Application of AADL in Integrated Electronic Systems

Lei Liu\textsuperscript{1a}, Guangding Feng\textsuperscript{2b,}\* Weipeng Wang\textsuperscript{3c}

\textsuperscript{1} Beijing Electro-Mechanical Engineering Institute, Beijing 100074, China,
\textsuperscript{2} School of Reliability and Systems Engineering, Beihang University, Science and Technology on Reliability and Environmental Engineering Laboratory, Beijing, China,
\textsuperscript{3} Beijing Electro-Mechanical Engineering Institute, Beijing 100074, China,
\textsuperscript{a}liuleilarry@126.com, \textsuperscript{b}15313003532@buaa.edu.cn, \textsuperscript{c}hbbdwwp@163.com

Abstract. In this paper, we try to find a proper method to model the IMA (Integrated Electronic System) reconfiguration process, make an analysis on simulation, validation and certification of IMA platform. We proposed the analysis method on AADL (Architecture Analysis and Design Language) to modeling the components of IMA, and its error and behavior annex can help to describe the mechanism of reconfiguration, its ARINC653 annex can help to design safety-critical avionics systems. In addition, we gave the detailed process of IMA reconfiguration by steps. The whole process helps us to grasp how to apply AADL to the corresponding analysis of IMA, thus helping to improve and optimize the design.

1. Introduction

The AADL (Architecture Analysis and Design Language) standard development began with Meta H. Meta H is an architecture description language and non-commercialized support toolsets. The AADL is a component-based method that is greatly fit with the safety-critical systems analysis. The properties and requirements of the specialized hardware and software components can be extended or refined to model the system, such as the IMA. Using the AADL to model can ease system analysis before system implementation, thus helping to detect the defect of the system design. Now it is widely applied to the simulation, validation and verification of safety-critical systems. AADL can be used to describe the architecture of the system that software components map to the execution platform as well as the functional interfaces of the components (such as data input and output) and the performance-critical attributes of the components (such as real-time). AADL can also be used to describe how components interact with each other, such as how input and output ports of data are related and how application software components are located in execution platform components [1]. So an AADL description is made of components. By providing model operation modes and mode switching, AADL can also describe the dynamic behavior of the runtime architecture. Therefore, AADL is a modeling language that supports early analysis of the system's architecture through extensible markup, tool framework, and precisely defined semantics. Components are the core modeling concepts of AADL. Each component has a unique identifier. Each component kind can declare the component type and component implementation, and a component type can have multiple component implementations [2]. The component type describes the functional interface of the component and the component implementation describes the internal structure of the component. The components in AADL are divided into three categories: software components (data, thread, thread group, subprogram, process), execution platform components (memory, bus, processor, device, virtual processor, virtual bus), and composite components (system).
The component type provides features with the IMA. The inter-components communicate with each other according to the features (the interface specification). The component implementations all have their own component type. In the implementation of components, we will describe the call sequences, connections and subcomponents of it. AADL could match the properties with the AADL model, and it can greatly help us to understand the architecture of IMA.

2. AADL Modeling of IMA Reconfiguration

2.1. Reconfiguration

With the rapid development of IMA, the system structure has become more and more complex, and its scale has become larger. Its degrees of integration and complexity have been continuously enhanced, and the degree of coupling between components and subsystems has been continuously improved, resulting in an increase in the complexity of its interaction. And the possible failure modes are more and more diverse. The safety issues that come with it are also becoming more and more prominent [3].

The dynamic reconfiguration of the IMA system can make the aircraft recover quickly from the failure during the flight process. It plays a great role in enhancing flexibility and reducing hardware redundancy, and can greatly avoid the occurrence of failures. For those systems that no longer meet their performance or safety requirements after failure, reallocate the software components to produce a new hardware-software mapping which does meet the requirements [4]. That is, the reconfiguration of avionics systems refers to the reconfiguration of software applications on the hardware during system operation because hardware failures are irreversible and software failures can be repaired. Therefore, the reconfiguration we all have made refers to the software. IMA reconfiguration is a switching process from one configuration to another. To describe the single configuration and the switching process, we use the AADL to model static architecture and dynamic behavior of the system.

2.2. The Example of IMA Reconfiguration Process Analysis

![Diagram of the reconfiguration process]

**Figure 1. The Process of Reconfiguration**
As is shown in the above Figure 1, after module fails, the fault can trigger reconfiguration if it is detected, then the dynamic reconfiguration will occur. In the example shown in the figure above, the failure causes the partition (process) 1 to crash, causing system reconfiguration and the dynamic reconfiguration triggered [5]. The whole process of reconfiguration can be described as the three steps below.

