Hindawi
International Journal of Aerospace Engineering
Volume 2018, Article ID 9358461, 11 pages
https://doi.org/10.1155/2018/9358461

Research Article

Research on the Reliability Analysis of the Integrated Modular Avionics System Based on the AADL Error Model

Peng Wang[1,2] **Changxiao Zhao**[1,3] **and Fang Yan**[1]

[1]*Civil Aircraft Airworthiness and Repair Key Laboratory of Tianjin, Civil Aviation University of China, Tianjin 300300, China*
[2]*Key Laboratory of Civil Aircraft Airworthiness Technology, Civil Aviation Administration of China, Tianjin 300300, China*
[3]*School of Airworthiness, Civil Aviation University of China, Tianjin 300300, China*

Correspondence should be addressed to Changxiao Zhao; cxzhao@cauc.edu.cn

Received 11 November 2017; Accepted 22 January 2018; Published 27 March 2018

Academic Editor: Kenneth M. Sobel

In recent years, the integrated modular avionics (IMA) concept has been introduced to replace the traditional federated avionics. Different avionics functions are hosted in a shared IMA platform, and IMA adopts partition technologies to provide a logical isolation among different functions. The IMA architecture can provide more sophisticated and powerful avionics functionality; meanwhile, the failure propagation patterns in IMA are more complex. The feature of resource sharing introduces some unintended interconnections among different functions, which makes the failure propagation modes more complex. Therefore, this paper proposes an architecture analysis and design language- (AADL-) based method to establish the reliability model of IMA platform. The single software and hardware error behavior in IMA system is modeled. The corresponding AADL error model of failure propagation among components, between software and hardware, is given. Finally, the display function of IMA platform is taken as an example to illustrate the effectiveness of the proposed method.

1. Introduction

As an important development direction of future large aircraft avionics system, IMA completes the real-time processing and information exchange task in navigation, communication, monitoring, and flight management through the integrated technology of avionics system [1], to ensure the flight safety effectively.

IMA is a safety critical system of civil aircraft. It uses resource sharing, data fusion, and restoration reconfiguration technology, which makes IMA highly complex and brings great challenges to IMA safety and reliability assessment. In addition, the failure propagation mechanism of IMA is so complex that the traditional reliability analysis method is not applicable to solve the problems of IMA architecture reliability assessment. Therefore, it is of great significance to study the reliability assessment method and to complete the work of IMA architecture reliability assessment.

To solve these problems, this paper introduces the reliability analysis technology based on AADL error model [2]. The AADL error model is bound to the IMA platform architecture on the basis of the error behavior and error propagation modeling of IMA platform to complete the reliability modeling. We take the display function as an example to make a comparison and analysis on different IMA architectures of its availability to draw conclusions on quantitative analysis.

2. AADL Error Model Overview

AADL uses the architecture design and analysis language, which supports the text or graphical modeling in three levels of hardware, software, and system [3]. It is used to complete the description and analysis of software and hardware in the real-time system and has been widely used in the flight control system and avionics system. In order to further extend the description ability of AADL language, SAE issued the AS5506/1 in November 2006, which includes the error model and the behavior model annex, so that the AADL has its unique syntax and semantics. For the error behavior of IMA platform, error model annex is analyzed in detail.

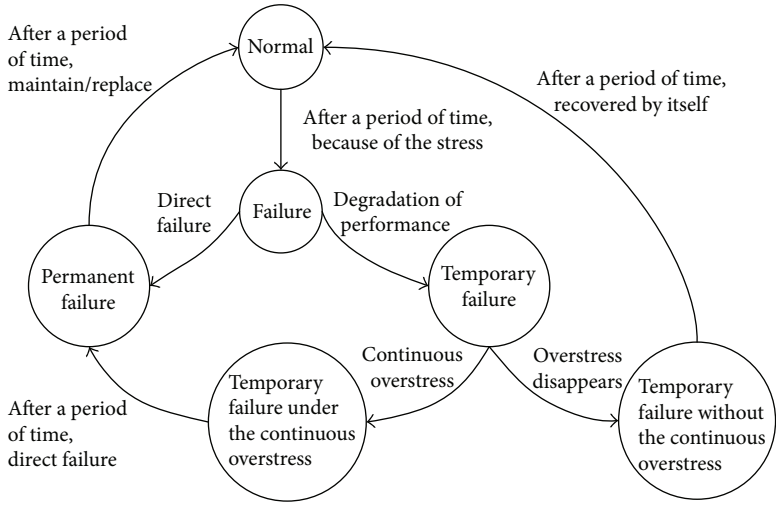

Figure 1: Transition behavior of IMA hardware error states.

The error model annex [4] is a state machine that can be associated with an AADL component or connection, to describe system errors, error behavior, and error propagation [5]. Including

(1) error model type: describing a set of error states, error events, and error propagation; specifying the probability of occurrence of error events and error propagation by property;

(2) error model implementation: a more detailed error model is defined, including the transition between error states and the occurrence of random events, which are used for further quantitative analysis [6].

In summary, the principle of error model can be expressed under certain conditions; components will trigger the error events due to its failure, and the error events will trigger the transition between the related error states; at the same time, the running states between the components that interact with each other will influence mutually due to the existence of the error propagation.

