minimum cut sets generated by the proposed method contain the state of nodes, which can illustrate the current situations with the dynamic behavior, as well as the cross influence of multiple elements, thus presenting more details for the top failure occurrence.

**IV. CASE STUDY OF CASCADING FAILURE CAUSALITY**

This paper selects the aircraft integrated surveillance system (ISS), which is a typical avionics system, to verify the proficiency of the proposed method in various integration mechanism-oriented scenarios. The integrated surveillance system plays an essential role in the safe operation of an aircraft because of its capability to monitor traffic, terrain, and weather conditions around the aircraft, and transmit the environmental conditions to the pilot. Moreover, the inherent fault of the system is displayed by the cockpit alert system.

By taking the traffic surveillance and meteorological surveillance functions referencing the A380 [33] as a research object, this paper demonstrates the cascading effect of the underlying events on the functional safety status of the ISS in typical abnormal scenarios related to the power supply, physical components, and safety mechanisms. In addition, the root causes and minimum cut sets that cause the top-level failure event were estimated.

In the verification process, the Enterprise Architect [34], which is a model-based system engineering tool, was utilized to construct the system model of the ISS. SQL Server was used to store the data, and C# programing language was adopted to develop the algorithms for cascading failure search and visualization.

**A. OPERATIONAL PROCESS MODEL CONSTRUCTION IN SysML**

The critical models in the simulation environment, constructed using the block definition diagram demonstrated in Fig. 8, were introduced in the following, ranging from the functional model to the physical model.

As for the functional model, two functions were included in the simulation, the traffic surveillance function and the meteorological surveillance function, of which the former one was the trunk function, while the latter one was utilized to represent the comprehensive impact of the fault event on the two functions.

In the model construction process, the main logic of the traffic surveillance function involved the acquisition of input parameters from the external systems, the integration and transmission of the input information to the calculation
TABLE 4. Major models of the traffic surveillance function in the simulation environment.

| Model Name              | Model Description                                                                 | Model Type         |
|-------------------------|----------------------------------------------------------------------------------|--------------------|
| Proc_top_traffic_surv   | Realize the surveillance of air traffic threats                                   | Top level function |
| Proc_access_altitude_info | Access altitude parameter                                                          | Trunk process      |
| Proc_collect_speed_info  | Access speed parameter                                                            | Trunk process      |
| Proc_transfer_info      | Fuse and transfer the input information to the threat calculation module          | Trunk process      |
| Proc_calcu_traffic      | Calculate traffic threats, serving as the main branch in the dual redundancy config | Trunk process      |
| Proc_calcu_traffic_threat_backup | Calculate traffic threats, serving as the backup branch in the dual redundancy config | Trunk process      |
| Proc_display_threat     | Display traffic information and threat                                             | Trunk process      |
| Phy_power_supply        | Supply electrical power to the specific process according to the power supply config | Supportive Process |
| Phy_power_supplyb       | Supply electrical power to the specific process according to the power supply config | Supportive Process |
| Phy_ISSPU1work         | The physical reliability status of ISSPU1, affected by different failure modes    | Supportive Process |
| Phy_ISSPU2work         | The physical reliability status of ISSPU2, affected by different failure modes    | Supportive Process |
| Phy_NDwork             | The physical reliability status of cockpit Navigation Display panel, affected by different failure modes | Supportive Process |

FIGURE 8. Operational process models of the traffic surveillance function.

As for the physical components and resources, the realization of the traffic surveillance function mainly involved the support of the Traffic Collision Avoidance System Module of Integrated Surveillance System Process Unit (ISSPU TCAS Module, with dual ISSPU configuration), ISSPU I/O module, ISSPU power module, and cockpit navigation display (ND) panels. The ISSPU TCAS module performed the threat calculation, the ISSPU I/O module conducted the information transmission, while the power module provided electricity to other modules according to the configuration of power supply. All the information was displayed on the ND panels. The physical components were organized flexibly to perform the relevant process and were configured in the blocks with the prefix of “Phy.” The traffic surveillance-related models used in the simulation are given in Table 4.

In addition, for assessing the coupling effect of the bottom failure on different functions, the operational models of the meteorological surveillance function rotated around the coupled processes with traffic surveillance function were modeled. For example, the meteorological threat calculation process was implemented by ISSPU1. The models related to the meteorological threat calculation are shown in Table 5.

