

Model-based Method to Automate the Design of IMA Avionics System Based on Cosimulation

Lin Bao and Guy Bois
École Polytechnique de Montréal

Jean-François Boland
École de Technologie Supérieure

Julien Savard
Mannarino Systems & Software Inc.

ABSTRACT

In the aerospace industry, as the modern avionics systems became more and more complex, the Integrated Modular Avionics (IMA) architecture has been proposed as a replacement of the federated architecture, in order to offer better solutions on SWaP constraints (Size, Weight and Power). However, the development process of IMA avionics systems is much more difficult. This paper aims to propose to the aerospace industry a set of time-effective and cost-effective solutions for the integration and functional validation of IMA systems.

Based on MBE methodology, which is considered as an interesting solution for the IMA systems development [8], this paper proposes a design flow, that integrates three steps of refinement, for the configuration and the validation of IMA platforms. In the first step of the design flow, the modeling language AADL is used to describe the IMA architecture. The AADL modeling environment OCARINA, a code generator initially designed for the real-time operating system POK, has been modified to generate software integration code and system configuration files for the IMA simulator named SIMA. This solution is a cost effective alternative to expensive commercial development environments to validate ARINC653 software applications. In the second step of the design flow, a cosimulation platform composed of two simulators is proposed: Simulink for the simulation of peripherals and SIMA for the simulation of IMA modules. In the third step, the validated avionics applications and system configuration can be ported with minimum effort from the cosimulation environment to an implementation platform.

A case study, which consists in integrating several avionics applications to SIMA and then porting them to PikeOS development environment, was brought in the purpose of demonstrating the proposed design flows and co-simulation platform. The research work realized in this paper is a part of collaboration between industrials and academics through the CRIAQ AVIO509 project.

CITATION: Bao, L., Bois, G., Boland, J., and Savard, J., "Model-based Method to Automate the Design of IMA Avionics System Based on Cosimulation," *SAE Int. J. Aerosp.* 8(2):2015, doi:10.4271/2015-01-2531.

INTRODUCTION

In an aircraft, no catastrophic failures may be tolerated during its service. An aerospace system has to meet many constraints [1]: safety, dependability and real-time constraints. In order to optimize these constraints, federated avionics architecture and IMA architecture [2] have been proposed one after the other.

The traditional federated architecture defines some avionics systems, where software and hardware are tightly coupled. It follows the design concept "one function = one dedicated system". In this way, many redundancies are created in order to guarantee high level system reliability in such architecture. However, the design period

and cost of modern avionics systems are accordingly increased as they become more complex. In addition to the safety consideration, the SWaP constraints are gaining more and more importance in aviation industry for the design of modern planes. Therefore, the IMA architecture has been proposed to meet the industry's needs. IMA is considered as the new generation avionics architecture that supports a strict and robust time and space partitioning. However, compared with the previous federated architecture, the design of IMA systems is much more complex to achieve. In this work, we show how availability of some dedicated design tools becomes an important evaluation factor for an IMA system development environment.

Model-based Engineering defines the development by constructing a model for a system which can be transformed into something concrete such as C/C++ or Java code [3]. The transformation from high-level abstract models into running systems can usually be automatically done by some code generators or compilers. Consequently, this methodology is becoming more and more popular because of its efficiency, and the possibility to identify and analyse errors earlier in the software design cycle [8]. Therefore, the MBE philosophy has been used to develop safety-critical IMA systems, to decrease their design time and resolve software problem at the earliest possible time.

In real avionics systems, for most of the time, applications executed on the IMA module have to exchange data with the external world, for example, some avionics peripherals such as sensors or actuators. Therefore, peripherals must be physically present or simulated (e.g. Simulink [35]) by software when testing the whole system.

The main contribution of this paper is to propose a design flow to perform the configuration and validation of IMA platforms. It represents the integrated result of three steps. In the first step, AADL is used to describe the IMA architecture, while Simulink models the peripherals (sensors/actuators) presenting the external world of the IMA module. Based on the MBE, these two models are automatically refined by their code generators. In the second step, SIMA, an IMA simulator, and Simulink form a cosimulation platform by using TCP/IP network for data exchange. Finally, in the third step, we show how a validated avionics applications and system configuration can be ported with minimum effort from the cosimulation platform to an implementation environment.

This article is organised as follows. [Section 2](#) introduces the project's background and the related works. [Section 3](#) describes the proposed design flow. [Section 4](#) presents the methodology. A case study is analysed in [section 5](#) and the results are given in [section 6](#). Finally, [section 7](#) presents the conclusion.