3. Implementation of IMA Error Model

3.1. Error Behavior Modeling of IMA Single Component

3.1.1. Error Behavior Modeling of Single Hardware. General processing modules of IMA platform can be divided into two parts: hardware and software; the software performs the related functions by loading itself into the hardware. First, considering only the single hardware error behavior model, we assume that the hardware of the general processing module is in the normal working state at the initial time; after running for some time, the hardware may fail because of the stress, such as temperature stress and electromagnetic stress. The hardware may be able to self-recover after the external stress disappears, that is, the failure lasts only a short time and only to reduce the hardware performance, which is called temporary failure [7]; however, if the external stress continues to be maintained, after a certain period of time, it will lead to permanent failure of the hardware, and it cannot recover anymore. This transition behavior of hardware error states is shown in Figure 1.

The AADL annex sublanguage is used to describe the above hardware error behavior, as shown in Figure 2. The hardware error model is divided into two parts, which are hardware error model type and error model implementation. The error model type contains 6 error states (including initial state) and 8 error events; the hardware error model implementation contains all of the transition rules between the error states, and the time occurrence or probability of the error events are described. The corresponding model meaning is shown in Figure 2 notes.

3.1.2. Error Behavior Modeling of Single Software. We assume that the software of the general processing module is in normal state at the initial time; after a period of operation, the software will be in failure due to the specific input (for the inevitable defects in the design of the complex software, the harsh design assurance level can only reduce the number of defects). Because of the fault-tolerance design of IMA software, the software will not lose its function immediately, and IMA will detect the failure and determine the failure type. If the failure is temporary, it needs to be further divided and determined whether the failure can be removed; if the failure can be removed, after a period of processing, the software can be restored to the normal state; if the failure cannot be removed, after a long time, this temporary failure will become permanent failure, and the software can be restored only through restarting operation. Figure 3 describes the transition rules between software error states.

The AADL annex sublanguage is used to describe the above software error behavior, as shown in Figure 4. The software error model is also divided into two parts, which are software error model type and error model implementation. The error model type contains 7 error states (including initial state) and 9 error events; the software error model implementation contains all of the transition rules between the error states, and the time occurrence or probability of the error

```
error model for hardware//error model type
  features//error mode features
  errorfree:initial error state;//initial state
  failed:error state;//failure state
  termporary_failed:error state;//failure state
  continual_temporary_failed:error state;//failure state
  dissappeared_temporary_failed:error state;//failure state
  overstressed:error event;//failure state
  direct_damage:error event;//failure state
  performance_degradation:error event;/failure state
  continual_overstess:error event;//failure state
  disappeared_overstess:error event;//failure state
  time_direct_damage:error event;failure state
  repaire:error event;//failure state
  recovery:error event;//failure state
end for hardware;//error model type end

error model implementation for hardware.general//error model implementation
  transitions//states transitions explanation
  errorfree-[overstress]->failed;
  failed-[direct_damage]->permanent_failed;
  failed-[performance_degradation]->temporary_failed;
  permanent_failed-[repaire]->errorfree;
  temporary_failed-[continual_overstress]->continual_temporary_failed;
  temporary_failed-[disappeared_overstress]->disappeared_temporary_failed;
  continual_temporary_failed-[time_direct_damage]->permanent_failed;
  disappeared_temporary_failed-[recovery]->errorfree;
  properties//random events properties explaination
  occurence=>poisson os applies to overstress;
  occurence=>fixed dd applieds to direct_damage;
  occurence=>fixed 1-dd applies to performance_degradation;
  occurence=>fixed co applies to continual_overstress;
  occurence=>fixed 1-co applies to disappeared_overstress;
  occurence=>poisson tdd applies to time_direct_damage;
  occurence=>poisson lambda applies to repair;
  occurence=>poisson miui applies to recovery;
end for hardware.general;//error model implementation end
```

Figure 2: Single hardware error model.

