Hindawi
International Journal of Aerospace Engineering
Volume 2021, Article ID 8875872, 14 pages
https://doi.org/10.1155/2021/8875872

Research Article

Safety Assessment of the Reconfigurable Integrated Modular Avionics Based on STPA

Changxiao Zhao,[1] **Lei Dong**,[1] **Hao Li**,[2] **and Peng Wang**[3]

[1]*College of Airworthiness, Civil Aviation University of China, Tianjin 300300, China*
[2]*AVICAS Generic Technology CO.LTD, Yangzhou 225000, China*
[3]*Key Laboratory of Civil Aircraft Airworthiness Technology, CAAC, Tianjin 300300, China*

Correspondence should be addressed to Lei Dong; dlcauc@126.com

Received 9 August 2020; Revised 5 January 2021; Accepted 11 January 2021; Published 27 January 2021

Academic Editor: Zhiguang Song

The reconfiguration technology, which is the significant feature of the newly designed Integrated Modular Avionics (IMA) system, enables the transfer of avionics functions from the failed module to the residual normal module, thereby enhancing the robustness of the whole system. The basic target of the IMA reconfiguration is to ensure the safe flight and correct execution of the mission. To solve the problem of lack of effective management mechanism for the IMA system development and safety assessment, a safety analysis method based on STAMP/STPA and UPPAAL for IMA reconfiguration is proposed. The method focuses mainly on system characteristics and multiparty interactions. On the basis of this approach, some studies and analyses have been carried out. Firstly, the STAMP/STPA principle is studied and used to identify unsafe control actions in the reconfiguration process. Secondly, a formal model of IMA reconfiguration is developed using UPPAAL. Finally, the accessibility analysis of the formal model is used to analyze UCAs and the corresponding loss scenarios. The method enables a detailed description of the interactions between the components and a rigorous mathematical analysis of the system, thereby diluting the effect of human factors while ensuring the accuracy and reliability of the safety constraints.

1. Introduction

Integrated avionics, one of the three iconic technologies of the aircraft industry, is the "brain" and "nerve center" of the aircraft and a crucial system to ensure flight safety. Since 1970, avionics technology has gone through three stages of development [1]. In the 1970s, federated avionics was widely used on Boeing B737 and Airbus A320. In the federated avionics system, each subsystem is relatively independent, and there is little exchange of information among different subsystems. The enclosure of each equipment forms a natural fault propagation barrier, so that an internal failure of one subsystem is clearly distinguishable from that of other subsystems. With the growing demand for avionics, some weak points have been gradually exposed, such as low integration, expensive system upgrades, inadequate fault tolerance, a mass of spare resources, and maintenance difficulties. After the 1980s, Integrated Modular Avionics (IMA) was gradually developed and widely used in aircraft such as Airbus A380, Boeing B787, and COMAC C919. The IMA system is a distributed real-time shared computer network consisting of a set of flexible, reusable, and interoperable hardware and software. Each computing module can run multiple applications at different safety-critical levels simultaneously for multiple aircraft functions and separate each application based on a robust partitioning mechanism to ensure functional independence [2]. The core concept of IMA is hardware sharing, which can effectively cut down the cost, weight, volumes, and energy consumption of avionics systems. Currently, avionics technology is further evolving towards a new IMA system—Distributed Integrated Modular Avionics (DIMA) [3]. IMA reconfiguration, the significant technology of the next-generation DIMA system, not only effectively reduces hardware redundancy, but also greatly strengthens the system flexibility and the ability to cope with different missions and resource failures [4, 5]. The safety

requirements of the IMA reconfiguration are significant to ensure the flight safety and proper mission execution.

In the absence of a suitable management mechanism appropriate to IMA system development and safety assessment, it draws lessons from the management mechanism of traditional federated avionics. For example, SAE ARP4754A, "Guidelines for Development of Civil Aircraft and Systems," describes the full life-cycle process for the development of civil aircraft systems, and ARP4761, "Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne System and Equipment," defines aircraft and system level safety assessment process. The main methods used in ARP are traditional safety analysis methods, such as Fault Tree Analysis (FTA) and Failure Mode and Effect Analysis (FMEA), most of which are based on the event chain model, attributing the safety problem to reliability. It is believed that the reliability of the system components can ensure system safety. It finds the "root cause (component failure)" that directly leads to the accident based on the top-level events. On the one hand, the root cause is prevented, and related accidents are avoided. On the other hand, obstacles are added during event propagation to reduce the failure effects. The traditional event chain models oversimplify the causality and process of the accident, and many hazards are neglected, such as the nonlinear interactions among system components. The system theory model focuses on component constraints, which include constraints on and among components. Since hazards in IMA systems arise primarily due to interactions among components, system theory models are more appropriate than event models in the safety analysis of IMA systems. However, there are few studies applying system theory to the safety analysis of IMA systems. In this paper, an STPA-based analysis method is put forward to apply system theory to IMA safety analysis. Moreover, the STPA method is combined with the formal model detection method based on Time Automata to dilute the effect of human factors.

2. Literature Review

The new safety issues caused by the integration of aircraft avionics systems have attracted wide attention from aviation, space agencies, and scholars from various countries. Onera, Thales, and Airbus jointly implemented the SCARLETT project [6] and proposed an alternative avionics system. Great attention has been paid to the safety of the dynamic reconfiguration system, but a feasible safety assessment theory has not been reestablished. NASA's Langley Research Center proposed a solution which added an independent decision module to the IMA system to monitor sharing independence among the applications of avionics resources [7]; The thesis [8] proposed a method for analyzing the impact of interrupt correlation using the AADL model. The thesis [9] inverted the risk of redundant sequences in the cumulative transmission of the AFDX protocol; an analysis was carried out, and a method to mitigate this risk by limiting the transmission length was proposed. Some scholars have shown through research [10] that the traditional safety assessment methods based on the chain-of-events model considering only component failure are not applicable to IMA, which is a software-intensive system. A large number of potential hazards in IMA are caused by component interactions, rather than component failures.

Nancy Leveson raised System Theory Accident Model and Process (STAMP), in which the system can be described by hierarchical control based on adaptive feedback mechanism, and the failure in safety constraints is the main cause of accidents [11]. System Theory Process Analysis (STPA) is a hazard analysis method based on STAMP. It can be applied at any stage of the system's life cycle and has the same objectives as other hazard analysis techniques, such as identifying system unsafe control actions (UCAs) and analyzing the potential hazards of each control component, which is more suitable for complex systems. Nevertheless, it relies much heavily on the experience of analysts and is subject to the integrity as well as the objectivity of the result.

STPA has been successfully applied to the analysis of multiple safety and demanding systems in various fields. Since the early 2000s, there have been 23 methods, 37 approaches, 8 tools, and about 176 case study applications, most of which are in and emerging from the fields of aviation, medicine, automotive, and space industries. In the work of Khawaji et al. [12], the dynamic correlation properties of STAMP were used to provide a new safety analysis method for chemical processes. Another similar effort from Rejzek and Hilbes [13] utilized STAMP theory to provide new support for the safety analysis of critical systems and equipment in nuclear power plants for their safe operation. In two other published works, STAMP was used to perform functional safety analysis on the train control system, and the relevant factors that trigger the hazard were analyzed [14], and the safety system engineering process of STAMP-based maritime safety management system was designed [15]. In the work of Yousefi and Rodriguez Hernandez [16], STPA was applied to a processing plant as an industrial case study, and recommendations generated by STPA were compared with the HAZOP study.

In civil aviation, STPA [17] was used to model the wheel braking system and to analyze the unsafe control behavior and causes of the system. In addition, a new STPA method was developed to deal with closed-loop actions on a continuous control system and was effectively used to analyze data collected during a crosswind flight test campaign [18]. The work of Schmid et al. [19] extended STPA to pilot behaviors to investigate how these knockout events may be encountered throughout the system and to prevent hazardous pilot behaviors caused by medical incapacitation and homicidal-suicidal behaviors at different levels of the system. Although the STPA method has been widely used for the hazard analysis of large complex systems in aerospace, it is rarely used in IMA systems. Fleming et al. [20] firstly applied the STPA to the IMA system security analysis, focusing on the harm caused by application interactions and data coupling. At the same time, Wang et al. [21] used STPA to model and analyze the communication safety requirements of IMA partitions. However, these analyses did not provide an overall framework for safety analysis, but rather focused on the hazards of IMA sectional interaction.