A state machine model was designed inside each module based on the operational logic, as mentioned above. Some of the typical state models concerning the input parameter transmission process, and traffic threat calculation process, including the main branch and the backup branch, are presented in Fig. 9.

B. SIMULATION SCENARIO SETTINGS

Based on the integration mechanism of the integrated surveillance system, typical scenarios are selected from the perspective of potential common mode fault for verification purpose. The selected simulation verification scenarios include the following aspects:

1) Physical component failure and safety mechanism failure: Measure the cascading effect of the failure of physical components ISSPU along with the redundancy switching function.

2) Input parameter error: Assess the effect of input altitude error related to both traffic surveillance function and meteorological surveillance function.
TABLE 5. Major models of the meteorological surveillance function in the simulation environment.

| Model Name                     | Model Description                                           | Model Type     |
|-------------------------------|-------------------------------------------------------------|----------------|
| Func_top_weather_surv         | Realize surveillance of weather threats                      | Top level function |
| Proc_access_altitude_info     | Access altitude information                                  | Trunk process   |
| Proc_collect_speed_info       | Access speed information                                     | Trunk process   |
| Proc_transfer_info            | Fuse and transfer the input information to the calculation module | Trunk process   |
| Proc_calcul_windshield_threat | Calculate weather threats                                    | Trunk process   |
| Phy_power_supply              | Supply electrical power to the specific process according to the power supply configuration | Supportive process |
| Phy_ISSPU1work                | The physical reliability status of ISSPU1, affected by different failure modes | Supportive process |

3) Power supply failure in different configurations: Measure the impact of power supply failure on various processes and functions under different power supply configurations of the ISS system.

4) Failure of the top-level traffic surveillance function: Identify the root causes of the function failure.

C. CASCADING FAILURE EFFECT ANALYSIS RESULTS

1) SCENARIO 1: ISSPU1 PHYSICAL FAIL

Scenario 1 simulated the combined effect caused by the ISSPU1 physical failure and the redundancy switching mechanism. The search result, including the failure propagation paths, dynamic event queue status, state activation conditions, cascading event list, and cascading failure effect range, are shown in Fig. 10. As for the failure propagation process demonstrated in Table 6, the cascading failure of the ISSPU1 INNERFAIL (herein referred to ISSPU1 TCAS failed) included ISSPU1FAIL, which further degraded the traffic threat calculation process (Proc_calculate_traffic_threat_main degraded) under the premise that the ISSPU1 state monitor was normal. As ISSPU2 was enabled normally and performed the backup traffic threat calculation process (Proc_calculate_traffic_threat_backup operate), the top-level traffic surveillance function operated and completed normally.

The subsequent cascading events triggered by the failure of the ISSPU1, as well as the scope of the failure effect are given in Table 7.

When the ISSPU1 monitor could not detect the abnormality of the ISSPU1 component, the following ISSPU2 could not be activated, thus the current and downstream processes were affected, which resulted in the failure of traffic surveillance function, as given in Table 8.

2) SCENARIO 2: INPUT PARAMETER ERROR

Scenario 2 simulated the impact of the altitude parameter error which was required to calculate traffic and weather threats. The simulation results are presented in Table 9.

Since the altitude parameter represented the common input information of the traffic and the meteorological surveillance processes, the injected error (ALTITUDEINNERFAIL) led to the error in the altitude input information accessment process (Pro_access_altitude_info failed) at the first step. Then the input parameter could not be delivered at the second step (Pro_transfer_infoin failed), so the downstream traffic threat calculation process shifted to the degraded state (Pro_calcu_traffic_threat_main degraded) at the third step. Since the MONITOR detected the abnormality of ISSPU1 successfully, the cold reserved traffic threat calculation process was enabled. Nevertheless, as the ISSPU1 and ISSPU2 utilized the same altitude parameter as the input, which hindered the alternative process from being completed (Pro_calcu_traffic_threat_backup failed), thus resulting in the error of the subsequent process (Pro_display_threat failed) at the fifth step, and eventually led to the failure of the top-level traffic surveillance function (Func_top_traffic_surv failed).
With regard to the meteorological surveillance function, which utilized the same altitude parameter as the traffic surveillance function for the meteorological threat calculation, the meteorological threat calculation process could not be completed (Proc\_calc\_windshear\_threat failed) at the third step, resulting in the failure of the top-level meteorological surveillance function (Func\_top\_weather\_surv failed). The subsequent cascading events triggered by the altitude information error as well as the affected processes are presented in Fig. 10.