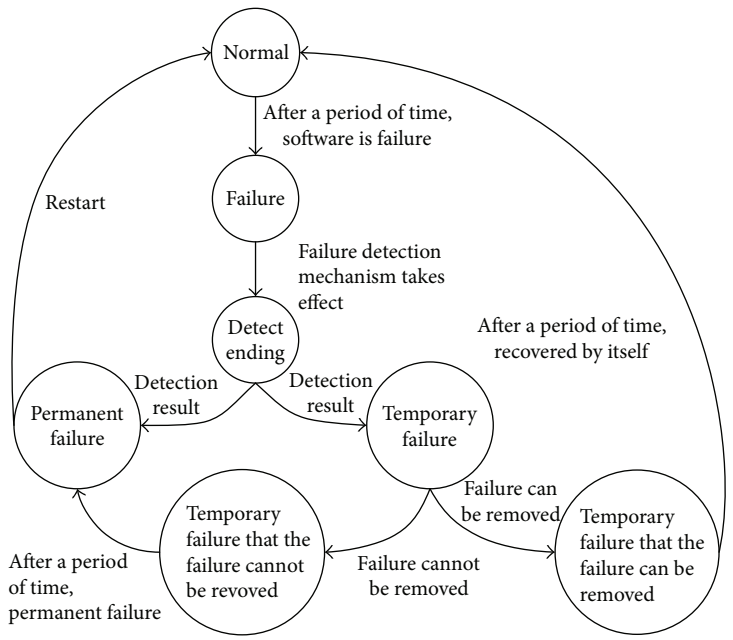

Figure 3: Transition behavior of IMA software error states.

```
error model for software//error model type
   features//error mode features
   errorfree:initial error state;//initial state
   failed:error state;//failure state
   detection_end:error state;//failure state
   permanent_failed:error state;//failure state
   temporary_failed:error state;//failure state
   unremovable_temporary_failed:error state;//failure state
   removable_temporary_failed:error state;//failure state
   fail:error event;//failure state
   detection:error event;//failure state
   direct_damage:error event;//failure state
   performance_degradation:error event;//failure state
   unremovable:error event;//failure state
   removable:error event;//failure state
   time_damage:error event;//failure state
   restart:error event;//failure state
   recovery:error event;//failure state
end for software;//error model type end

error model implementation for software.general//error model implementation
   transitions//states transitions explaination
   errorfree-[fail]->failed;
   failed-[detection]->detection_end;
   detection_end-[direct_damage]->permanent_failed;
   detection_end-[performance_degradation]->temporary_failed;
   permanent_failed-[restart]->errorfree;
   temporary_failed-[unremovable]->unremovable_temporary_failed;
   temporary_failed-[removable]->removable_temporary_failed;
   unremovable_temporary_failed-[time_damage]->permanent_failed;
   removable_temporary_failed-[recovery]->errorfree;
   properties//random events properties explaination
   occurrence=>poisson os applies to fail;
   occurrence=>poisson os applies to detection;
   occurrence=>fixed dd applieds to direct_damage;
   occurrence=>fixed 1-dd applies to performance_degradation;
   occurrence=>fixed phi applies to unremovable;
   occurrence=>fixed 1-phi applies to removable;
   occurrence=>poisson mu applies to restart;
   occurrence=>poisson td applies to time_damage;
   occurrence=>poisson theta applies to recovery;
end for software.general;//error model implementation end
```

Figure 4: Single software error model.

events are described. The corresponding model meaning is shown in Figure 4 notes.

3.2. Error Propagation Modeling of IMA

3.2.1. Internal Hardware/Software Error Propagation Modeling for Individual Components. The software runs on the hardware, so the hardware failure will affect the normal operation of the software. It specifically includes three cases:

(1) The stress on hardware can disappear, and the hardware is in temporary failure; it depends on whether the software is in a normal state. If the software is in normal state, after a period of operation, the software will be in failure under the influence of the hardware failure; if the software itself is in failure now, after a period of operation, the software may be in permanent failure or temporary failure because of the hardware failure, which depends entirely on the hardware failure type.

(2) Hardware is in permanent failure, which will lead the software to stop immediately, and now the software can be recovered only by restarting after the hardware is being back to normal.

(3) As for the impact of software on hardware, we only consider the permanent failure of software that will occur, which will lead the hardware to transfer from the normal state to a temporary failure state where the stress can disappear. In other cases, the software has no effect on the hardware.

The interaction between hardware and software [8] is shown in Figure 5. The AADL annex sublanguage is used to model the above error output and input behavior of software, as shown in Figures 6 and 7.

3.2.2. Error Propagation Modeling between IMA Components. IMA uses the partition method to divide each application software or function into a single partition, so that each partition has its own storage space and time gap and