Formal methods are effective in diluting the influence of human factors by modeling the system through strict semantics and using rigorous mathematical methods to analyze and verify the relevant characteristics of the system. Therefore, many scholars have studied the combination of STPA and formal verification. Sun and Zhong [22] considered combining STPA with a technique that converts natural language to automaton taking the traffic lights as an example. Abdulkhaleq and Wagner [23] presented an extensible STAMP platform called XSTAMPP as a specific design of tool support for the widespread adoption and use of STPA in various domains, to facilitate STPA application to different systems and to be easily extended to meet diverse requirements and features. Meanwhile, Adbulkhaleq and Wagner [23] proposed SAHRA, a software tool that integrated STPA into the UML/SysML environment, classifies software tools, and performs risk assessments. Dakwat and Villani [24] proposed an approach combining STPA and model checking in order to represent the threat of the system identified by STPA. These studies provided methods to validate unsafe control actions in STPA, providing support for STAMP formalization.

To resolve the aforementioned problems, we analyze the IMA system by combining STPA with formalization and provide an overall safety analysis framework.

3. Preliminaries

3.1. Integrated Modular Avionics. The IMA is comprised of a set of shared hardware and software resources. Its platform is divided into several partitions which can host one or more functions. Different partitions are isolated by the virtual system boundaries spatially or temporally. The platform manages all the resources to provide communications, computing capabilities, and interfaces for different functions. This architecture qualifies IMA with highly configurable capability of resources, which denotes that resources can be easily allocated to meet the different requirements of different functions and reallocated in case of failure of any function. So, the reliability analysis of the IMA partitioning mechanisms should take the failure tolerance of the IMA architecture into consideration. The system model is shown in Figure 1.

In the IMA system, different IMA system functions are configured by different types of core processing modules. As shown in Table 1, the A380 aircraft allocates most of the aircraft functions to 7 types of 22 CPIOMs to support the system's requirements of residing functions.

3.2. Reconfiguration Process. Dynamic reconfiguration capability is the core technology of the new IMA system, which not only reduces hardware redundancy and unexpected maintenance costs, but also improves resource utilization, increases system flexibility, and enhances the responseability of avionics systems to different missions and resource failures. Moreover, it improves the reliability of aircraft operations while maintaining the current safety levels.

The ASAAC standard proposes a new architecture of IMA architecture with reconfiguration features, as shown in Figure 2. It contains a total of three layers. The Application Layer (AL) is the top layer of the three-tier structure, used to perform various aircraft functions. Each functional application is decomposed into several parallel processing units. The Operating System Layer (OSL) is the middle layer, mainly responsible for managing onboard resources and providing an execution platform for upper application software; its main components include Generic System Management (GSM) and Operating System (OS). The Module Support Layer (MSL) contains all details of the underlying hardware, including services for loading, communication, time, and self-test. In the ASAAC standard, the reconfiguration behavior of the IMA system is controlled by the generic system management. When the reconfiguration is triggered due to the module failure, the generic system management obtains the configuration information from the blueprint system to implement the system reconfiguration.

Taking the hosted application "Landing Gear System" of Table 1 as an example for analyzing the IMA reconfiguration process, there are 4 Common Functional Modules (CFMs) in IMA which are interconnected by the AFDX network to provide common processing capabilities for the application. A CFM can support one or more processes and run them independently. This application includes 10 processes, in which there are 5 source processes. P1 and P2 are running on CFM1. P6 is running on CFM2, P4 is running on CFM3, and P10 is running on CFM4, as shown in Figure 3(a). When CFM2 fails, the IMA system will assign the task running on the failed module to other available CFM. Figure 3(b) shows the correct reconfiguration after CFM2 fails.

3.3. Stamp/STPA. Systems-Theoretic Accident Model and Process, referred to as STAMP, is a new causal model based on system theory and control theory. It assumes that the loss is caused by the failure of security constraints to be effectively implemented and changes the focus of system safety from preventing failures to implementing security constraints. STAMP's key feature is the STAMP control model. It is a system model that is composed of feedback control loops, which contains at least 5 types of elements: controllers, control actions, feedback, other inputs to and outputs from systems/components (neither control nor feedback), and controlled processes, as shown in Figure 4.

The System Theory Process Analysis (STPA) is a safety analysis method based on STAMP. It includes 4 steps:

(i) The first step is to define the purpose of the analysis

(ii) The second step is to build a model of the system called a control structure, which captures functional relationships and interactions by modeling the system as a set of feedback control loops. This control structure is based on the STAMP control model

(iii) The third step is to analyze the control actions in the control structure, examining how they could lead to the losses defined in the first step

(iv) The fourth step identifies the reasons why unsafe control might occur in the system

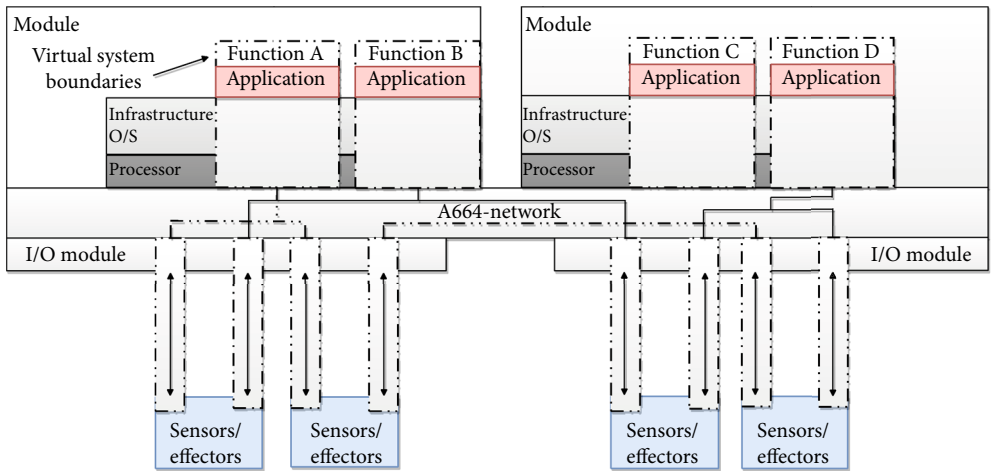

Figure 1: Model of the IMA system.

Table 1: CPIOMs in the A380.

Module type	Number	Application deployment
CPIOM-A	4	Cabin function
CPIOM-B	4	Air management
CPIOM-C	2	Flight control and communication alarm
CPIOM-D	2 + 1 (optional)	Data link function
CPIOM-E	2	Electrical system
CPIOM-F	4	Fuel management
CPIOM-G	4	Landing gear system
IOM	8	Input/output

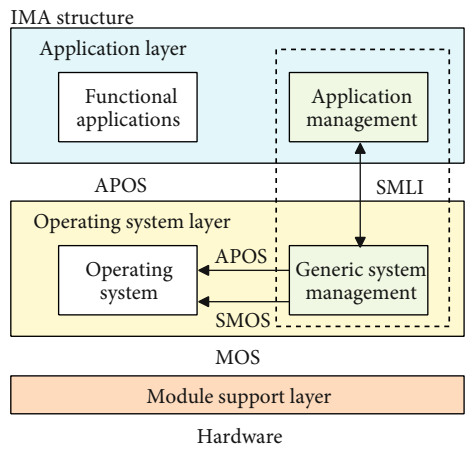

Figure 2: IMA structure of ASAAC.