**FIGURE 10.** Simulation results of the cascading failure effect of the ISSPU1 physical failure.

**TABLE 6.** Failure propagation path of the ISSPU1 physical failure.

| Step | Process state                                                                 | Failure Propagation                                                                 |
|------|-------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| 1    | Inner ISSPU1 failed physically                                                |                                                                                     |
| 2    | The traffic threat calculation process (Proc\_calculate\_traffic\_threat\_main) was degraded |                                                                                     |
| 3    | ISSPU2 was enabled and conducted the traffic threat calculation process (Proc\_calculate\_traffic\_threat\_backup) |                                                                                     |
3) SCENARIO 3: POWER SUPPLY CONFIGURATION
Scenario 3 simulated the impact of electric power failure under four different configurations, performing the comparative analysis of the results, as represented by radar charts.

The power supply mechanism can be set as follows:
(1) ISSPU1, ISSPU2, and input information transmission process applied the same power supply path.
(2) ISSPU1 and ISSPU2 used the same power supply path, while another independent path was used for the input information transmission process.

(3) ISSPU1, input information transmission process used the same power supply path, while ISSPU2 used an independent power supply path.

(4) ISSPU1, ISSPU2, and input information transmission processes used separate power supply paths.

Among them, the input information transmission process was completed by ISSPU I/O module, while the power supply process was realized by the ISSPU power module. The search results of different configurations are listed in Table 11, demonstrating the different range of the failure effect.

Taking the first configuration as an example, if ISSPU1, ISSPU2, and the input information transmission process adopted the same power supply mechanism, power supply error was radiated to the input information transmission process, the traffic threat calculation process, and the meteorological threat calculation process simultaneously. Therefore, whether the monitor work normally or not, the top-level traffic and meteorological surveillance function could not be completed normally, as reflected by the fail state in the first radar chart.

As for the second configuration, the ISSPU1 failure led to the activation of the cold reserved ISSPU2. As the two branches utilized the same bus bar for power supply, the alternative branch could not perform the traffic surveillance function despite the rest function of ISSPU2 remained normal (such as input information transmission process), which led to the failure of top-level function eventually.

With regard to configuration 3, if the input transmission process and ISSPU1 adopted the same power supply path, the power supply failure triggered the same result as the first configuration, as shown in the third radar chart. Besides,
the separate power supply option of configuration 4 resulted in the minimum range of effect shown by the fourth radar chart, whereas confronted with the cost of the deployment.

D. CASCADING FAILURE CAUSE ANALYSIS RESULTS

As for the cascading failure cause analysis, the traffic surveillance function failure was exemplified to analyze the failure propagation chains and root causes of the top-level functional failure as scenario 4. The failure of the traffic surveillance function was attributed to the abnormalities of various required processes for the function implementation, and the root cause failure was traced backward according to the process interaction. The searching results included seven direct failure reverse propagation chains and 28 indirect failure reverse propagation chains, as shown in Fig. 11. The indirect failure reverse propagation chains were generated based on the failure cascading relationship between separate processes.

The cascading failure causation tree structure of the selected first 14 propagation chains is shown in Fig. 12. As shown in Fig. 12, MONITORFAIL and POWERINNERFAIL1 were the root events by expending the subbranches of the AND gate of Pd_failed (depicted by the red triangle). Therefore, MONITORFAIL could be combined with POWERINNERFAIL1 to prompt the top failure occurrence. In addition, as Pd_ExitPoint included an OR gate (depicted by a purple semicircle in Fig. 12), and its subordinate branch contained a cascading failure event (PRO3 FAIL, depicted by a green circle in Fig. 12), so the upstream search could be conducted using the cascading relationship launched to the P2_failed branch. By tracing the causal events by the OR gate branch, three events were obtained: SPEEDINNERFAIL, POWERFAIL, and ALTITUDEINNERFAIL, which could be combined with MONITORFAIL to trigger the top event. Therefore, the minimum cut sets were obtained by means of the afore-mentioned branches reasoning:

1) MONITORFAIL + POWERINNERFAIL1 (branch1).
2) MONITORFAIL + SPEEDINNERFAIL (branch2).
3) MONITORFAIL + POWERFAIL (branch3).
4) MONITORFAIL + ALTITUDEINNERFAIL (branch4).