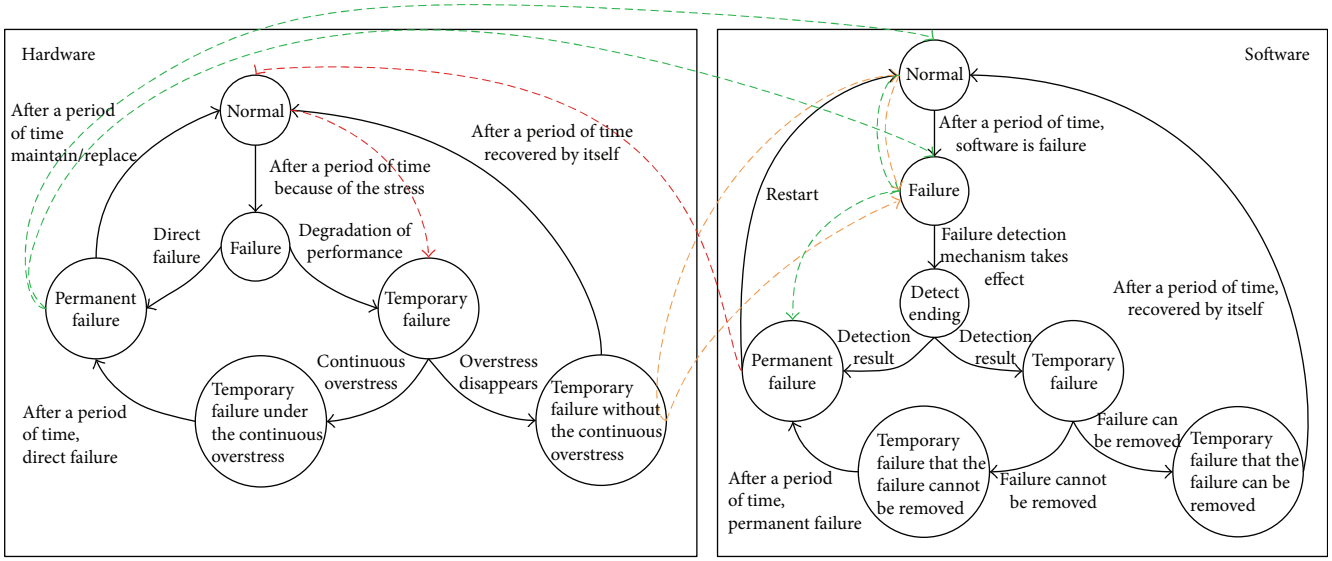

Figure 5: Interaction between software and hardware.

```
error model for software//error model type
   features//error mode features
   [...]
   (+)temporary_failed:error state;//failure state
   (+)reboot_state:error state;//failure state
   [...]
   (+)to_permanent_failed:error state;//failure state
   (+)to_temporary_failed:error state;//failure state
   (+)sw_permanent_failed_outflow:output error propagation;// output failure
   (+)disapeared_temporary_failed_outflow:input error propagation;// input failure
   (+)hw_permanent_failed_outflow:input error propagation;// input failure
   (+)reboot:input error propagation;// input failure
end for software;//error model type end
error model implementation for software.general//error model implementation
   transitions//states transitions explaination
   [...]
   (+)permanent_failed-[out_sw_permanent_failed_outflow]->permanent_failed;// output failure and state transfer
   (+)errorfree-[in disappeared_temporary_failed_flow]->failed;// input failure and state transfer
   (+)failed-[in disappeared_temporary_failed_outflow]->temporary_failed_sw;// input failure and state transfer
   (+)temporary_failed_sw-[in to_permanet_failed]->permanent_failed;//input failure and state transfer
   (+)temporary_failed_sw-[in to_temporary_failed]->temporary_failed;//input failure and state transfer
   (+)errorfree-[in hw_permanent_failed_outflow]->reboot_state;/ input failure and state transfer
   (+)failed-[in hw_permanent_failed_outflow]->reboot_state;// input failure and state transfer
   (+)permanent_failed-[in hw_permanent_failed_outflow]->reboot_state;// input failure and state transfer
   (+)unremovable_temporary_failed-[in hw_permanent_failed_outflow]->reboot_state;// input failure and state transfer
   (+)removable_temporary_failed-[in hw_permanent_failed_outflow]->reboot_state;/ /input failure and state transfer
   (+)reboot_state-[in reboot]->errorfree;//input failure and state transfer
   properties//random events properties explaination
   [...]
   (+)occurence=>poisson spfo applies to sw_permanent_failed_outflow;
   (+)occurence=>fixed 1 applieds to disappeared_temporary_failed_outflow;
   (+)occurence=>fixed tpf applies to permanent_failed;
   (+)occurence=>fixed 1-tpf applies to temporary_failed;
   (+)occurence=>fixed 1 applies to hw_permanent_failed_outflow;
   (+)occurence=>fixed 1 applies to reboot;
end for software general;//error model implementation end
```

Figure 6: Error output and input model of software.

ensures that they do not have the influence on each other. However, because of the needs for communication between applications, the errors may be transferred across different partitions during the data transmission and may damage the partition [9].

All of the message communication between partitions are completed through channels, that is, one message source transfers from a partition to one or more destination partitions through channels. The channel defines a logical connection from one source to another or multiple destinations, and the source and destination may belong to different partitions. In the initial system configuration, we need to define the message type, port, and channel. The error propagation simplified model of IMA is shown in Figure 8.

```
error model for hardware//error model type
  features//error mode features
  [...]
  (+)disappeared_temporary_failed_ovverflow:output error propagation;// output failure
  (+)hw_permanent_failed_outflow:output error propagation;// output failure
  (+)reboot:output error propagation;// output failure
  (+)sw_permanent_failed_outflow:input error propagation;// input failure
end for software;//error model type end

error model implementation for hardware.general// error model implementation
  transitions//states transitions explaination
  [...]
  (+)errorfree-[in sw_permanent_failed_outflow]->disappeared_temporary_failed;
  //input failure and state transfer
  (+)disappeared_temporary_failed-[out disappeared_temporary_failed_outflow]->disappeared_temporary_failed;
  //input failure and state transfer
  (+)permanent_failed-[out hw_permanent_failed_outflow]->permanent_failed;
  //output failure and state transfer
  (+)errorfree-[out reboot]->errorfree;
  //output failure and state transfer
  properties//random events properties explaination
  [...]
  (+)occurence=>fixed dtfo applieds to disappeared_temporary_failed_outflow;
  (+)occurence=>fixed 1 applies to hw_permanent_failed_outflow;
  (+)occurence=>poisson r applies to reboot;
  (+)occurence=>fixed 1 applies to sw_permanent_failed_outflow;
end for hardware.general;//error model implementation end
```

Figure 7: Error output and input model of hardware.