1) First backup the running application process1, close the application, and destroy the connection;
2) The system selects the appropriate target module to create a new partition to restart application process1;
3) After the new partition is established, new connections and channels will be created, reload the application process1 and restart on the new partition.

The most important thing in this reconfiguration process is to make redundant backups, we need the redundant module and redundant internet communication to assign the different I/O. And fortunately, the IMA platforms consist of the common functional modules, that is, the modules can help achieve the function of another one [6]. So it is relatively easy to find the appropriate target module.

In the above figure, the mode is used to represent a configuration state during the dynamic reconstruction of the system. The initial mode is a state in which the system operates without failure. When a fault occurs, the system transitions to mode 1 indicating that module has failed. After reconfiguration, the system transitions to mode 2, representing the system running in a new, non-failure configuration state.

In the analysis process of reconfiguration, we have developed behavior annex, error annex and ARINC653 annex of AADL to refine the modeling process. Their specific use can be seen below.

The two modes use behavior annex to represent the details of mode transitions, a series of actions, triggers and conditions, such as data backup of process1, and the creation of partitions on module [7]. Defined attributes can be added in behavior attachments. And the improvement of the attachment gets the sub-state defined between mode 1 and mode 2. In this process, behavior attachments describe sub-state transitions and actions between modes in more detail. We can define the stop and reload of the application, connection creation, destruction, partition creation, and so on. The process of each action occurs is described by the transition, and the action connects two sub-states, that is, the state defined in the behavior attachment.

