A Safety Evaluation Method of IMA Dynamic Reconfiguration Process Based on CPN

Lei Liu¹, Wei Zhao¹, Zeyong Jiang² and Shihai Wang²,³

¹ Beijing Electro Mechanical Engineering Institute Beijing, 100074, P. R. China
liuleilarr@126.com; qiuhuabb@163.com
² School of Reliability and Systems Engineering, Beihang University, Beijing 100191, P. R. China
jiangzeyong0106@163.com; wangshihai@buaa.edu.cn
³ Science & Technology on Reliability & Environment Engineering Laboratory, Beihang University, Beijing 100191, P. R. China

Abstract. With the rapid development of integrated modular avionics (IMA), the dynamic reconfiguration of IMA provides great advantages in resource utilization and aircraft configuration and acts as a valid means for resource failure management. Due to the complexity of dynamic reconfiguration process, it is difficult to analyze and evaluate the safety of dynamic reconfiguration process. The Architecture Analysis and Design Language (AADL) is effective in modeling such process of the embedded real-time system. Colored Petri Net (CPN) has advantages of simulation and evaluation. Therefore, this paper model the IMA dynamic reconfiguration process based on AADL, and transfer it to CPN model to analyze and evaluate the safety of the process of dynamic reconfiguration. Finally, a case study is provided to indicate the effectiveness of the method.

1. Introduction
An avionic system is developed from a discrete one, to the federal one and to the integrated modular avionics (IMA) [1]. The system has more open and more complex architectures. The IMA system executes functions based on common functional modules (CFMs). The CFMs help to reduce the weight and size of a plane. In an IMA system, various software functions run on CFMs. The software system is highly integrated because of its complex structure. In this paper, dynamic reconfiguration refers to configuration changes conducted when failure occurs during flying. Dynamic reconfiguration can help in creating new backup areas to restart an application, which makes the plane more flexible and utilizes the hardware resources more effectively [2].

However, based on its complexity, it is difficult to analyze the IMA dynamic reconfiguration process. The Architecture Analysis and Design Language (AADL) could describes the system components [3] and also describes the system behaviors and other elements, such as mode and all types of annex [4]. Modes can represent different configuration states of a system or component when an event triggers a mode change [5]. All of these features make AADL a good method to describe the transition process of a system, for example, the dynamic reconfiguration of IMA. A Petri Net, a graphical tool to describe the dynamic system [6], has the elements of tokens, places, arcs, and transitions [7]. A Petri Net can represent the system states with places [8]. Transitions can describe the system behavior and changes [9]. Sending of tokens presents the dynamic activities of the system. Then, the Petri Net is extended to a colored Petri Net (CPN) [10], generalized stochastic Petri Net
(GSPN), to enhance its description ability so that it can simulate the dynamic process more effectively. In this study, we used the CPN to simulate the dynamic reconfiguration of IMA.

The rest of this paper is organized as follows. Section 2 provides a brief introduction about IMA dynamic reconfiguration. In section 3, AADL is applied to model the IMA dynamic reconfiguration process and CPN is proposed to evaluate the safety in IMA dynamic reconfiguration process. Finally, a case study is proposed to describe this method in detail in section 4. Conclusions of our study are provided in section 5.

2. IMA Dynamic Reconfiguration

The IMA system is a complex system that has more open architectures, more widespread integration, more integrated functions, and high coupling between modules. Many challenges also appear when reconfiguration occurs. Dynamic reconfiguration in this study pertains to software. Then, the architecture of IMA software is introduced in this paper. The IMA system includes the IMA core system and noncore equipment according to the ASAAC standard. The IMA core system contains several avionic racks. These racks contain CFMs and communication nets between them. Moreover, the racks have functional applications based on hardware, the operational system, and system management software.

A CFM provides computing power, net support, and power conversion for the IMA core system. The software system is divided into three layers—the module support layer (MSL), operating system layer (OSL), and application layer (AL). The MSL provides an interface for the above layer to access the resources and separates the operating system and the hardware platform. The OSL includes a real-time operation system and system management. Software application and application management are conducted on the AL. The general system management (GSM) in the OSL, which performs system management, configures and manages the system by the accessing the system blueprint. The GSM includes health management, fault management, configuration management, and safety management. Application management is part of system management and is operated based on the application. The software architecture is displayed in figure 1.