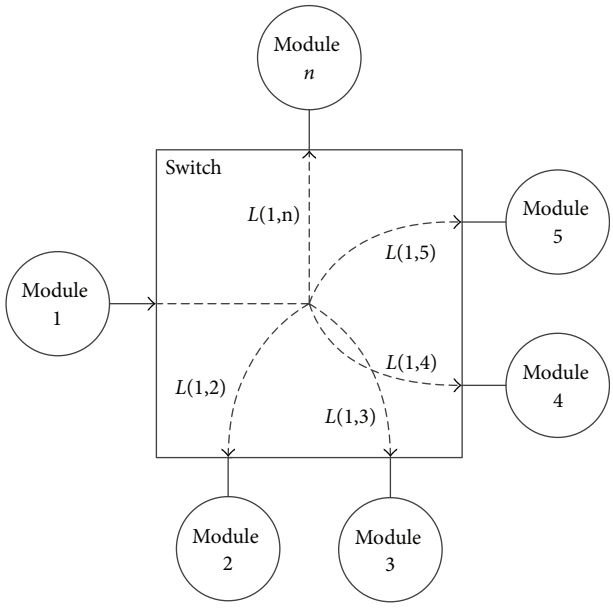

Figure 8: Error propagation simplified model of IMA.

After the module 1 fails, IMA will deal with it with the corresponding failure handling mechanism (such as failure detection and failure limit). We consider two factors: on the one hand, we assume that the module has 100% failure detection capability, namely, the failure must be detected after it occurred, but it does not have 100% failure limit capability, namely, the error may be transferred through the established channel after it happened; on the other hand, we consider the correctness of the channel configuration, for example, when module 1 transfers the data to module 2, the connection L(1,2) should be connected; however, because of the incorrect configuration of the switch configuration table resulting in that the connection L(1,3) and/or L(1,4) and/or L(1,n) are connected, while the connection L(1,2) is unconnected. Thus, considering these two factors, the error propagation probability from module m to module n is

$$P(m,n) = [1 - r(m)] \times L(m,n). \quad (1)$$

In the above formula, $r(m)$ is the failure limit capability of module m, that is, the ability that the module can limit the error to the area without transferring it out. $0 \le r(m) \le 1$; when $r(m) = 0$, the error must be transferred out; when $r(m) = 1$, the error cannot be transferred out. $L(m,n)$ is the probability of connecting the channel from module m to n; obviously, $0 \le L(m,n) \le 1$. Generally speaking, the reliability of AFDX network is so high that the value of $L(m,n)$ will be close to 1, when the data is transferred from module m to n.

Based on the above analysis, modeling the error propagation between IMA components, here, we only show the error transition from one component to another, and the rest is analogous. The error propagation behavior is shown in Figure 9.

The AADL error model annex language is used to describe the error propagation behavior between two components, as shown in Figures 10 and 11. Component A and component B describe their error output and input by matching the out-in names.

4. Verification Case of Safety Analysis on Display Function of IMA

The display function of IMA platform is taken as an example to illustrate the effectiveness of the proposed method. Firstly, the system architecture and error model

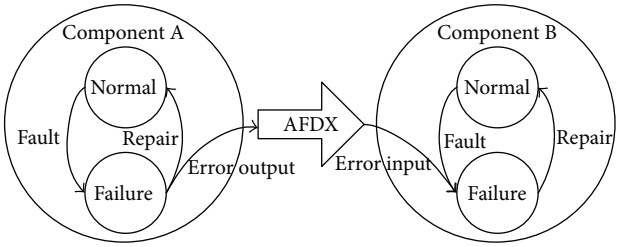

Figure 9: Error propagation behavior between components.

```
error model A//error model type
  features//error mode features
  [...]
  (+)error_propagation:output error propagation:// output failure
end A://error model type end

error model implementation A.general//error model implementation
  transitions//states transitions explanation
  [...]
  (+)failed-[output error_propogation]-failed;// output and transfer
  properties//random events properties explaination
  (+)occurrence=>fixed ep applieds to error_propagation;
end A.general;//error model implementation end
```

Figure 10: Error output model of component A.

```
error model B//error model type
  features//error mode features
  [...]
  (+)error_propagation:input error propagation;// input failure
end B;//error model type end

error model implementation B.general//error model implementation
  transitions//states transitions explanation
  [...]
  (+)errorfree-[input error_propogation]-failed:// input and transfer
  properties//random events properties explaination
  (+)occurrence=>fixed 1 applieds to error_propagation;
end B.general;//error model implementation end
```

Figure 11: Error input model of component B.

is constructed by AADL, and then the AADL model is transferred to GSPN model according to certain grammatical rules [10]. This method is shown in Figure 12. Finally, the availability of different architectures for display function is compared.

4.1. Reliability Modeling of Display Function. With the improvement of aircraft performance, the display control system of aircraft is gradually integrated. Cockpit display technology has undergone a long process of development from the mechanical to electronic, from dedicated devices to shared resources, and from the separated to the integrated. The typical architecture of basic integrated cockpit display system [11] is shown in Figure 13.

Sensors are distributed in various parts of the aircraft, sensing the measured information and converting the information into output signals according to certain rules, and then outputting these signals; data processing module is used to determine the display working mode and components of

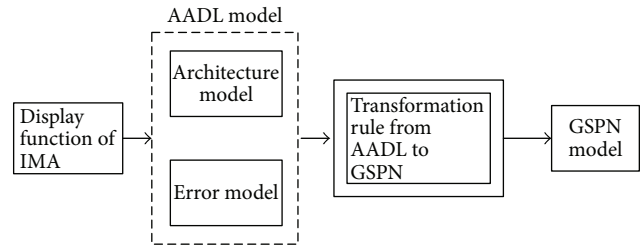

Figure 12: The experiment method.