4. Safety Assessment of the IMA System

4.1. STPA-UPPAAL Model. The advantage of STPA is to provide a system model based on control loops and a framework for identifying and analyzing hazards. STPA describes the system through a hierarchical control structure based on an adaptive feedback mechanism, instead of the event chain model. It can better describe the multiparty interactions of the system and the nonlinear interactions among system components, rather than blindly oversimplify the causality and process of the accident, so as to better identify and analyze the hazards. STPA is very suitable for complex systems, especially sociotechnical systems. While when dealing with complex avionics systems, it will face some limitations, such as heavy workload and prone to human errors. To solve these problems, the formal verification UPPAAL based on time automata is introduced considering that the IMA system is a real-time system with strict time constraints. UPPAAL can provide STPA with a formal modeling method and automated analysis, which improves the efficiency and accuracy of the analysis and reduces the workload and the influence of human factors on the analysis results. At the same time, STPA provides modeling and analysis frameworks for formal modeling and analysis. The combination of the above two makes up for each other's shortcomings and is more suitable for complex avionics systems.

UPPAAL is a real-time system verification tool based on the Timed Automata theory developed by Uppsala University of Sweden and Aalborg University of Denmark, which is comprised of an editor, a simulator, and a verifier. Timed Automata (TA) is a finite-state machine extended with clock variables, using a dense-time model in which the clock variables are computed as real numbers. All the clocks progress synchronously [25]. In UPPAAL, a system is modeled as a network of several such timed automata in parallel. The model is further extended with bounded discrete variables that are part of the state. These variables are used in programming languages: they are read, written, and subject to common arithmetic operations. The state of the system is defined by the locations of all automata, the clock constraints, and the values of the discrete variables. Every automaton may trigger an edge separately or synchronizer with another automaton, which leads to a new state. UPPAAL can not only describe the continuous-time characteristics of the system but also reflect the characteristics of

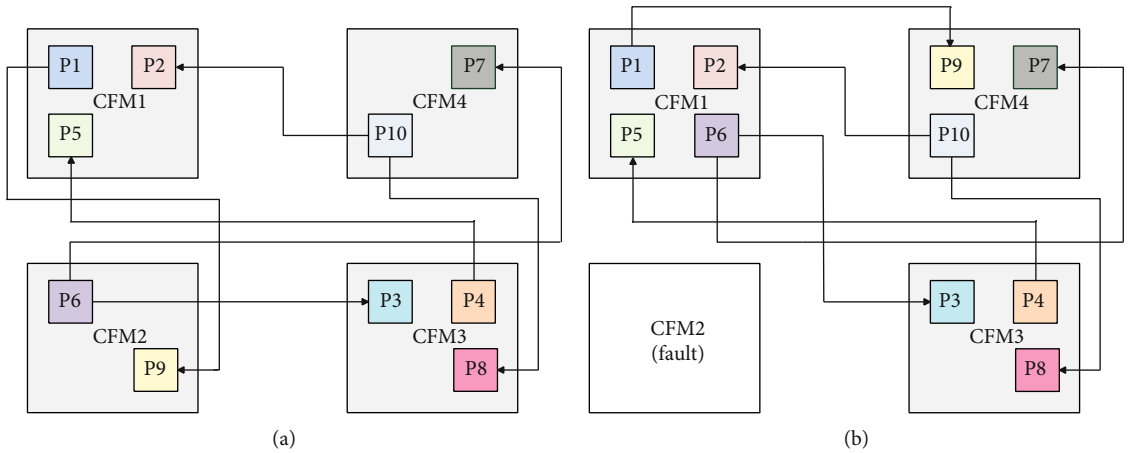

Figure 3: Configuration/reconfiguration for the hosted application "Landing Gear System": (a) initial configuration; (b) correct reconfiguration after CFM2 fails.

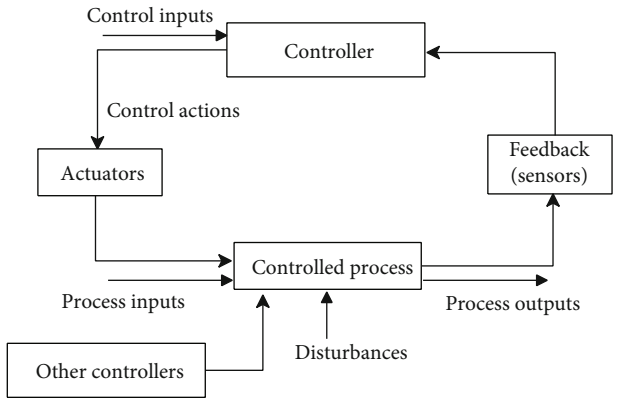

Figure 4: The STAMP control model.

multiparty interactions. With Backus-Naur Form (BNF) grammar for the accessibility analysis of the model, it is widely used in many industrial fields [26–28].

A new method for IMA reconfiguration based on STAMP and Time Automata, STPA-UPPAAL, is proposed. The process of combining the two and the conversion method is studied, and the detailed analysis framework is given. As shown in Figure 5, there are a total of 4 steps. In order to clarify the issue, the following three aspects are provided.

First, the possible accidents and system-level hazards are identified by analyzing the system application scenarios. Then, the system hierarchical safety control structure is constructed. The control structure captures the functional relationships and interactions of the system through a set of feedback control loops. Control structures usually start at a more abstract level and are iteratively tuned to capture more system details. Once the safety control structure is constructed, STPA is used to identify the potential UCAs of the system and to classify the damage and hazards that may result.

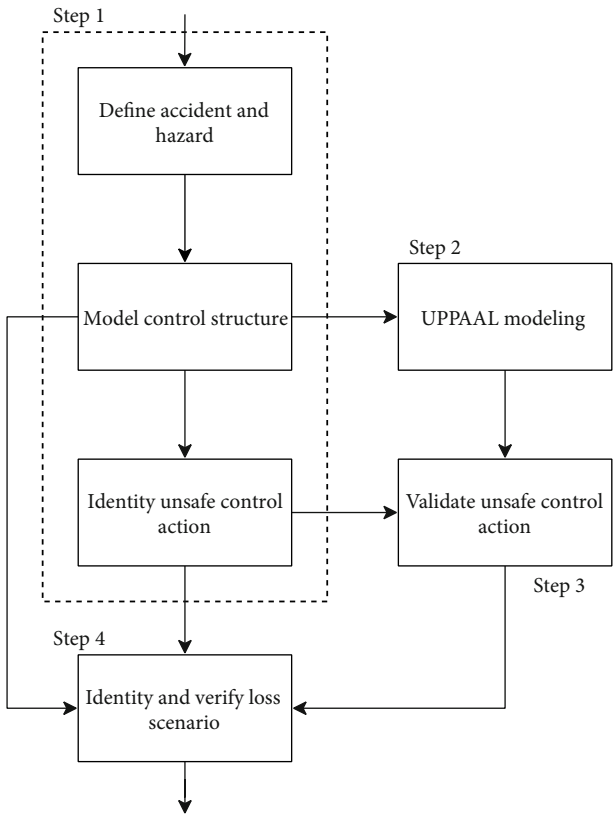

Figure 5: STPA-UPPAAL steps.

Next, the system is modeled by UPPAAL based on the system hierarchical safety control structure. The UPPAAL model mainly includes the following elements:

(i) *TEMPLATE*. It represents a time automata model, which is composed of several LOCATIONs, with a total of four types: Initial, Conventional, Urgent, and Committed, the latter two locations have no time delay.

(ii) *SYNCHRONISATION*. It can be synchronized through the channel for information transmission. "!" means sending and "?" means accepting.

(iii) *UPDATE*. It can assign values to variables during position transfer.

(iv) *SELECT*. It is used to constrain the value range of integer variables.

(v) *GUARD*. It is a conditional trigger for the position transfer.

(vi) *CLOCK*. It is used to constrain the time of the position and trigger the transfer.

The STPA-UPPAAL modeling rules are as follows:

(1) Create and declare a system-level UPPAAL model

(2) Create a corresponding TEMPLATE according to the STPA safety control structure; each component corresponds to a TEMPLATE

(3) Create corresponding LOCATIONs according to the functions of component, define location variables, processing functions, and declare them

(4) Define location transitions and boundaries by using SYNCHRONISATION, UPDATE, SELECT, and GUARD

(5) Identify the failure mode of each component, declare the failure mode, and inject it into the model by CLOCK

After modeling, the potential UCAs identified by the STPA are converted into a BNF statement, which can be verified by the state accessibility analysis of the time automaton network model. To verify the passing UCAs, a detailed root cause analysis is required.