BACKGROUND

IMA Architecture

Based on the resource sharing among different avionics applications with different criticalities, IMA creates “partitions” by separating the system in terms of time and space. A partition is a container for applications including code, data and system configuration information. Time partitioning means that all the defined partitions are executed in a predefined order and during a predefined period, for the purpose of avoiding time conflict between them. Space partitioning indicates that every partition has its own memory block which cannot be shared between different partitions, so that no memory conflict among partitions will happen. Because of the partitioning mechanism, IMA has a two-level hierarchical schedule: in the first level, the OS module (MOS) schedules statically all the partitions; in the second level, the OS partition (POS) schedules all the processes of the same partition by priorities. In short, the resource sharing and the system partitioning are two major characteristics for IMA architecture [1].

ARINC 653 Standard

The ARINC653 standard [4, 5, 6] specifies a set of services for the development of safety-critical avionics systems. It defines an ARINC653 Application Executive (APEX) that is considered as an API that offers the following standard services [8]: partitioning and scheduling policies, tasking service, intra/inter-partition communication service, health monitoring service, and ARINC653 systems validation needs. The functionalities offering these services are standardized. Since the APEX layer isolates the avionics applications and the underlying OS that supports them, the applications become independent of their OS. As a consequence of this separation, the migration of avionics applications from one ARINC653 execution environment to another becomes less complex.

Past Related Work

Many commercial ARINC653 compliant RTOS exist already, such as PikeOS by Sysgo, VxWorks by Wind River and Integrity-178B by Green Hills Software. In general, those commercial environments are used at the implementation step, when the OS and the target platform are fixed.

In this work we consider also the system and early simulation before acquisition of a target platform. In this context, [7] identifies the need of an IMA design flow consisting of low cost tools that offer both systems modeling with MBE functionalities and a simulation environment compliant with ARINC653. More precisely, some low cost partitioning RTOS alternatives are studied, including POK [12], SIMA [13], AIR [10] and XtratuM [11]. Several conditions are taken into consideration as selection factors: the dependency of hardware platform; the level of compliance to the ARINC653 standard; the availability of technical support and documentation; the integration with an MBE environment. Since authors [7] have not identified an existing design flow meeting these requirements, they propose a new one (Figure 1). The approach consists in modeling systems using AADL language, and then generating integration source code and configuration files for the simulator SIMA by using an extended version of the open source OCARINA tool.

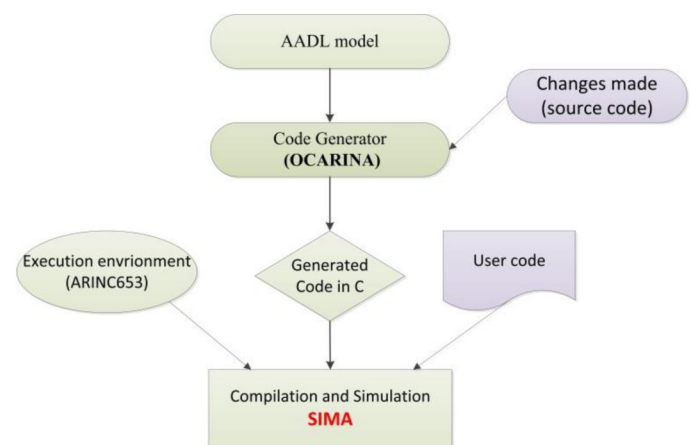


Figure 1. AADL design flow for SIMA

The case study experiment has demonstrated that system configuration and setup time can be considerably reduced. It takes seconds to generate software files that may otherwise take hours when done manually, especially considering the risk of human error.

PROPOSED DESIGN FLOW

In this paper we go further in the refinement of an AADL-based design by improving the OCARINA generator, integrating the generated code into the external environment and by porting the software to a hardware platform with an ARINC653 compliant partitioning RTOS.

We propose a design flow (Figure 2) integrating three levels of abstraction: 1) a modeling level that is an improvement of [7], 2) a simulation level and 3) an implementation level. It provides a set of solutions to model, simulate and implement some avionics systems.