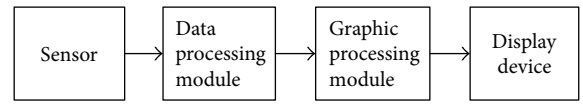

Figure 13: Architecture of basic integrated cockpit display system.

display function, changing the data information into graphic data and instructions, to display the control task scheduling; graphic processing module is used to construct characters and graphics; display devices include various displays and support units for displaying the flight information, engine working states, and so on.

Based on the above architecture, in fact, the integrated cockpit display system realizes the fault-tolerant property through many display devices, a lot of graphic processing modules and backup sensors. The typical architecture of it is shown in Figure 14.

There is a set of sensors on each side of the aircraft to collect the flight data, and they are backups to each other. As long as a set of sensors can work normally, data processing modules can receive the normal data. Then, data will be transferred to two graphic processing modules after being processed by the data processing module, and the two graphic processing modules can adopt the cold/hot backup configuration. After being processed by the graphic processing modules, images will be shown on the display. From the overall architecture, the same software and hardware configuration is used in both left and right display function, but tasks are different. They are working independently and backup to each other; once one side fails, the other side will take over its mission, thereby improving the reliability of the aircraft [12].

On the basis of mastering the architecture of IMA display function, the error models of basic display function and bilateral fault-tolerant display function are established, respectively, by using the AADL. Simplify every component according to the modeling method which is described in Section 3.2.2, that is, only consider the normal and failure states, regardless of the remaining error states. The error model of the components is bound to the architecture model to construct the AADL reliability model of the display function. The graphical representation of AADL reliability models is shown in Figures 15 and 16.

In Figure 15, when sensor S fails, the error will transfer along the data chain; thus, the data processing module will be influenced; once the data processing module fails, no matter how it fails (it fails by itself or affected by the failure of the sensor), the error will transfer along the data chain

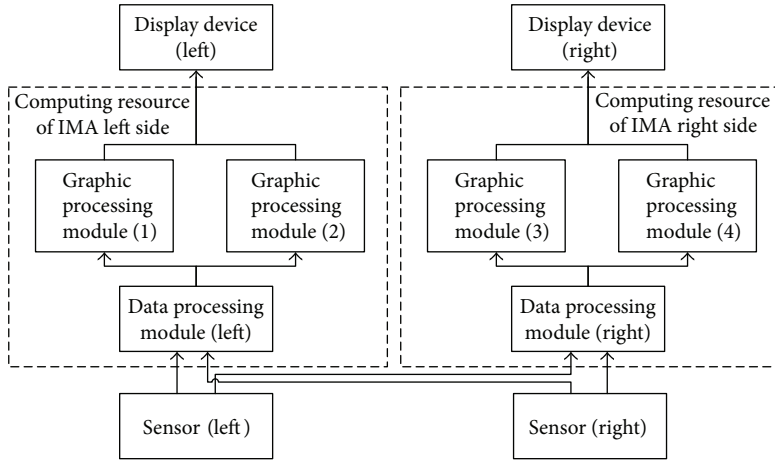

Figure 14: Architecture of bilateral tolerant display function.

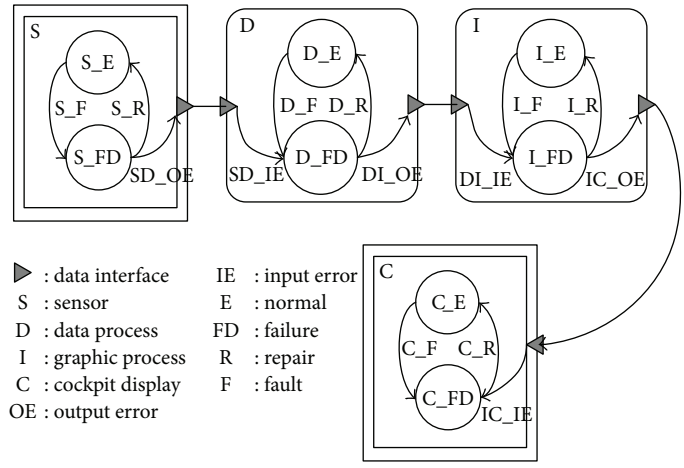

Figure 15: AADL reliability model of basic cockpit display system.

continuously and will make influence on graphic processing module; similarly, the graphic processing module and display device will be influenced by the transferred errors; finally, the cockpit display function will be influenced.

In Figure 16, sensors "S_1" and "S_2" are backups to each other; they are both in working mode and both make output to the system; only when both of them fail, the error will transfer along the data chain, then affecting the data processing module; the graphic processing module uses a hot backup architecture, namely, "I_1" and "I_3" are in working mode and make output to the system in initial conditions and "I_2" and "I_4" are also in working mode, but they do not make output to the system; once "I_1" or "I_3" fails, graphic processing module will change to hot backup mode immediately, and "I_2" or "I_4" will make output to the system. Therefore, only when both of two graphic processing modules fail, the error will transfer along the data chain and affect the cockpit display device.

4.2. Comparative Analysis of Availability of Different Architectures for Display Function. The design of IMA is an iterative process. As a necessary part of this process, the safety assessment is also an iterative process [13]. We should consider not only the cost but also the availability, so as to find a balance between them. The availability of the following ten cases is studied in detail for both the basic and the bilateral fault-tolerant display function architecture, as shown in Tables 1 and 2.