Finally, some appropriate context is constructed to explain how incorrect feedback, inadequate requirements, design errors, component failures, etc., lead to a UCA and how improper compliance with or enforcement of the provided safety controls can lead to danger. Based on the established formal specifications, the rigorous algorithms are utilized to describe and validate the relevant characteristics of the system in detail to determine whether the system meets the desired characteristics and to find the scenarios of UCAs, thereby diluting the impact of human factors on the results of the analysis.

Next, take the simple IMA system assumed in Section 3.2 as an example for case analysis. The following assumptions are made for the next analysis: (1) the aircraft operation is in the approach phase, (2) the initial configuration process for the default application is correct, and (3) the reconfiguration scenario is CFM2 fails and IMA needs to be reconfigured.

4.2. Defining Accident and Hazard. System-level accidents that may result from IMA reconfiguration during the approach phase include the following:

(i) (A-1) Personal injury

(ii) (A-2) Aircraft damage

(iii) (A-3) Ground facilities damage

System-level hazards that can be obtained based on the definition of system-level accidents and application scenarios include the following:

(i) (H-1) Aircraft collision with obstacles, related accidents: A-1, A-2, A-3

(ii) (H-2) Aircraft out of control, related accidents: A-1, A-2, A-3

(iii) (H-3) Rush out of runway, related accidents: A-1, A-2

4.3. Modeling Control Structure. The safety control structure has two layers: controller and controlled process, which interact through control commands and feedback information. The safety control structure of the hosted application is defined based on IMA technical documentation and the STPA handbook, as shown in Figure 6.

For the hosted application "Landing Gear System," the safety control structure consists of IMA, hydraulic lines, landing gear, and sensors. IMA is the controller. Hydraulic lines are the actuators. Landing gear is the controlled process. Sensors are the feedbacks. The pilot starts the application and sends operation commands to the IMA through the operation panel. After the application is started, IMA collects data from other systems and sensors, processes it, and sends it to the hydraulic line. Then, hydraulic pressure is used to drive the landing gear. At the same time, the sensors on the landing gear will monitor its status in real-time and return the data to IMA. IMA will also send the collected data and part of the processed data to the Primary Flight Display (PFD) for the pilot to view.

In the IMA, the reconfiguration process can also be described by the subsafety control structure. It includes 4 parts: Application Layer (AL), Operating System Layer (OSL), Module Support Layer (MSL), and Hardware. AL and OSL are controllers; MSL is both the actuator and the feedback. Hardware is the controlled process. The IMA subsafety control structure is shown in the IMA part of Figure 6.

AL includes applications and Application Management (AM). OSL includes Generic System Management (GSM), Operating System (OS), and a real-time partitioned operating system. After the application is started, OS allocates partitions to the application and performs unified management through AM. At the same time, OS will request configuration from the GSM. GSM includes four parts: Health Monitoring (HM), Fault Management (FM), Configuration Management (CM), and Security Management (SM).

HM is responsible for assessing the health status. Its main purpose is that of filtering faults/errors and passing on any information concerning confirmed faults/errors to the Fault Management function where further diagnostics and corrective action can be taken. AL, MSL, and OS all have HM service and can communicate with HM of GMS. FM is

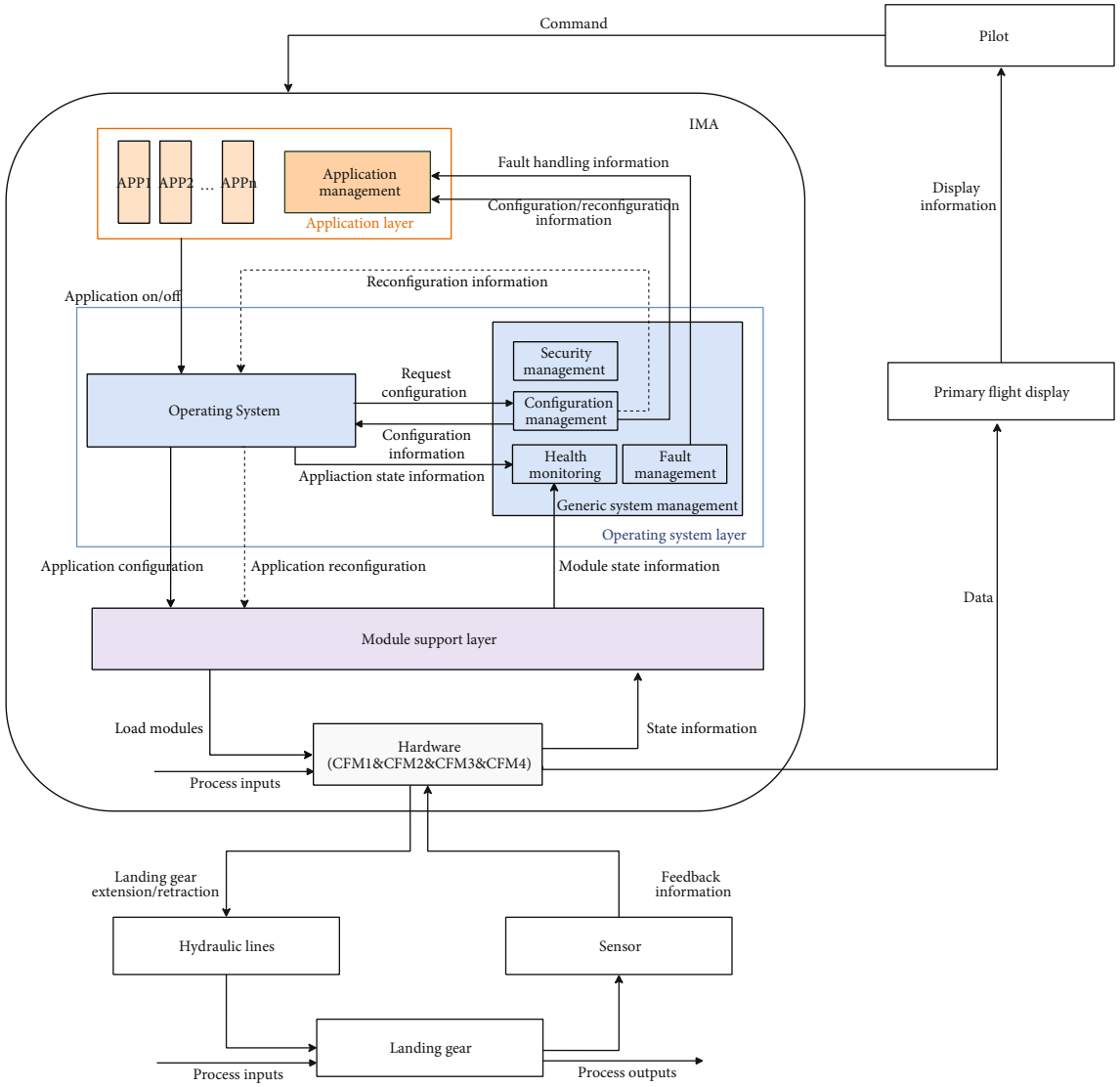

Figure 6: Safe control structure of the hosted application.

responsible for identifying, masking, confining, and localizing faults in order to prevent a total or partial system failure and to ensure that the system may remain operational for the required period in the presence of faults. CM is responsible for establishing the initial system configuration, any subsequent reconfiguration due to either pilot mode change requests or faults/errors being handled, and ultimately shutdown of the system. All configuration type behavior shall be described by the information contained with the RTBPs. SM is responsible for implementing the system security policy.

After GMS receives the request, CM returns configuration information to the operating system. The operating system configures the corresponding hardware processing module for the software through the module support layer according to the received configuration information. In addition, the hardware and software will feedback their status information to the operating system, and the operating system will feedback the collected status information to the HM of the general management system. HM recognizes the fault and sends the information to FM. After FM analysis, if the fault can be resolved through reconfiguration, it will request reconfiguration from CM, and CM will send reconfiguration information to the operating system for reconfiguration.