![IMA software architecture](image)

**Figure 1. IMA software architecture**

3. Method of Evaluation

1) Dynamic reconfiguration process

When one or several failures occur on a module of IMA, a health manager detects the failure and informs a fault manager to handle it. The fault manager can handle a series of failures under all types of mechanism. Then, the fault manager determines the type of failure to take actions to solve it, for example, closing dynamic reconfiguration, or reporting to the upper layer manager. If a failure causes dynamic reconfiguration, then the system stops the failure application and backups the data. The connections are destroyed. Then, the target module of reconfiguration is selected based on the functional and nonfunctional requirements such as minimizing the cost of communication. The next
3) Modeling of the dynamic reconfiguration process

A mode of a system can be associated with the logical configurations. Mode transitions imply that the configuration state changes from one to another. A system or a component has different static structures and properties in different modes. A property can describe task scheduling, real-time characteristics, communication, memory, etc. Then, modes at the system level represent the content of a system configuration. A system has its own modules, partitions, processors, and communication bus in each mode. Thus, the static structure of the system in one mode is built by ARINC 653 annex in AADL.

3) Transformation

The AADL model of IMA dynamic reconfiguration is effective in describing the system structure and complex reconfiguration process. Some analysis can be conducted in tools for AADL]. However, automatic simulation and analysis are not the strong points of AADL but fit for Petri Net. In this study, a place of the Petri Net represents a mode or a substate of the system. As a result, the structure of the system in a stable mode and thus cannot be expressed in detail. However, the detailed model of a system in AADL can help determine what resource a place is related to. The resource is one of the constraint conditions for analysis.

4. Case Study

For the IMA system in the case, a series of functional modules including navigation, display, communication, and integrated radio frequency sensors (IRFS) are integrated. The navigation module provides the place of the plane and guides the plane in a definition router. The module for an aircraft cockpit display provides the man–machine interface for a pilot. The communication module is responsible for the communication between an aircraft and a ground unit. IRFS integrates all the RF sensors in the aircraft for sending and receiving signals at all frequency ranges.

For simplification, an IMA system with four modules is modeled based on AADL. We denote each module with the first letter of its name—navigation module (N), display module (D), communication (C), and IRFS (I). There are several partitions on each module according to their functions. An application runs on a partition. Process refers to the application here. Moreover, it is assumed that there is one partition in module N and module D. Three partition are set up in the module I. The other two partitions are in module C. The application on each partition communicates with GSM to define the operation of connections and applications.

![Figure 2. Model of dynamic reconfiguration based on AADL.](image-url)
When the module N fails, the GSM detects the failure and starts the failure management. In this case, the failure causes process 1 to break down and the system reconfiguration. Thus, dynamic reconfiguration is triggered.

1) After data backup for the process 1, process 1 is shut down and the connections of process 1 in the module N are destroyed.

2) The system selects a proper module to establish a new partition to run process 1. The strategy for selecting the target module is not introduced here. The target module is module D in this case.

3) A new partition is created in the target module D. Moreover, new channels and connections are set up. Process 1 is reloaded and restarted on the new partition in module D.

The process is presented in figure 2.

Based on the transformation rules defined, the AADL model of dynamic reconfiguration is converted to CPN as shown in figure 3. The modes and states of the behavior annex are converted to places in CPN. Mode transitions and behavior annex transitions are converted to transitions in the CPN. Other resources such as memory and data are represented by the color set of tokens in places. The triggering condition and constraints are added to the CPN as guard functions for a transition.