On the basis of establishing the AADL reliability model, we use the GSPN (general stochastic Petri net) [14] to analyze and compute the availability of the display function under the steady states. The availability is the proportion of time that a system is in a functioning condition. Firstly, we need to set the changeable parameters. Set failure probability of the display system components to 2E-3 per flight hour and complete repair once an hour and then further analyze all of the existent states and capture the availability of display function under the steady states. Results can be seen in Tables 3 and 4.

We can get the following conclusions from the above results: the availability of bilateral tolerant architectures is higher than that of the basic architectures under the steady states. When sensors are in the same architecture, the steady-state availability of the hot backup of graphic processing

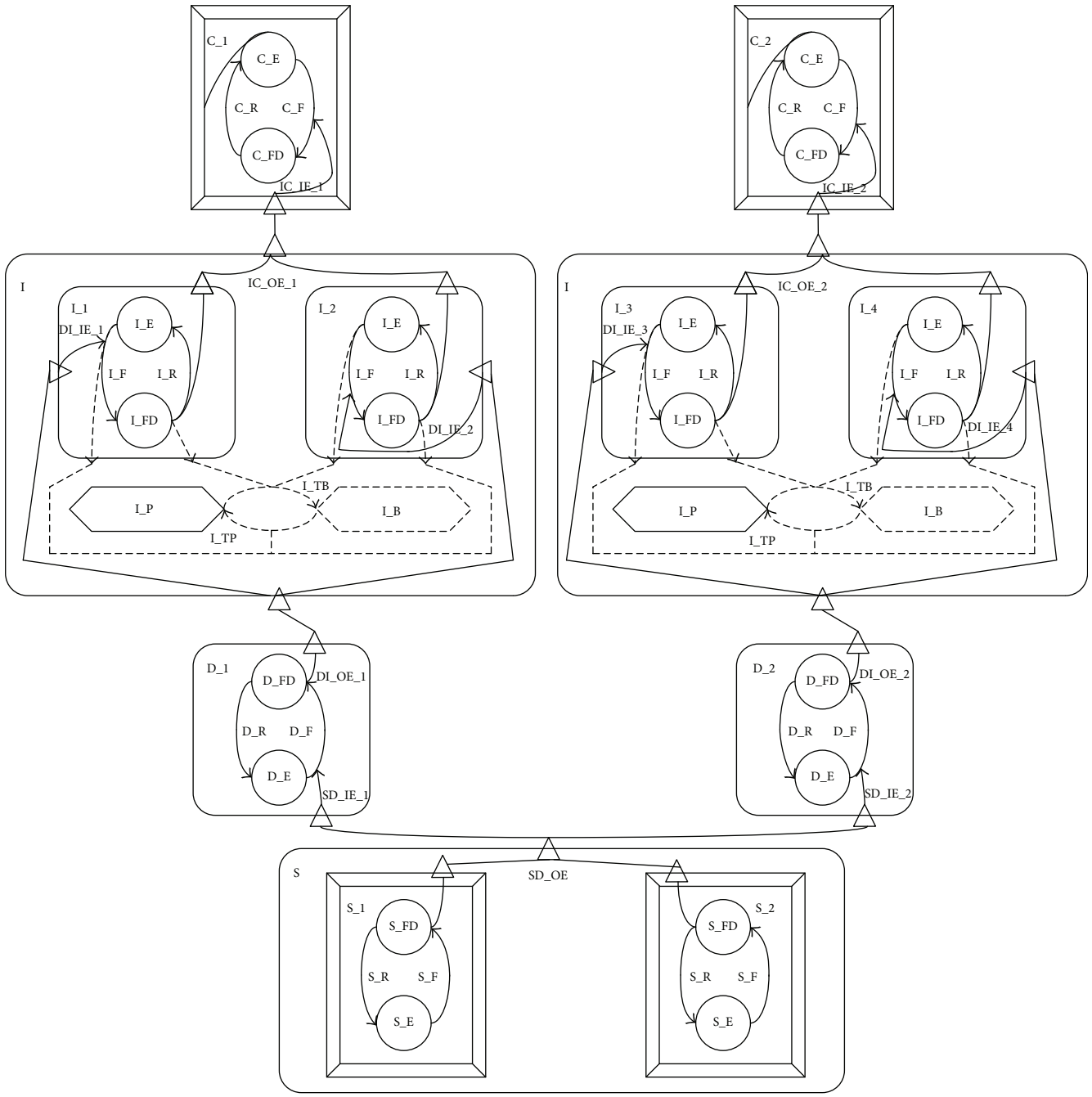

Figure 16: AADL reliability model of bilateral tolerant display function.

Table 1: Design of different architectures for basic display function.

Architecture	Sensor	Graphic processing module
1	No backup	No backup
2	No backup	Hot backup
3	Mutual backup	No backup
4	Mutual backup	Hot backup
5	Nonjudgement mutual backup	No backup
6	Nonjudgement mutual backup	Hot backup

Table 2: Design of different architectures for bilateral tolerant display function.

Architecture	Sensor	Graphic processing module
7	Mutual backup	No backup
8	Mutual backup	Hot backup
9	Nonjudgement mutual backup	No backup
10	Nonjudgement mutual backup	Hot backup

Table 3: Availability for different architectures of basic display function under the steady states.