4.4. Identifying Unsafe Control Action. During the STPA analysis, the UCA is a control action that, in a particular context or worst-case scenario, will lead to a hazard. A control action can be unsafe in the following ways:

(i) Not providing the control action leading to a hazard

(ii) Providing the control action leading to a hazard

(iii) Providing a potentially safe control action but too early, too late, or in the wrong order

Table 2: Unsafe control actions.

Type	UCA description	Possible hazards	Possible accident
Not providing causes hazard	(UCA-01) the IMA system was not reconfigured after CFM2 failed	H-1, H-2, H-3	A-1, A-2, A-3
Providing causes hazard	(UCA-02) the IMA system was incorrectly reconfigured after CFM2 failed	H-1, H-2, H-3	A-1, A-2, A-3
Provided too late or too early	(UCA-03) the IMA system was not reconfigured in time after the failure of CFM2	H-1, H-2, H-3	A-1, A-2, A-3
Stop too late or too early	(UCA-04) the IMA system reconfiguration took too long after CFM2 failed	H-1, H-2, H-3	A-1, A-2, A-3

(iv) The control action lasting too long or being stopped too soon

Since the focus of this paper is the analysis of the IMA reconfiguration process, UCA identification and subsequent analysis are mainly based on the IMA subsafety control structure.

Here, taking the scenario "CFM2 fails and IMA needs to be reconfigured" as an example, "IMA reconfiguration" is taken as the control action to identify the UCAs for analyzing the safety of IMA reconfiguration. The results are shown in Table 2.

4.5. UPPAAL Modeling. For UCAs identified by STPA, further validation is indispensable to determine the actual hazards that occur during system operation. The rigorous semantics and strict mathematical logic are used to model and analyze the system, which can effectively dilute the influence of human factors and cut down system development costs. Due to the real-time nature of IMA, the formal modeling tool—UPPAAL—is used to perform detailed modeling of it.

Based on the IMA architecture, system functions, and STPA safety control structure, a timed automaton network model of the hosted application "Landing Gear System" is constructed. Since the focus of this paper is the IMA reconfiguration process, only a timed automaton network model of IMA reconfiguration is given, as shown in Figure 7, which is IMA part of the complete timed automaton network model.

The timed automaton network model of IMA reconfiguration includes 8 UPPAAL templates: AL, OS, GSM, MSL, CFM1, CFM2, CFM3, and CFM4, where the CFM models are the same.

The IMA reconfiguration is as follows: after MSL's Health Management (HM) service receives a fault or error report, it filters the fault based on the current configuration and fault filtering algorithms of the system. Once a fault is confirmed, it will send a message to the OS's HM service notifying the information of the fault and the diagnostic data. The GSM's HM obtaining the fault information filters the errors and reports it to GSM's Fault Management (FM). The FM determines the fault solution by querying the database. If there is a feasible solution to the fault, a message will be sent to the GSM's Configuration Management (CM) requesting a change in the current system configuration. Upon receipt of the message, it goes into RTBP, obtains a list of actions, and implements them.

To make the safety analysis completely, the failure modes of the system obtained through FMEA need to be injected into the model. UPPAAL can use CLOCK variables to distinguish the functional state of the operating system layer and the module support layer. Variable t represents the maximum time limit for component operation. When the CLOCK variable is less than or equal to t, the function is considered to be in the executed state, which contains two types: $\{1, 2\} = \{\text{correct execution, wrong execution}\}$; when the CLOCK variable is greater than t, the function is considered to be in an unexecuted state: $\{0\} = \{\text{no execution}\}$. For the hardware layer (CFM1-4), the status of the detection function includes the following three types: $\{-1, 0, 1\} = \{\text{wrong detection, no detection, correct detection}\}$.

In order to ensure the correctness of the analysis, the validity of the model must be checked first. The test content includes two aspects: system logic and time-series. UPPAAL checks the model by using the BNF grammar [25]. The descriptions are as follows:

(i) E<>p means that if a state in a transition sequence satisfies p, E<>p is true

(ii) A[]p is equivalent to (not E)<>(not p).

(iii) E[]p means if there is a transition sequence that satisfies p in all states, then $E[]p$ is true

(iv) A<>p is equivalent to $(\text{not } E)[](\text{not } p)$

(v) $p \longrightarrow q$ means that if p is true, then q is also true

Specific logic verification statements and time-series verification statement are shown in Table 3. The results show that all properties have been satisfied. The formal model is valid, meeting the operational requirements of the system.

4.6. Validating Unsafe Control Action. A state accessibility analysis of the network model of time automaton is used to verify the unsafe control behavior identified by STPA. The BNF statement "E<>UCA" (i.e., the presence or absence of a transfer path to make the unsafe control action occur) should be used to verify the presence of a UCA. If the statement "E<>UCA" is met, it means that the unsafe control actions will occur, and the corresponding safety will not be met. Table 3 shows the BNF verification statements for

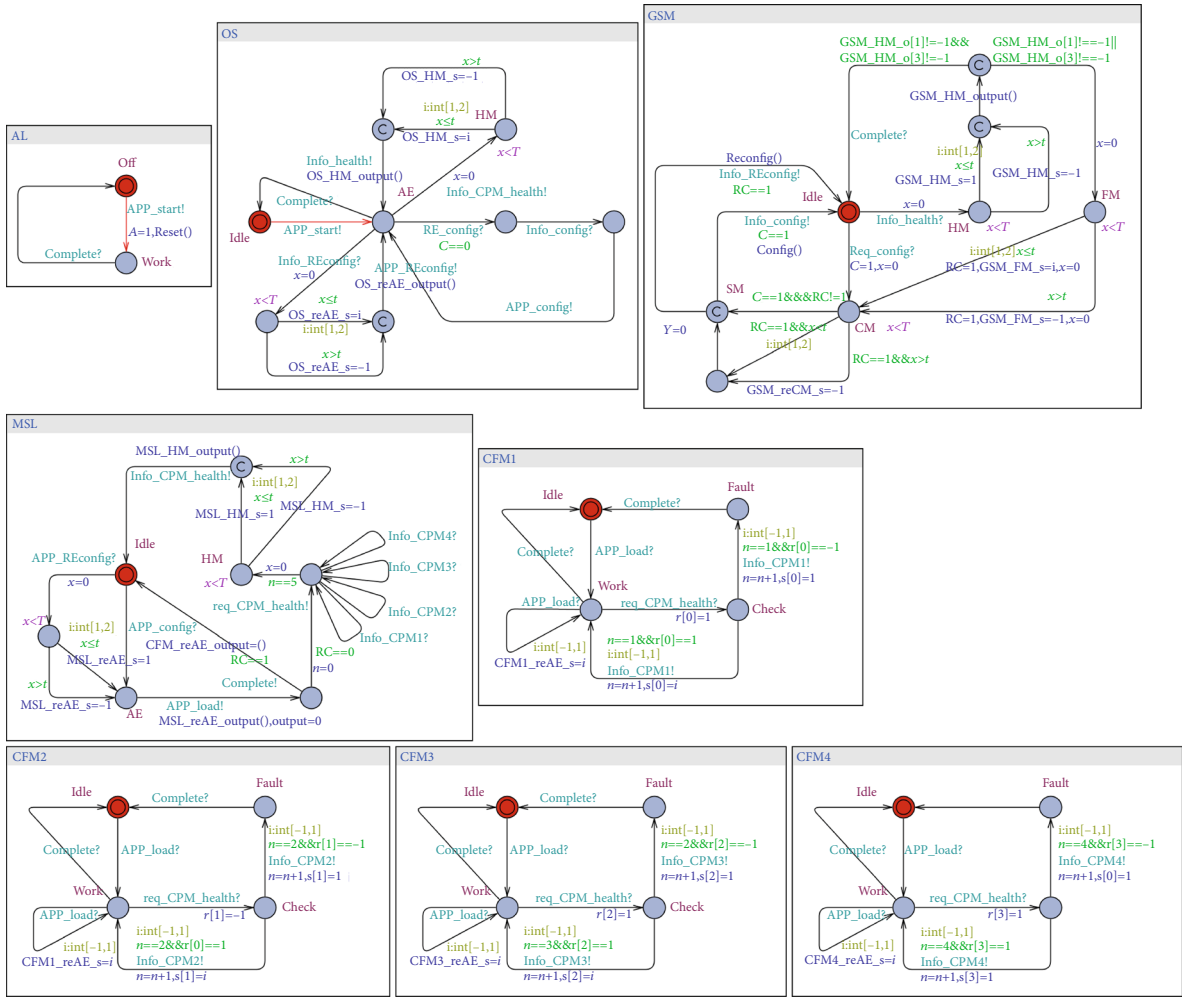

Figure 7: Timed automaton network model of IMA reconfiguration.