The remaining minimum cut sets were further deduced by cycling chains. The final minimum cut set elements are shown in the list at the bottom left part of Fig. 11, and they can be divided into the four categories, as given in Table 12.

E. METHOD COMPARISON

In order to verify the feasibility of the proposed state-based operational process method (OPSBM), the results were compared to the results of the mainstream safety-based analysis methods: the HiP-HOPS and AltaRica methods.

As for the HiP-HOPS, first, the ISS model was established by the software application package “SafetyDesigner” of SimulationX [11]. Then, the top failure was specified, and the fault tree was constructed, following the FMEA was generated reversely. The AltaRica method was applied by
the OpenAltaRica software [25], with the components and the interactions constructed as AltaRica 3.0 models in the form of formula. The fault trees were constructed in an open probabilistic safety assessment (OPSA) format from the AltaRica 3.0 model.

When the model of the ISS system was established, the failure effect of the input failure of scenario 2, the failure causes of the traffic surveillance failure of scenario 4, as well as the computational costs of different methods were obtained and compared.

As results in Table 13 show, the failure ranges and the minimum cut sets of different methods were identical considering multiple scales of the model, which illustrated the equivalent accuracy of the proposed method towards the cascading failure effect analysis and the cause analysis, respectively. Therefore, the correctness of the proposed method has been validated inductively and deductively.

With respect to the computational cost, the proposed method took shorter modeling and searching time than the other two methods, with the attributing factors considering model establishment and failure analysis method further listed in Table 14.

Generally speaking, the computational superiority of the proposed method can be summarized from the two following aspects.

First, in terms of the modeling cost, the quantity of model connections in the proposed method was less than those of the HiP-HOPS and AltaRica methods, as shown in Table 13. The information transmission of the HiP-HOPS required the manual annotation of the failure reaction logic in each component block, as well as all the connections between the blocks, which made the modeling process complex. In addition, the failure propagation in the AltaRica method was based on “state(element1)-> variable(element1)-> variable(element2)” mechanism, following the semantics of transition and assertion, which meant the transmission of all the failure and variables must depend on the direct connection between models. In contrast, the proposed method was
TABLE 12. Minimum cut sets categorization of traffic surveillance function failure.

| Category | Minimum Cut set | Explanation |
|----------|-----------------|-------------|
| 1) The monitor was normal, but the common failure such as power supply or input parameter led to the joint failure of the two calculating branches directly. | POWERFAIL MONITOR NORM | Power supply failure under the premise of monitor normal |
| | SPEEDINNERFAIL MONITOR NORM | Speed parameter input error under the premise of monitor normal |
| | ALTITUDEINNERFAIL MONITOR NORM | Altitude parameter input error under the premise of monitor normal |
| | POWERINNERFAIL2 MONITOR NORM ISSPUINNERFAIL | ISSPU1 failed physically, ISSPU2 was enabled but did not operate due to the common power supply configuration |
| 2) The monitor was normal, the main branch failure of threat calculation process in the combination of the physical failure of ISSPU2 or the failure of power supply caused the overall function failed | POWERINNERFAIL2 MONITOR NORM POWERINNERFAIL RESOURCEFAIL | Power supply of the main threat calculation branch failed, ISSPU2 was enabled but did not operate due to the common power supply configuration |
| | ISSPUINNERFAIL MONITOR NORM ISSPUINNERFAIL | ISSPU1 failed physically, ISSPU2 was enabled but did not operate due to the ISSPU2 physical error |
| | ISSPUINNERFAIL MONITOR NORM RESOURCEFAIL | Resource distribution of the main threat calculation branch failed, ISSPU2 was enabled but did not operate due to the ISSPU2 inner error |
| | ISSPUINNERFAIL MONITOR NORM POWERINNERFAIL | Power supply of the main threat calculation branch failed, ISSPU2 was enabled but did not operate due to the ISSPU2 inner error |
| 3) The main branch failure of threat calculation process caused by the power supply, input parameter, physical component, or resources factor led to the threat calculation process could not be realized due to the monitor failure | MONITORFAIL ISSPUINNERFAIL | ISSPU1 failed physically, ISSPU2 was not enabled due to the monitor error |
| | MONITORFAIL RESOURCEFAIL | Resource distribution of the main threat calculation branch failed, ISSPU2 was not enabled due to the monitor error |
| | MONITORFAIL POWERFAIL | Power supply of the input transmission process failed, ISSPU2 was not enabled due to the monitor error |
| | MONITORFAIL ALTITUDEINNERFAIL | Altitude parameter input of the main threat calculation branch failed, ISSPU2 was not enabled due to the monitor error |
| | MONITORFAIL SPEEDINNERFAIL | Speed parameter input of the main threat calculation branch failed, ISSPU2 was not enabled due to the monitor error |
| 4) Single point of failure caused by the information display process resulted in the function failure | POWERINNERFAIL1 | Power supply of the main threat calculation branch failed, ISSPU2 was not enabled due to the monitor error |
| | NDINNERFAIL | Navigation display screen failed |