Behavioral annex language description examples shown in the figure 2:

```plaintext
annex behavior_specification(**
  composite state between All_WORK and M2_FAILURE state1
  states
  AllWork: initial state;
  Stop_Process: complete state;
  Destroy_Transition: complete state;
  New_Partition: complete state;
  Create_process: complete state;
  M2_failure: complete state;

  transitions
  AllWork-[M2_Failure]->Stop_Process;
  Stop_Process-[GLI_configuration_stopped]->Destroy_Transition;
  RealtimeProperty::ProcessTime=>10.0; (%destroy connections)
  );

  Destroy_Transition-[GLI_new_partition]->New_partition;
  RealtimeProperty::ProcessTime=>10.0; (%create the new partition)
  memory size=>12MB;
  );

  New_Partition-[GLI_load_configuration]->Create_process;
  RealtimeProperty::ProcessTime=>10.0; (%create process)
  memory size=>12MB;
  );

  Create_process-[GLI_run_configuration]->M2_failure;
  RealtimeProperty::ProcessTime=>10.0; (%running)
  );

end state1;
**);```

**Figure 2. Behavior Annex Modeling**
Behavioral annex describes the transition of the system's sub-states, and attributes are added to the transitions as a basis for further safety analysis. In addition, regarding the error annex that triggers reconfiguration, it is also modeled and used the error model annex to describe. It can be described as the figure 3.

![Error Annex Modeling](image)

**Figure 3. Error Annex Modeling**

The error model annex can be used to indicate the trigger condition caused by the fault. In the above figure 3, the occurrence of a fault event will cause the system to start dynamic reconstruction. The annex describes the system's change from an error-free initial state to an error state, declared as "error_free - [error_occurred] -> error_state." The transition in the error model annex triggers the system to begin reconfiguration [8]. The declaration statement is "Error1_trigger => self [detected_state] applies to mode_transition_event."

In general, when system changes its mode, a confirmed fault is detected, there will trigger the reconfiguration. A reconfiguration is defined as the transient activity between two ultimate system states/modes of the system. After the fault is detected, we will isolate the fault, call redundant modules and communications, and complete the reconstruction process according to all action sequences and pre-allocated time [9].

The mode describes the configuration and connection status of sub-components in a static structure. The transition from one mode to another may need to be applied to a series of configurations. The mode of the entire system describes the current system function or a state that isolates the fault hardware resources. The component type or implement of the different mode have different property value.

To improve the reliability of the reconfiguration strategies, now there is this study that proposes an approach to support the modeling and reliability analysis of reconfiguration in IMA based on AADL [10]. AADL provides a standard and sufficiently precise way of modeling the architecture of an embedded, real-time system such as IMA, and thus permitting analysis of its properties; facilitates the automation of code generation, system creation, and other development activities; and significantly reduces design and implementation defects.

When modeling static architecture, the ARINC653 annex of AADL plan out the reconfiguration path. It is the typical partition architecture. The component of AADL is corresponded with the part of ARINC653 Annex. The annex is the binding declaration of interface between application and operation system in specific time and space partition [11]. ARINC653 is an industrial standard that defines a set of services for the design of safety-critical avionics systems. Above all, AADL can well model the static partitioned architecture of the IMA system [12]. When modeling that, we used to use the process to represent the partition.

And in the whole process, we use the stratified modeling method. AADL modeling first creates a model component, creates the appropriate component type according to the type of module and adds the
function for the type of component created according to the requirement analysis. Based on this, AADL modeling creates the component type and component implementation of all modules of each component type. It is also necessary to create a special component - the component type and component implementation of the Containing System, to mark the boundaries of the entire embedded system. The next step is to fill in the system implementation, that is, to add sub-components for the components of the Containing System and model component system types and connect the sub-components and sub-components, sub-components and model component ports. The sub-components here include both the created model components and some declared package. The model components and system implementation complete the modeling of the top-level design. The hardware components and software components model the bottom-level design and map the top-level design model to the model of the execution platform components and application software components. When creating hardware components and software components, we need to create the component types firstly and add features of it, then create the corresponding component implementation. Hardware components also need to add the required bus access, create execution platform instances, connect instances, and connect model components and hardware components. Software components also need to create process components and thread instances, and add thread instances to the process components, then create instances of the process and add them to system-type components.

When modeling dynamic mode transition, that is dynamic reconfiguration, we will apply the behavior annex and error annex to represent the special state. Modes are states within a state machine abstraction. The source mode changes to destination mode by way of triggers. We map the initial mode in AADL to place with token of Petri net as the starting point during the analysis by the transition rules. We set the mode of “poweroff” to mark the terminal point of reconfiguration path. Petri net can be constructed from AADL model with those mapping rules to perform reliability analysis [13].

Using AADL to model IMA and its reconfigurations is very practical. In particular, some mechanism like mode, property and ARINC653 annex make possible the description of reconfigurable IMA. The process of IMA reconfiguration becomes very clear.

3. Simulation, Validation And Certification of IMA Platform based on AADL

Avionics systems have developed into more and more integrated architecture in the past few years. Now in the integrated architecture of IMA, one processor can host some applications. It would greatly reduce the weight, space, power consumption and the number of modules used in avionics platform. In that integrated architecture, the modules are interconnected and communicated through a deterministic network which generally is AFDX.