	No backup of graphic processing module	Hot backup of graphic processing module
No backup of sensor	0.98025	0.98611
Mutual backup of sensor	0.98807	0.99300
Nonjudgement mutual backup of sensor	0.97249	0.97925

Table 4: Availability for different architectures of bilateral tolerant display function under the steady states.

	No backup of graphic processing module	Hot backup of graphic processing module
Mutual backup of sensor	0.99985	0.99995
Nonjudgement mutual backup of sensor	0.99925	0.99957

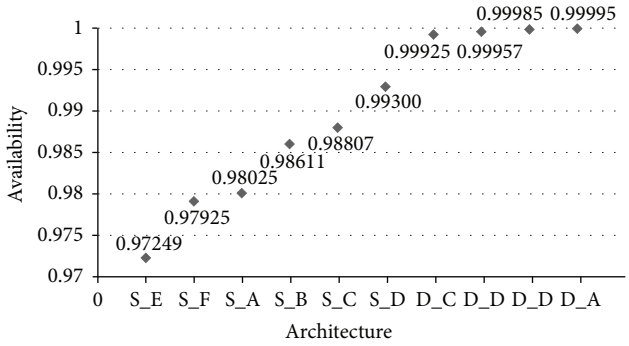

Figure 17: Availability comparison of different display function architecture.

modules is higher than that of nonbackup; when graphic processing modules are in the same architecture, the steady-state availability of the mutual backup of sensors is higher than nonjudgment mutual backup and nonbackup of sensors. The availability of ten architectures is shown in Figure 17. It should be noted that the hot backup will increase the cost; thus, the architecture of lower cost can be chosen when it meets the reliability needs.

In summary, when single component is under the pressure of failure, for the actual system which contains thousands of components, the key way to keep it operating reliably is fault limit and backup. Therefore, if the cost is limited, we should use these two ways to ensure the reliability of the system.

5. Conclusion

This paper proposed a reliability modeling method for IMA based on AADL error model, which effectively solves the problem of complex error propagation mechanism due to the highly complex and resource-sharing characteristics of IMA. It accurately describes the dynamic characteristics of IMA fault and makes up the defects of static reliability assessment method. At the same time, the establishment of AADL reliability model based on system architecture ensures the synchronization of IMA reliability model and IMA design and avoids the traditional problems of "postdesign evaluation." By analyzing the availability of display function under different architecture, we proposed that the reasonable use the fault limit and backup mechanism can improve the system reliability effectively under the condition of cost allowable. It is meaningful to safety assessment work of complex electronic hardware in avionics system.

Conflicts of Interest

The authors declare that they have no competing interests.

Acknowledgments

This paper was funded by joint fund of the National Natural Science Foundation of China and the Civil Aviation Administration of China (no. U1533105). The authors would like to thank Ms. Fan Zhang and Ms. Peipei Xing for their preliminary work.

References

[1] Q. Y. Li, "Application analysis of SMP and MPP architecture in integrated avionics system," *Electronics Word*, vol. 2013, no. 9, pp. 23-24, 2013.

[2] SAE International, "Architecture analysis and design language (AADL) annex volume," Tech. Rep. AS5506/1, 2006.

[3] P. H. Feiler and D. P. Gluch, *Model-Based Engineering with AADL-an Introduction to the SAE Architecture Analysis & Design Language*, Pearson Schweiz Ag, 2012.

[4] K. S. Kushal, M. 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.

[5] P. Feiler and A. Rugina, *Dependability Modeling with the Architecture Analysis & Design Language (AADL)*, ARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, 2007.

[6] P. H. Feiler, J. Hansson, and D. de Niz, "System architecture virtual integration: an industrial case study," Technical Report CMU/SEI-2009-TR-017, CMU, 2009.

[7] C. R. Spitzer, "Digital avionics technology (2): avionics development and implementation," in *Translation*, W. T. Xie, Ed., p. 119, Aviation Industry Publishing Company, Beijing, 2010.

[8] J. Shi, Y. X. Meng, and S. P. Wang, "Reliability and safety analysis of redundant vehicle management computer system," *Chinese Journal of Aeronautics*, vol. 26, no. 5, pp. 1290–1302, 2013.

[9] S. Park and G. Kwon, *Formal Verification for Inter-Partitions Communication of RTOS Supporting IMA//LNEE : Frontier and Innovation in Future Computing and Communications*, Springer, Netherlands, Berlin, 2014.

[10] P. Wang, F. Zhang, L. Dong, H. U. Jianbo, and C. Zhao, "Translation rules of fault extended model to probabilistic

checking model," *System Engineering and Electronics*, vol. 11, pp. 2501–2508, 2017.

[11] W. S. Niu, *Airborne Computer Technology*, Aviation Industry Publishing Company, Beijing, 2013.

[12] S. Liu and R. C. Lin, "Design and implementation of integrated cockpit display control system," *Modern Electronic Technology*, vol. 33, no. 15, pp. 160–162, 2010.

[13] Z. X. Xiu, *System Safety Design & Assessment in Civil Aircraft*, Shanghai Jiao Tong University, Shanghai, 2013.

[14] Y. T. Xu, Y. F. Yin, and J. Sun, "Real-time embedded software architecture modeling and reliability estimation based on time-extended petri net," *Acta Armamentarii*, vol. 36, no. 2, pp. 363–373, 2015.