UCA-1 to UCA-4. The results show that the properties of UCA-1 to UCA-4 are of a satisfactory nature in Table 4.

4.7. Identifying and Verifying Loss Scenarios. Once the unsafe control actions that lead to the hazards have been identified and validated, the reasons for their occurrence need to be analyzed. Based on the control feedback model and general categories summarized for scenario analysis, a scenario analysis framework is obtained, as shown in Figure 8. There are four general categories:

(i) Unsafe controller behaviors

(ii) Causes of inadequate feedback information

(iii) Scenarios involving the control path

(iv) Scenarios related to the controlled process

Each one of them can be further refined according to the system architecture and be converted into the corresponding BNF statement.

After that, UCA-2 has been analyzed as an example.

(i) For the Operating System Layer, there are three reasons for UCA-2, inappropriate control algorithms, unsafe control inputs from other controllers, and inadequate process models. Scenarios involving the operating system layer are as follows:

(a) *SCENARIO-01.* The OS receives fault feedback information at a certain moment, while in fact, CFM is running normally.

(b) *SCENARIO-02.* The OS incorrectly assumes that CFM2 has no failure and other CFM fails.

(c) *SCENARIO-03.* The GSM's control algorithm is flawed.

(d) *SCENARIO-04.* The GMS incorrectly assumes that CFM2 has no failure and other CFM fails.

(e) *SCENARIO-05.* Unsafe control inputs from other controllers.

(ii) For the Module Support Layer, improper execution of control action and inappropriate feedback will

Table 3: Model verification statements.

Type	Property	BNF statement	Result
Logic verification statements	IMA system is not deadlock	A[] not deadlock	Satisfy
	Application layer works normally	E<>AL.Off or AL.Work	Satisfy
	Operating system works normally	E<>OS.Idle or OS.AE or OS.HM	Satisfy
	Generic system management works normally	E<>GSM.Idle or GSM.CM or GSM.SM or GSM.HM or GSM.FM	Satisfy
	Module support layer works normally	E<>MSL.Idle or MSL.AE or MSL.HM	Satisfy
	Common functional module X works normally (X = 1,2,3,4)	E<>CFMX.Idle or CFMX.Work or CFMX.Check or CFMX.Fault	Satisfy
Time-series verification statements	The MSL will not load the module until the system is configured or reconfigured	A[]MSL.AE imply (C==1 or RC==1)	Satisfy
	MSL will enter HM only after the module works	A[]MSL.HM imply (s[0]!=0 and s [1]!=0 and s[2]!=0 and s [3]!=0)	Satisfy
	MSL will enter HM only after MSL and OS enter HM	A[]GSM.HM imply (MSL_HM_s! =0 and OS_HM_s!=0)	Satisfy
	GSM will enter FM after detecting faulty	A[]GSM.FM imply (HM_s!=2&& HM_s!=3)	Satisfy
	GMS will enter CM after FM provides the solution	E<>GSM.CM imply RC==1	Satisfy

Table 4: UCA validation statement.

Number	Property	BNF statement	Result
UCA-01	The IMA system was not reconfigured after CFM2 failed	E<> (CFM2.Fault) and (RC==0)	Satisfy
UCA-02	The IMA system was incorrectly reconfigured after CFM2 failed	E<> (CFM2.Fault) and (RC==1)and(o!=1)	Satisfy
UCA-03	The IMA system was not reconfigured in time after the failure of CFM2	E<> (CFM2.Fault) and (RC==1) and (GSM.CM) imply x<=t&&x>t1	Satisfy
UCA-04	The IMA system reconfiguration took too long after CFM2 failed	E<> (CFM2.Fault) and (MSL.AE)and (RC==1) and (OS_reAE_s==1) and (MSL_reAE_s==1) imply y<=T&&T>T1	Satisfy

lead to the occurrence of UCA-2. In general, scenarios involving MSL might include the following:

(a) *SCENARIO-06*. Reconfiguration information is sent by OS but MSL responds inadequately.

(b) *SCENARIO-07*. Reconfiguration information is not sent by MSL, but MSL or other elements respond as if they had been sent.

(c) *SCENARIO-08*. The CFM sensor does not send feedback information, but the feedback information is received or applied by the MSL.

(d) *SCENARIO-09*. The CFM sensor is working properly, but the MSL receives incorrect feedback information.

(iii) For Hardware Layer, when the CFM sensor feeds back the incorrect status information, it will cause the OSL to perform an incorrect reconfiguration. Furthermore, even if CFM received reconfiguration information, it may be invalidated or overridden by other controllers, resulting in the occurrence of UCA-2. The relevant scenarios are as follows:

(a) *SCENARIO-10*. The CFM sensor feeds back the wrong status information.

(b) *SCENARIO-11*. The reconfiguration information is received by the CFM, but the CFM makes an error response.

Next, as shown in Table 5, the scenarios obtained from the above analysis have been converted into BNF and verified by UPPAAL. From the verification results, it is found that incorrect transmission or reception of CFM status information and reconfiguration information is the main cause of UCA-2. Usually, these problems are caused by communication failures such as transmission errors, data loss, and communication delays. Therefore, it is necessary to analyze them and give corresponding safety constraints.

5. Safety Constraint Analysis

To reduce data transmission loss, a communication CHANNEL_{ij} assigned between the source process PROCES S_i running on CFM_i and the destination process PROCESS_j running on CFM_j is analyzed. The source process PROCES

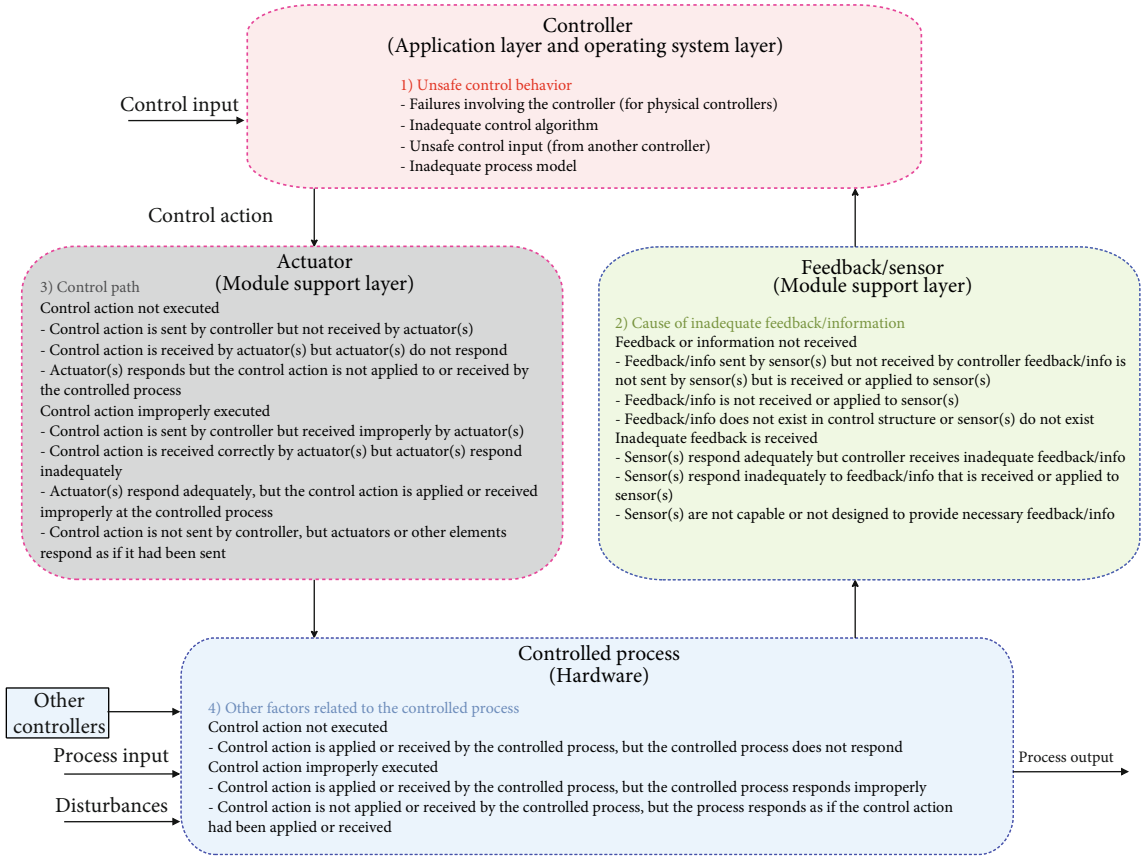

Figure 8: Loss scenario analysis framework.