![Figure 3. The CPN model of the dynamic reconfiguration process](image)

| No. | Original conditions | Simulation results | Analyzing |
|-----|---------------------|--------------------|-----------|
| 1   | Val mem_size=200.0  | Simulation finish well. | Memory constraints for system state is fulfilled. |
| 2   | Val mem_size=70.0   | T8 can’t be fired, simulation stop. | Memory constraints for system state don’t meet the requirements. |
| 3   | Time cost during the simulation beyond the real-time constraint | The time stamp '@+48' reveals the running time 48 beyond the limitation 30. | Real-time constraints for system state transition is not satisfied. |
| 4   | 1'(s, failure2)->initial mode and 1'failure1->failure | The simulation not running, T1 is not fired. | The system is in a fault propagation state and not fit for reconfiguration. |
| 5   | There is no 'w' sending to the place named Data | Value of guard function [p=w] is false. W is not fired. Simulation stop. | A demanding mark is not written to data. components, so the next state failed to sharing the data. |
The results are listed in table 1 after the simulation was conducted many times under different conditions. The precondition for each result was that the other constraint set in this net meets the requirements for finishing simulation except the condition pointed out in Table 1.

1) Condition 1: All the constraints are satisfied.
   The system model obtained when the simulation is conducted for 58 ms.

2) Condition 2: Constraint of the memory size is not satisfied.
   The simulation will stop running when it runs for 48 ms, because the guard function \([m \leq \text{mem\_size}]\) is not satisfied. The upper limit of the system memory size \(\text{mem\_size}\) is only 70 M, but the state needs to occupy a memory size of 70.1 M, so the simulation ends.

3) Condition 3: Real-time constraint for the system state transition is not satisfied.
   By comparing the time consumed and real-time requirements, it can be shown whether the real-time constraint is satisfied.

4) Condition 4: System state constraint for dynamic reconfiguration is not satisfied:
   When the system runs to transition T1, it can be judged by the guard function \([f1=\text{failure1 andalso I =}(s,\text{nofailure2})]\). If fault propagation occurs before system reconfiguration and other modules are affected, the reconfiguration scheme cannot be adopted. The reconfiguration process stops and cannot be conducted.

5) Condition 5: Ability constraint for sharing data resources is not satisfied.
   When the system performs an operation of a shared data resource, if the forward state backup cannot write to the data component, then the checking of the data component and last state are not triggered. Then, the process stops at a step of 15 ms.

5. Conclusions
The dynamic reconfiguration of IMA provides great advantages in resource utilization and aircraft configuration and acts as a valid means for resource failure management. Due to the complexity of dynamic reconfiguration process, it is difficult to analyze and evaluate the safety of dynamic reconfiguration process. In the paper, AADL is applied to model the IMA dynamic reconfiguration process and transfer AADL model to CPN model in order to simulate and analyze the dynamic reconfiguration process. It could help to solve some safety issues in IMA dynamic reconfiguration process.

6. Acknowledgments
This work was supported by the Defense Industrail Technology Development Program (JCKY2016204A102) and Foundation of No.61400020404.

7. References
[1] Jolliffe G 2005 Conf. on 24th Digital Avionics Systems Conference (Washington: USA) pp 1-14
[2] Ellis S 1997 J. Microprocess. Microsyst. 21 29-39
[3] Yang Z, Hu K, Ma D, Bodeveix J, Pi L and Talpin J 2013 J. J. Syst. Softw. 93 42-68
[4] Bozzano M, Cimatti A, Katoen J, Nguyen V, Noll T and Roveri M 2010 J. Comput. J. 54 754–775
[5] Zhang F, Zhao Y, Ma D and Niu W 2007 J. IEEE Access. 5 27421–27438
[6] Wu Y and Li S 2014 J. Comput. Technol. 24 88–91
[7] Kabir S and Papadopoulos Y 2019 J. Saf. Sci. 115 154–175
[8] Li Y, Chen Y, Tang N and Yang L 2016 Conf. on Prognostics and System Health Management Conference (Chengdu: China) pp 1-7
[9] Wieland C, Schmid O, Meiler M, Wachtel A and Linsler D. J. J. Power Sources. 190 34–39
[10] Li Z, Wang S, Zhao T and Liu B 2016 J. Chin. J. Aeronaut. 29 1027–1041