In the IMA platform, complexity of the interaction between different components is continuously increasing. The functions, management, and resources are continuously integrated. The configuration collection space and the modules are highly coupled. The issues surrounding IMA are numerous and complex. Therefore, the design complexity of an avionics system platform needs to be correspondingly improved. The changes in these developments entail a corresponding modeling of the system to simulate, validate and certificate avionics platform, thus supporting our subsequent works. Because the reconfigurations are known before the runtime, so we can model the system to simulate, validate and certificate. When we make those analyses, we assume that the sub-systems can be assessed independently and they are deterministic. We use the modeling method based on AADL to describe the complete avionics platform. Following that evolution, suppliers developed the ARINC 653 AADL annex which specifies space and time partitioning [14]. It can be described to the interface between applications and operating system as the binding declaration. To avoid making effect to each other, each application is enclosed in their own partition. So when identifying the fault, we need not to check others because it is isolated from each other. Also, the memory area and execution time window is independent. And the intra and inter-partitions communications is the same purpose. Otherwise, we perform a simulation using systemC kernel. As is shown in the figure 4, it takes the platform generate, the viewpoint(s) set, and applies the simulation scenario. Then we need to examine performances graphs or simulation traces to analyze the platform performances. According to viewpoint(s) set, we can see if the platform matches the requirements. If not, we can find what the problem is, and try to refine or modify the components implementation. When modeling the Avionics system, hardware component is seen as
pseudo black box element. We can only define the interface information and some properties. We can’t describe the behavior change. But for the software component, we can define their behavior.

When we make the validation of IMA, we would use the new validation approach which can automatically generate hardware and simulation scenario to simulate it. We will model the complete avionics platform. Then we analyze its performances which consist of different indexes according to the dynamic simulation of the platform. The process takes advantage of the AADL, modeling software architecture, describing hardware architecture.

When we make the verification of AADL ARINC653 architectures, we would perform such verifications by various techniques, such as queuing systems theory and real-time scheduling theory. If we would like to verify the IMA platform, we can use analytical methods or the algorithms in order to perform verifications with scheduling simulations. By current feasibility tests and classical real-time scheduling algorithms, we need to model corresponding specific schedulers and task models, the former ones is more simple.

Considering that hardware components can’t be dynamically simulated, so we concentrate on modeling the software, we can describe the software framework and their communication. And when occurring the failure, we also reconfigure the software in hardware. In addition, using AADL, we model software architecture and make hardware architecture description in textual way. Those things all we do are intended to find the potential problems.

4. Conclusion and Future Works

AADL is widely applied to Integrated Avionic Systems, it can describe the configuration of real-time and critical system throughout its life-time. When using it to model the reconfiguration of IMA, we can describe the software and hardware components in detail, then react to the corresponding fault of them, that is, give the reconfigured strategies. For the steps of IMA reconfiguration process, we model that that consist of stop, destroy, creation, connection and running of modules, inter-partition and intra-partition communication between modules. AADL can greatly help us to analyze the whole process. Otherwise, when we simulate, validate and certify the IMA platform using the AADL, we can find
ARINC653 of AADL plays a greatly important role in that. We will model the complete avionics platform as possible as we can, then describe the dynamic change of the timing, power consumption and safety. To improve our accuracy of the analysis process, we will obey the electronic evaluation boards. According to the concrete experimental results, we will make corresponding improvements.

But we should notice that there still are some things of IMA that we can’t model by AADL, and the process should be simplified further. Also, the results’ accuracy of the IMA platform simulation, validation and certification should be measured more accurately and improve. The modeling process need to develop towards a more intelligent direction. And we’d better be able to generate the codes automatically, thus finishing the modeling process. So these problems should be studied further in our future works.

5. Acknowledgement
This work was supported by a grant from the National Defense Basic Scientific Research program of China (No. JCKY2016204A102).

6. References
[1] G Biggs, T Sakamoto and T Kotoku. A profile and tool for modelling safety information with design information in SysML [J].Software & Systems Modeling, 2014, 25(6): 18-23.
[2] Wang Peng, Zhao Changxiao and Yan Fang. Research on the Reliability Analysis of the Integrated Modular Avionics System Based on the AADL Error Model [C]. Tian Jin: International Journal of Aerospace Engineering, 2018: 1-11.
[3] Liu Zhenxu and Zhao Zhen. Modeling and Schedulability Verification of IMA Partitioning Based on AADL[C]. Hang Zhou: IEEE,2017: 417-420.
[4] Robati Tiyam, Gherbi Abdelouahed and Mullins John. A Modeling and Verification Approach to the Design of Distributed IMA Architectures Using TTEthernet [J]. Affiliated Workshops, Procedia Computer Science, 2016 8(3):229-236.
[5] J. Delange, L. Pautet and A. Plantec, etc. Validate, Simulate and Implement ARINC 653 systems using the AADL[J]. Acm Sigada Ada Letters, 2009, 29 (3) : 31-44.
[6] J. Delange, J. Hugues, L. Pautet, and B. Zalila. Code generation strategies from aadl architectural descriptions targeting the high integrity domain[C]. Toulouse: ERTS, 2008:37-41.
[7] P. H. Feiler, D. P. Gluch, and J. J. Hudak. The Architecture Analysis and Design Language (AADL): An Introduction [J]. Technical report, 2006, 2(7):32-37.
[8] P. H. Feiler and J. Hansson. Flow Latency Analysis with the Architecture Analysis and Design Language (AADL) [J]. Real-Time Application Development with OSEK, 2008, 3(5): 21-28.
[9] J. Hansson and A. Greenhouse. Modeling and Validating Security and Confidentiality in System Architectures [J]. Technical report, 2008, 6 (8):23-31.
[10] Schmidt D C. Model-driven engineering [J]. COMPUTER-IEEE COMPUTER SOCIETY-, 2006, 39(2): 25.
[11] J. Chilenski. Aerospace Vehicle Systems Institute Systems and Software Integration Verification Overview[C]. Toulouse: ERTS, 2007: 16-21.
[12] Gu Q, Wang G, Xu W. Research on resource fusion for Integrated Modular Avionics system[C].IEEE/AIAA Digital Avionics Systems Conference, 2012 : 7A6-1-7A6-9
[13] R. N. Kashi and M. Amarnathan. Perspectives on the use of model based development approach for safety critical avionics software development[C]. International Conference on Aerospace Science and Technology, 2008: 32-45.
[14] Pierre Biever, Eric Noulard, Claire Pagetti, Thierry Planche and etc. Preliminary design of future reconfigurable IMA platforms [J].ACM SIGBED Review, 2009, 5(2): 1-5.