Table 5: SCENARIO verification statement.

Property	BNF statement	Result
SCENARIO-01	E<>(CFM2.Fault)and(CFM3.Work)and (GSM.FM)and(GSM_HM_o[3]!=1)	Satisfy
SCENARIO-02	E<>(CFM2.Fault)and(GSM.HM)and (OS_HM_o [1]==1 && OS_HM_o[3]==-1)	Satisfy
SCENARIO-03	E<>(GSM.FM)and(GSM_HM_o[0]==1&&GSM_HM_o[1]== 1&&GSM_HM_o[2]==1&&GSM_HM_o[3]==1)	Dissatisfy
SCENARIO-04	E<>(CFM2.Fault)and(GSM.FM)and(GSM_HM_o[1]==1&&GSM_HM_o[3]==-1)	Satisfy
SCENARIO-05	E<> (OS.AE) and (A!=1)	Dissatisfy
SCENARIO-06	E<>(CFM2.Fault)and(RC==1)and(OS.AE)and(output==1) and(MSLd[0]!=OSd[0]&&MSLc[2] !=OSc[2])	Satisfy
SCENARIO-07	E<>(CFM2.Fault)and(RC==1)and(OS_reAE_s==-1)and (output==1) and(MSLd[0]!=0&&MSLc[2]!=0)	Dissatisfy
SCENARIO-08	E<>(CFM2.Fault)and(OS.HM)and(s[1]==0)and(MSL_HM_o[1]!=0)	Dissatisfy
SCENARIO-09	E<>(CFM2.Fault)and(OS.HM)and(MSL_HM_o[1]==s[1]||MSL_HM_o[3]==s[3])	Satisfy
SCENARIO-10	E<>(CFM2.Fault)and(MSL.HM)and(s[1]==r[1]||s[3]==r[3])	Satisfy
SCENARIO-11	E<>(RC==1&&output==1)and(AL.Off)and(MSL.Idle)and (Hd[0]!=MSLd[0]||Hc[2]!=MSLc[2])	Satisfy

S_i periodically sends a message before its execution ends, and the generated message must be read before the destination process $PROCESS_j$ gets the next processing time slice. Here, $MESSAGE_n$ is used to represent the nth message on the virtual link.

There are several delays during the communication:

(i) *Source Delay (SD)*. Messages generated by the sensor must be first stored in the buffer memory. When the source process on the module connected to the

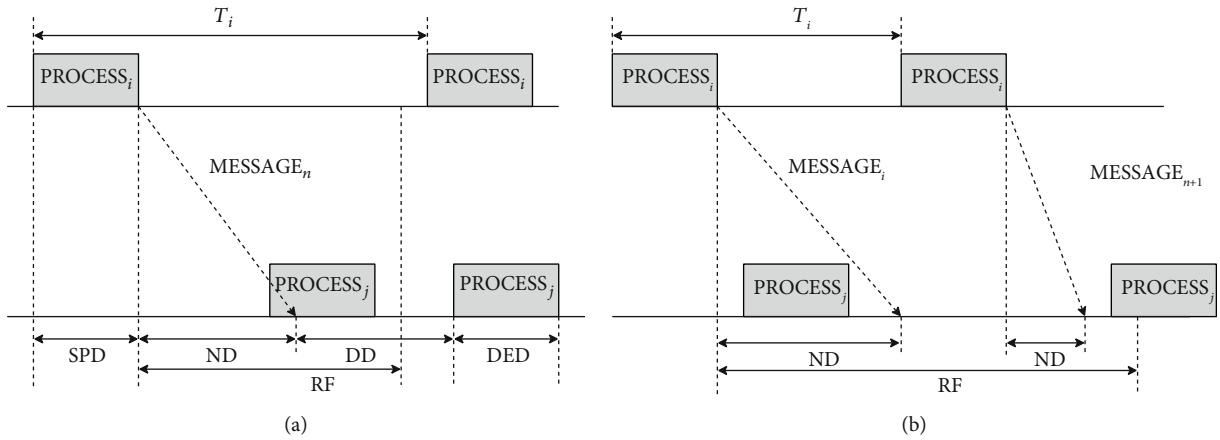

Figure 9: Message loss.

sensor obtains the processing time slice, the message is read from the buffer memory.

(ii) *Source Processing Delay (SPD)*. The processing time required for the source process.

(iii) *Network Delay (ND)*. Each message consumes a certain period of time in the communication channel. This period of time is called network delay. The network delay ND is assumed between the intervals $[\mathrm{ND}_{\min}, \mathrm{ND}_{\max}]$, where $\mathrm{ND}_{\min}$ is network delay in the best case and $\mathrm{ND}_{\max}$ is network delay in the worst case.

(iv) *Destination Delay (DD)*. When the message arrives, it may need a period of time for the process to acquire the message before it can be read. This period between the arrival of data and the acquisition of data by the process is called the destination delay.

(v) *Destination Execution Delay (DED)*. The processing time required for the destination process.

There are two reasons for the loss of messages:

(i) The end-to-end communication delay consumed by the message $\mathrm{MESSAGE}_n$ is greater than the refresh parameter value RF, as shown in Figure 9(a)

(ii) The message $\mathrm{MESSAGE}_n$ is overwritten by the message $\mathrm{MESSAGE}_{n+1}$ before being read by the destination process $\mathrm{PROCESS}_j$, as shown in Figure 9(b)

In order to reduce the loss of messages, it is necessary to adjust the period T_j of $\mathrm{PROCESS}_j$ to make constraints so that the two loss scenarios do not occur.

Constraint:

$$T_j \le \min\left(\mathrm{RF} - \mathrm{ND}_{\max}, T_i - \left(\mathrm{ND}_{\max} - \mathrm{ND}_{\min}\right)\right). \quad (1)$$

Prove:

To meet the channel-related refresh parameter RF constraint, the delay that the message is sent to be acquired by the process cannot exceed RF. In the worst case, the maximum network delay and the destination delay are equal to the period T_j of $\mathrm{PROCESS}_j$, so:

$$\begin{aligned} \mathrm{ND} + \mathrm{DD} &\le \mathrm{RF}, \\ \mathrm{ND}_{\max} + T_j &\le \mathrm{RF}, \\ T_j &\le \mathrm{RF} - \mathrm{ND}_{\max}. \end{aligned} \quad (2)$$

For preventing the $\mathrm{MESSAGE}_n$ from being overwritten by the $\mathrm{MESSAGE}_{n+1}$, the period T_j of $\mathrm{PROCESS}_j$ must be reduced to avoid that $\mathrm{PROCESS}_j$ receives two messages at the same time. Suppose $\mathrm{PROCESS}_i$ sends a message at moment M_n and M_{n+1}, respectively, it gives:

$$\begin{aligned} M_n &= \mathrm{SD} + nT_i + \mathrm{SPD} + \mathrm{ND}_n, \\ M_{n+1} &= \mathrm{SD} + (n+1)T_i + \mathrm{SPD} + \mathrm{ND}_{n+1}. \end{aligned} \quad (3)$$