based on the “state(element1)- event- state(element2)” mechanism; namely, the events were first reserved in the dynamic event queue and then broadcasted globally, therefore all the states of different models could be coupled without establishing a direct connection, thus reducing the complexity of the modeling process. Besides, the modification and optimization of a design could be conducted more easily as the models were loosely coupled.

Second, in the failure propagation searching process, since one physical component could participate many processes utilizing different resources and information, therefore, the failure propagation searching based on a component was complex because all the process logic in one physical model was intertwined, so the search paths must be extended according to the concrete path in the physical architecture. The operational process state-based method could describe the failure propagation in a more accurate and flexible way, as each process was established independently and connected by the operational logic, so the failure propagated along the relevant downstream process model rather than traverse...
all the connected components, which streamlined the search volume and accelerated the search speed while avoiding the unnecessary reactions of irrelevant models. Consequently, the failure propagation paths based on the operational state could be calculated faster and depicted with fewer events and gates compared to other methods, as shown in Table 13.

In order to analyze the scalability of the proposed method, the tests considering the traffic surveillance function, the traffic surveillance function along with the weather surveillance function, as well as the weather, terrain, and traffic surveillance functions were conducted in sequence. The results are presented in Table 13, where it can be seen that the proposed

---

**TABLE 13. Results comparison of failure effect and cause analysis of different methods.**

| Method          | Time for modeling | Quantity of model | Quantity of connection<sup>a</sup> | Failure propagation analysis cost | Failure effect range | Fault tree generalization cost | Fault tree element | Quantity of fault cut set | Quantity of minimum cut set |
|-----------------|-------------------|------------------|------------------------------------|----------------------------------|----------------------|------------------------------|---------------------|--------------------------|---------------------------|
| Case 1<sup>a</sup> | HiP-HOPS[11]      | 1h               | 9                                  | 30                               | <10s                 | 1 function                  | <30s                | 7 OR gate                | Static, 8                 |
| AltaRica[25]    | 1.5h              | 5                | 40                                 | <45s                             | 1 function           | <80s                         | 10 OR gate            | 5 AND gate               | Dynamic, 10               |
| OPSBM           | 0.2h              | 12               | 19                                 | <5s                              | 5 processes          | 1 function                  | <5s                 | 3 OR gate                | Dynamic, 10               |
| Case 2<sup>b</sup> | HiP-HOPS         | 1.5h             | 18                                 | 43                               | <15s                 | 2 functions                 | <150s               | 10 OR gate               | Static, 14                |
| AltaRica        | 2h                | 8                | 59                                 | <70s                             | 2 functions          | 200s                        | 13 OR gate            | 24 OR gate               | Dynamic, 16               |
| OPSBM           | 0.4h              | 14               | 29                                 | <5s                              | 7 processes          | 17 AND gate                 | <5s                 | 4 OR gate                | Dynamic, 16               |
| Case 3<sup>c</sup> | HiP-HOPS         | 3h                | 20                                 | 67                               | <60s                 | 3 functions                 | <200s               | 18 OR gate               | Static, 19                |
| AltaRica        | 4h                | 12               | 84                                 | <220s                            | 8 processes          | 23 AND gate                 | <300s               | 33 OR gate               | Dynamic, 23               |
| OPSBM           | 0.5h              | 17               | 37                                 | <7s                              | 8 processes          | 49 AND gate                 | <15s                | 7 OR gate                | Dynamic, 23               |

<sup>a</sup> Case 1 focuses on the traffic surveillance function in the ISS system.  
<sup>b</sup> Case 2 focuses on the traffic surveillance function along with the meteorological surveillance function in the ISS system.  
<sup>c</sup> Case 3 focuses on the traffic, terrain and meteorological surveillance function assembly in the ISS system.  
<sup>d</sup> The connection in AltaRica denotes the annotations of assertion and transitions between the states, events and variables.