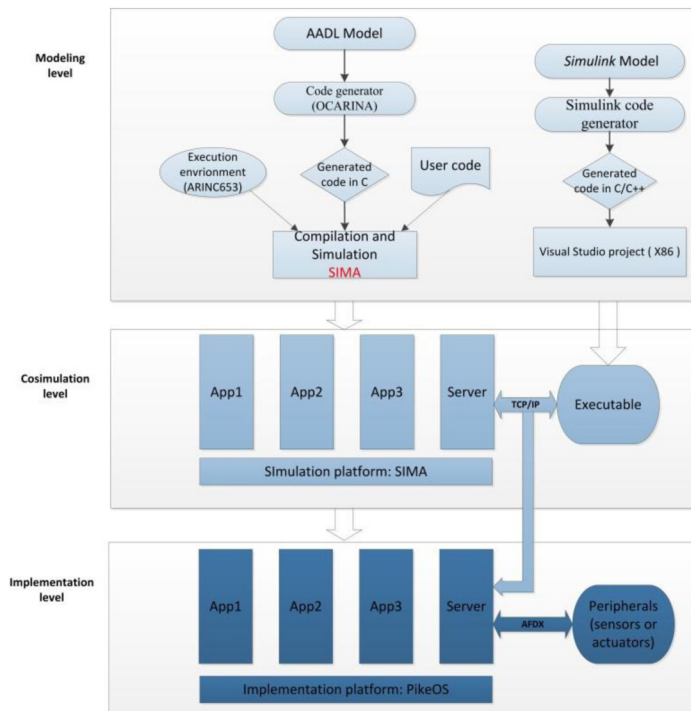


Figure 2. The proposed design flow

At the first level, AADL and its extended version of code generator, named OCARINA, are used to efficiently integrate the API ARINC653 and make system configuration for SIMA. Simulink is used to develop quickly avionics peripherals such as sensors and actuators for simulating the flight environment. Thus, comparing to the design flow presented in [7], more complete avionics systems can be modeled with this new flow. Also, OCARINA was improved in order to provide better performance. In the second level, a cosimulation platform consisting of IMA modules, simulated by SIMA, and external flight environment, simulated by Simulink, achieves a functional validation of avionics applications. The data exchange at this level is done through TCP/IP network. At the third level, the RTOS is chosen as the implementation platform. In this work we selected PikeOS. The commercial AFDX bus and the real avionics peripherals should have been installed with the IMA module

at the implementation level. Since we do not have access to an AFDX, in this work, we reuse the simulated peripherals and the TCP/IP network in order to complete the implementation.

AADL Modeling Language

Being an ADL type language, AADL proposes a component-based technology to describe complex embedded systems. With the software and hardware components offered by its library, a complete embedded system, which includes applications, data, OS, memory, platform etc., can be modeled. It is standardized by SAE International for the development of safety-critical avionics system. AADL is extensible by defining new proprieties or enlarging its annexes. In such way, after an extension in 2011, AADL 2.0 [15] begins to support the standard ARINC653 standard. As a consequence, IMA architecture can be modeled by describing its entities such as partitions, processes, inter or intra partition communication and error monitoring with their corresponding components defined in the AADL library.

From AADL to Cosimulation

Simulink

Being integrated in MATLAB platform, the modeling tool, named Simulink, is used in many industrial fields today. Simulink offers an efficient graphical development environment. With the digital and analog components provided by Simulink, a system or application can be quickly created.

This modeling environment is also used for creating some avionics applications. For instance, “DO Qualification Kit” provides documentation, test cases, and procedures that allows qualify Simulink software verification tools for projects based on the commercial airborne software certification standard DO-178B [23, 24]. Another important feature is about the large number of avionic components available in the library. It allows the developer to create some avionics peripherals. Simulink is integrated with a code generator. Before starting the simulation, the model must be compiled successfully. During the compilation stage, syntax and semantic checks will be performed to assure that the model is complete [16]. Then, the generator can automatically create code such as C/C++ from a predefined model. In short, Simulink provides an efficient MBE development flow for the development of avionics peripherals, and the generated code can be easily extended and reused as required.

SIMA Simulator

SIMA (Simulated Integrated Modular Avionics) is a simulator that provides the avionic applications with an ARINC653 execution environment. SIMA works on top of an OS supporting POSIX such as Linux/GNU [9]. As a result, it cannot offer a safety-critical execution environment. However, SIMA is independent of the target platform, and it provides robust TSP, which guarantees the same application behaviors as on the target platform by strict and proven conformance to the ARINC653 specification. A partition is reserved for playing the role of MOS, which is responsible for the time partitioning and the inter-partition communication. The services offered by a dedicated library are available as POSIX. The