If $\mathrm{PROCESS}_j$ receives these two messages in sequence, T_j needs to satisfy:

$$\begin{aligned} &T_j \le M_{n+1} - M_n, \\ &T_j \le \mathrm{SD} + (n+1)T_i + \mathrm{SPD} + \mathrm{ND}_{n+1} - (\mathrm{SD} + nT_i + \mathrm{SPD} + \mathrm{ND}_n), \\ &T_j \le T_i + \mathrm{ND}_{n+1} - \mathrm{ND}_n. \end{aligned} \quad (4)$$

Since the period T_i of $\mathrm{PROCESS}_i$ is fixed and cannot be changed, T_j must be smaller than the minimum value of the right side in order to make the inequality be always true, therefore:

$$\begin{aligned} T_j &\le T_i + \mathrm{ND}_{\min} - \mathrm{ND}_{\max}, \\ T_j &\le T_i - \left(\mathrm{ND}_{\max} - \mathrm{ND}_{\min}\right). \end{aligned} \quad (5)$$

At last, combining the two constraints above, the maximum period T_j of PROCESS_j is defined as follows:

$$T_j \leq \min\left(\text{RF} - \text{ND}_{\max}, T_i - (\text{ND}_{\max} - \text{ND}_{\min})\right). \quad (6)$$

6. Conclusion

The contributions of the paper include two aspects. The first contribution of the paper is to propose an analysis framework of STPA-UPPAAL. It identifies the unsafe control (UCAs) of IMA using STPA and the UCAs are verified by the state accessibility analysis using UPPAAL. The proposed framework combines the advantages of the two methods. Then, the framework is applied to the reconfigurable IMA which is a typical complex system. The chief research conclusions can be summed up below:

(1) The combination process and conversion method of STPA and UPPAAL are studied, and the detailed analysis framework of STPA-UPPAAL is proposed to support the safety analysis of IMA reconfiguration

(2) Taking a simple IMA system as an example, a case analysis is carried out. STPA safety control structure and the formal model of IMA reconfiguration are established. Potential UCA and potential cause scenarios are identified by STPA. And they are validated and verified through UPPAAL. The identified IMA dangerous cause "data transmission loss" is analyzed, and the safety constraints are given

(3) The results show the feasibility of STPA-UPPAAL for complex avionics systems. Compared with the traditional methods, it describes the system through hierarchical control based on adaptive feedback mechanism, uses automated analysis to improve the efficiency and accuracy of the analysis, reduces the workload, and the influences human factors on the analysis results

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This project has received funding from the National Natural Science Foundation of China - Civil Aviation Joint Research Fund of China Civil Aviation Administration (U1933106), Aeronautical Science Foundation of China (20185167017), and Fundamental Research Funds for the Central Universities of Civil Aviation University of China (3122019167).

References

[1] H. Wang and W. Niu, "A review on key technologies of the distributed integrated modular avionics system," *International Journal of Wireless Information Networks*, vol. 25, no. 3, pp. 358–369, 2018.

[2] M. García-Valls, J. Domínguez-Poblete, I. E. Touahria, and C. Lu, "Integration of data distribution service and distributed partitioned systems," *Journal of Systems Architecture*, vol. 83, pp. 23–31, 2018.

[3] C. Zhao, P. Wang, and F. Yan, "Reliability analysis of the reconfigurable integrated modular avionics using the continuous-time Markov chains," *International Journal of Aerospace Engineering*, vol. 2018, Article ID 5213249, 8 pages, 2018.

[4] X. Zhou, H. Xiong, and F. He, "Hybrid partition- and network-level scheduling design for distributed integrated modular avionics systems," *Chinese Journal of Aeronautics*, vol. 33, no. 1, pp. 308–323, 2020.

[5] 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.

[6] R. Fuchsen, "IMA NextGen: a new technology for the Scarlett program," *IEEE Aerospace and Electronic Systems Magazine*, vol. 25, no. 10, pp. 10–16, 2010.

[7] G. Montano, *Dynamic Reconfiguration of Safety-Critical Systems: Automation and Human Involvement*, University of York, 2011.

[8] K. Kushal, S. Nanda, and J. Jayanthi, "Architecture level safety analyses for safety-critical systems," *International Journal of Aerospace Engineering*, vol. 2017, Article ID 6143727, 9 pages, 2017.

[9] M. Li, G. Zhu, Y. Savaria, and M. Lauer, "Reliability enhancement of redundancy management in AFDX networks," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 5, pp. 2118–2129, 2017.

[10] D. Suo, J. An, J. Wu, and J. Zhu, "Filling the gap between IMA development and safety assessment through safety-driven model-based system engineering," in *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC)*, p. 6C6-1, Williamsburg, VA, USA, October 2012.

[11] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*, MIT Press, 2012.

[12] I. A. Khawaji, *Developing a System-Based Indicators for Proactive Risk Management in the Chemical Processing Industry*, MIT, MA, USA, 2012.

[13] M. Rejzek and C. Hilbes, "Use of STPA as a diverse analysis method for optimization and design verification of digital instrumentation and control systems in nuclear power plants," *Nuclear Engineering and Design*, vol. 331, pp. 125–135, 2018.

[14] Y. Zhang and S. Liu, "STPA based safety analysis of regional data center in CTCS-1 train control system," in *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, pp. 240–245, Chongqing, China, December 2018.

[15] O. A. Valdez Banda and F. Goerlandt, "A STAMP-based approach for designing maritime safety management systems," *Safety Science*, vol. 109, pp. 109–129, 2018.

[16] A. Yousefi and M. Rodriguez Hernandez, "Using a system theory based method (STAMP) for hazard analysis in process industry," *Journal of Loss Prevention in the Process Industries*, vol. 61, pp. 305–324, 2019.

[17] L. Zheng and J. Hu, "Safety analysis of wheel brake system based on STAMP/STPA," *Hangkong Xuebao/Acta Aeronautica et Astronautica Sinica*, vol. 38, no. 1, pp. 1327–1339, 2017.

[18] D. S. Castilho, L. M. S. Urbina, and D. de Andrade, "STPA for continuous controls: a flight testing study of aircraft crosswind takeoffs," *Safety Science*, vol. 108, pp. 129–139, 2018.

[19] D. Schmid, M. Vollrath, and N. A. Stanton, "The System Theoretic Accident Modelling and Process (STAMP) of medical pilot knock-out events: pilot incapacitation and homicide-suicide," *Safety Science*, vol. 110, pp. 58–71, 2018.

[20] C. Fleming and N. G. Leveson, "Improving hazard analysis and certification of integrated modular avionics," *Journal of Aerospace Information Systems*, vol. 11, pp. 397–411, 2014.

[21] Y. Wang, L. Wang, J. Hu, and Y. Zhou, "Modeling and analysis of IMA inter-partition communication safety requirement based on STPA," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS),*, pp. 284–287, Beijing, China, November 2018.

[22] R. Sun and D. Zhong, "Using STPA thinking to help convert natural language into finite automaton," http://psas.scripts.mit.edu/home/wpcontent/uploads/2013/04/01_Sun_Zhong_Using_STPA_convert_natural_language_finite_automaton.pdf.

[23] A. Abdulkhaleq and S. Wagner, "XSTAMPP: an eXtensible STAMP platform as tool support for safety engineering," in *2015 STAMP Workshop*, MIT, Boston, USA, 2015.

[24] A. L. Dakwat and E. Villani, "System safety assessment based on STPA and model checking," *Safety Science*, vol. 109, pp. 130–143, 2018.

[25] A. David, K. Larsen, A. Legay, M. Mikučionis, and D. Poulsen, "Uppaal SMC tutorial," *International Journal on Software Tools for Technology Transfer*, vol. 17, no. 4, pp. 397–415, 2015.

[26] L. Nigro and P. F. Sciammarella, "Statistical model checking of distributed real-time actor systems," in *2017 IEEE/ACM 21st International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pp. 1–8, Rome, Italy, January 2017.

[27] Y. Lu and M. Sun, "Modeling and verification of IEEE 802.11i security protocol in UPPAAL for Internet of Things," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, pp. 1619–1636, 2018.

[28] B. Xu, Q. Li, T. Guo, and D. Du, "A scenario-based approach for formal modelling and verification of safety properties in automated driving," *IEEE Access*, vol. 7, pp. 140566–140587, 2019.