**TABLE 14. Comparisons of the three methods.**

| Method description method | HiP-HOPS | AltaRica | Operational process state-based method |
|---------------------------|----------|----------|---------------------------------------|
| Model description method  | Based on the physical component | Based on the physical component | Based on the operational process |
| Model construction method | Graphical and textual specifications | Textual specifications (formal language) | Graphical specifications |
| Failure analysis method   | Static, without the conception of state | Dynamic, state is introduced | Dynamic, and state is further depicted through the compositional state dimensionally |
| Failure propagation method| Transferred according to the manual annotations defined in each block | Transferred through the blocks with direct linkage | Transferred among all the blocks in the simulation environment |
| Propagation path depiction| Partial, displayed through fault trees | Partial, displayed through fault trees | Overall, displayed dynamically through the failure propagation path step by step, as well as the cascading failure causation tree |
| Failure effect depiction  | FMEA     | FMEA     | FMEA, as well as all the cascading events and failure effect range |
| Failure root cause depiction | Yes     | Yes     | Yes |
| Minimum cut set depiction | Static   | Dynamic, contain the event state | Dynamic, contain the event state |
method performance is compatible with the complexity of the existing functionally integrated avionics, as the proposed model achieves high searching accuracy at moderate modeling cost and calculation time. And it can be adopted to other industrial fields with a large scale and a long development period, such as aeronautics, nuclear energy, railway, and automobile field.


to verify the rationality of the system design and the effectiveness of the safety measures, which is conductive to both hazard identification and design optimization in the early stage of avionics development.

V. CONCLUSION
With the aim to solve the problem of cascading failure in the context of increasingly complex operational process of avionics, this paper proposes a cascading failure analysis method with the hierarchical operational process state modeling framework, considering both the functional integrated mechanism and the safety mechanism, and thus achieving the dynamic analysis of cascading effects of a certain underlying failure event, as well as identifying the root causes and minimum cut sets that can trigger a series of cascading failure.

The main contribution of this paper can be summarized as follows.

1) A state-orientated operational process modeling framework is proposed, which can express various functional integrated mechanisms and safety handling mechanisms more accurately than the existing physical component-based modeling frameworks, thus facilitating the discovery of the potential hazard in the early development stage.

2) The cascading failure propagation analysis algorithms are developed, that can achieve the dynamic and global evaluation of fault propagation without establishing a complex linkage or annotation between models in contrast to the existing model-based safety analysis methods, and thus elevates the efficiency and flexibility of modeling process and modification process.

3) The cascading failure causation tree structure is designed, which demonstrates the capability to determine the root failure and the cascading failure relationship in graphical means automatically and establish the failure relationships between various processes globally. The structure of the causation tree supplements the traditional FTA from the aspect of the manual analysis and partial demonstration while enhancing the information diversity and comprehensiveness from an overall perspective.

The proposed method realizes the dynamic and global analysis of safety state from the perspective of the operational process state, thus increasing the automation level and reliability of the safety analysis by combining the dynamic cascading failure analysis process with the design process.

A further expansion of quantitative analysis, as well as the uncertainty analysis will be the focuses of research in the future. This method can be applied to the safety assessment for the complex avionics with various integrated mechanisms, to verify the rationality of the system design and the effectiveness of the safety measures, which is conductive to both hazard identification and design optimization in the early stage of avionics development.

ABBREVIATIONS
The following abbreviations are used in this manuscript:

| Abbreviation | Description |
|--------------|-------------|
| FTA          | Fault Tree Analysis |
| FMEA         | Failure Modes and Effects Analysis |
| BDD          | Block Definition Diagram |
| DEDS         | Discrete Event Dynamic System |
| DEQ          | Dynamic Event Queue |
| DFS          | Depth First Search |
| HiP-HOPS     | Hierarchically Performed Hazard Origin and Propagation Studies |
| ISS          | Integrated Survelliance System |
| ISSPU        | Integrated Survellance System Process Unit |
| MBSA         | Model-Based Safety Analysis |
| OPSA         | Open Probabilistic Safety Assessment |
| OPSBM        | Operational Process State Based Method |
| SysML        | System Modeling Language |
| TCAS         | Traffic Collision Avoidance System |
| ND           | Navigation Display |
| MSC          | Minimum Cut Set |

REFERENCES
[1] S. Sierla, I. Tumer, N. Papakonstantinou, K. Koskinen, and D. Jensen, “Early integration of safety to the mechatronic system design process by the functional failure identification and propagation framework,” Mechatronics, vol. 22, no. 2, pp. 137–151, Mar. 2012.
[2] Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment, SAE International S-18 Committee, ARP4761, Warrendale, PA, USA, 1996.
[3] L. Chen, J. Jiao, and T. Zhao, “Review for model-based safety analysis of complex safety-critical system,” J. Syst. Eng. Electron., vol. 39, pp. 1287–1291, Jun. 2017.
[4] A. Joshi, S. P. Miller, M. Whalen, and M. P. E. Heimdahl, “A proposal for model-based safety analysis,” in Proc. 24th Digit. Avionics Syst. Conf., Washington, DC, USA, vol. 2, 2005, pp. 1–13.
[5] M. Bretschneider, H. J. Holberg, E. Böde, I. Brückner, T. Peikenkamp, and H. Spence, “Model-based safety analysis of a flap control system,” in Proc. 14th Annu. Int. Symp. Eng. (INCOSE), Toulouse, France, vol. 14, 2004, pp. 246–256.
[6] N. Jiang, G. Li, and B. Liu, “Model-based safety analyses of embedded system using stateflow,” in Proc. 11th Int. Conf. Rel., Maintainability Syst. (ICRMS), Hangzhou, China, Oct. 2016, pp. 1–6.
[7] P. Freeman, R. Pandita, N. Srivastava, and G. J. Balas, “Model-based and data-driven fault detection performance for a small UAV,” IEEE/ASME Trans. Mechatronics, vol. 18, no. 4, pp. 1300–1309, Aug. 2013.
[8] B. Khan, F. Khan, B. Veitch, and M. Yang, “An operational risk analysis tool to analyze marine transportation in arctic waters,” Rel. Eng. Syst. Saf., vol. 169, pp. 485–502, Jan. 2018.
[9] Z. Xie and P. Zhang, “Evaluative study of the culture construction of road traffic safety based on structural equation modeling,” in Proc. 5th Int. Conf. Transp. Inf. Syst. (ICTIS), Liverpool, U.K., Jul. 2019, pp. 1169–1173.
[10] P. Fenelon and J. A. McDermid, “An integrated tool set for software safety analysis,” J. Syst. Softw., vol. 21, no. 3, pp. 279–290, Jun. 1993.
[11] Y. Papadopoulos, M. Walker, D. Parker, E. Rüde, R. Hamann, A. Uhlig, U. Grätz, and R. Lien, “Engineering failure analysis and design optimization with HiP-HOPS,” Eng. Failure Anal., vol. 18, no. 2, pp. 590–608, Mar. 2011.
[12] S. Sharvia and Y. Papadopoulos, “Integrating model checking with HiP-HOPS in model-based safety analysis,” Rel. Eng. Syst. Saf., vol. 135, pp. 64–80, Mar. 2015.
[13] M. Wallace, “Modular architectural representation and analysis of fault propagation and transformation,” Electron. Notes Theor. Comput. Sci., vol. 141, no. 3, pp. 53–71, Dec. 2005.
[14] B. Cai, Y. Liu, and M. Xie, “A dynamic-Bayesian-network-based fault diagnosis methodology considering transient and intermittent faults,” IEEE Trans. Autom. Sci. Eng., vol. 14, no. 1, pp. 276–285, Jan. 2017.
[15] B. Cai, X. Shao, Y. Liu, X. Kong, H. Wang, H. Xu, and W. Ge, “Remaining useful life estimation of structure systems under the influence of multiple causes: Subsea pipelines as a case study,” IEEE Trans. Ind. Electron., vol. 67, no. 7, pp. 5737–5747, Jul. 2020.

[16] R. Schneider, S. Thoms, and D. Straub, “Reliability analysis and updating of deteriorating systems with subset simulation,” Struct. Saf., vol. 64, pp. 20–36, Jan. 2017.

[17] S. Rebello, H. Yu, and L. Ma, “An integrated approach for system functional reliability assessment using dynamic Bayesian network and hidden Markov model,” Rel. Eng. Syst. Saf., vol. 180, pp. 124–135, Dec. 2018.

[18] J. Zhou, “A fuzzy Petri-net approach for fault analysis considering factor influences,” IEEE Access, vol. 6, pp. 72229–72238, 2020.

[19] D. Wu and W. Zheng, “Formal model-based quantitative safety analysis using timed coloured Petri nets,” Rel. Eng. Syst. Saf., vol. 176, pp. 62–79, Aug. 2018.

[20] F. Lajmi, A. J. Talmoudi, and H. Dhouibi, “Fault diagnosis of uncertain systems based on interval fuzzy Petri net,” Stud. Inform. Control, vol. 26, no. 2, pp. 239–248, Jun. 2017.

[21] V. Kumar, L. K. Singh, A. K. Tripathi, and P. Singh, “Safety analysis of safety-critical systems using state-space models,” IEEE Softw., vol. 34, no. 4, pp. 38–47, 2017.

[22] R. Han, S. Wang, B. Liu, T. Zhao, and Z. Ye, “A novel model-based dynamic analysis method for state correlation with IMA fault recovery,” IEEE Access, vol. 6, pp. 22094–22107, 2018.

[23] P. Honig, R. Lunde, and F. Holzapfel, “Model based safety analysis with smartFlow,” Information, vol. 8, no. 1, pp. 7–22, 2017.

[24] B. Wang, T. Zhao, and J. Jiao, “Safety simulation technology based on finite state machine,” Beijing Hangkong Hangtian Daxue Xuebao, vol. 37, no. 4, pp. 428–438, Apr. 2011.

[25] H. Dong, Q. Gu, G. Wang, Z. Zhai, Y. Lu, and M. Wang, “Availability assessment of IMA system based on model-based safety analysis using AltaRica 3.0,” Processes, vol. 7, no. 2, pp. 117–131, 2019.

[26] A. Rauzy and C. Blieriot-Fabre, “Model-based safety assessment: Rational and trends,” in Proc. 10th France-Japan/ 8th Europe-Asia Cong. Mecatronics (MECATRONICS-Tokyo), Tokyo, Japan, Nov. 2014, pp. 1–10.

[27] Y. Dong, Z. Li, Y. Cheng, and H. Zhao, “A model DrivenTesting solution for embedded system with simulink/stateflow model,” in Proc. 2nd Int. Conf. Trustworthy Syst. Their Appl., Hualien, China, Jul. 2015, pp. 24–29.

[28] C. Bernardeschi, A. Domenici, and P. Masci, “A PVs-simulink integrated environment for model-based analysis of cyber-physical systems,” IEEE Trans. Softw. Eng., vol. 44, no. 6, pp. 512–533, Jun. 2018.

[29] S. Friedenthal, A. Moore, and R. Steiner, “OMG systems modeling language (OMG SysML) tutorial,” in Proc. 16th Annu. Int. Symp. Int. Council Syst. Eng. (INCOSE), Orlando, FL, USA, vol. 9, 2006, pp. 65–67.

[30] Y. Fan, J. Zhang, Q. Gong, and Y. Zhu, “Safety analysis for complex system based on the finite state machine theory,” in Proc. 9th Int. Conf. Rel., Maintainability Saf., Guiyang, China, Jun. 2011, pp. 594–598.

[31] L. Delligati, “State machine diagrams,” in SysML Distilled: A Brief Guide to the Systems Modeling Language, 6th ed. Upper Saddle River, NJ, USA: Addison-Wesley, 2014, pp. 155–175.

[32] J. M. Garrido, “Discrete-event simulation,” in Practical Process Simulation Using Object-Oriented Techniques and C++, Boston, MA, USA: Artech House, 1999, pp. 89–99.

[33] X. Gang, “Integrated aircraft environment surveillance system for large civil aircraft,” in Proc. IEEE/AIAA 30th Digit. Avionics Syst. Conf., Seattle, WA, USA, Oct. 2011, pp. 1B1–1B1–8.

[34] T. B. Ferreira, S. M. Stemplniuc, and J. Lisboa-Filho, “Geographical data modeling with the UML GeoProfile and MDA transformations on the enterprise architect tool,” in Proc. 9th Iberian Conf. Inf. Syst. Technol. (CISTI), Barcelona, Spain, Jun. 2014, pp. 1–6.