A New Modeling Approach for IMA Platform Early Validation

Michaël Lafaye, David Faura, Marc Gatti, Laurent Pautet

Telecom Paristech, LTCI
46 rue Barrault
75634 Paris Cedex 13, France
{lafaye,pautet}@telecom-paristech.fr

Thales Avionics, ACS/DTEA
18 avenue Maréchal Juin
92366 Meudon-la-Forêt Cedex, France
{david.faura,marc-j.gatti}@fr.thalesgroup.com

ABSTRACT

This past few years, avionics platform conception changed to integrated architecture, permitting one processor to host some applications, in order to reduce weight and space. But this method entails more complexity, especially in safety domain, while time to market tends to decrease, so new development processes are needed. Model-based approaches are now mature enough to design embedded critical systems and perform architecture exploration.

In this paper we present a new modeling approach allowing avionics platform description and dynamic simulation. This method aim at dimensioning the architecture according to the applications it has to process, and to achieve early platform validation.

General Terms

Performance, Design, Verification.

Keywords

modeling, avionics systems, real-time, simulation, AADL, systemC

1. INTRODUCTION

Avionics systems are critical real time systems, i.e. timing constraints have to be strictly respected at the risk of catastrophic issue. They are composed of applications, real-time operating system and hardware modules. Initially, avionics platform (hardware and operating system) were implemented as federated architectures, where one processing unit hosted one function. This relatively simple architecture was however costly in terms of space, weight and power consumption, but offers a simple approach regarding the certification.

In order to reduce these parameters, and also to reduce costs, the Integrated Modular Avionics (IMA) concept was developed in the 2000s. It defines integrated architectures, where one processor can host some applications, and so reduces the number of modules used in avionics platform. Following this evolution, suppliers developed network architectures, in which modules are interconnected and communicate through a deterministic network. However, aggregating applications in a few modules, and gathering communications in a central network entails an increase of complexity in avionics platform design, verification and certification processes. In the same time, time to market tends to decrease. These developments require an early modeling of the system to validate and maximize the use of the future platform, while ensuring the critical level required by current standards in aviation (DO-178B, DO-254, MILS-CC…).

To model this IMA platform and perform early validation, Model-Driven Engineering (MDE) approaches are now suitable to describe system high-level functionalities. Many projects aim at modeling these platform with Architectural Description Languages (ADL) as AADL [1] or MARTE [2], or with synchronous languages such Lustre or Signal [3,4]. However, they often focus on the applications description, model the hardware as connected blackbox components with a few properties, and perform static simulation. Moreover, there actually is no automated process for complete platform modeling and simulation.

We define a new modeling method, that aims at designing a complete avionics platform (hardware, operating system and the applications). It is a component-based approach, relying on two languages and taking advantages of both: AADL and systemC. AADL [5] is, as MARTE, an ADL particularly adapted for software architecture description [6,7], enabling the modeling of ARINC 653 embedded real-time system [8]. In view of the experience of partners who developed the ARINC 653 AADL annex, the ARINC 653 compliant runtime for AADL called POK, the Ocarina tool suite etc., we choose the AADL rather than MARTE. SystemC [9] is an IEEE standard widely used in industry for hardware platform description, and containing a simulation kernel for architecture simulation and exploration.

In this paper we present our new modeling approach. We first introduce IMA concept, then we detail our method, before concluding with perspectives of our work.

2. Integrated Modular Avionics platform

IMA concept introduced integrated architecture, allowing to reduce the number of different modules used for platform design. As illustrated in figure 2, an IMA platform is composed of avionics applications, embedded operating system and the underlying hardware. This latter is composed of several processing modules communicating through a deterministic network, the AFDX (Avionics Full DupleX).

MONPES'10, September 20, 2010, Antwerp, Belgium.

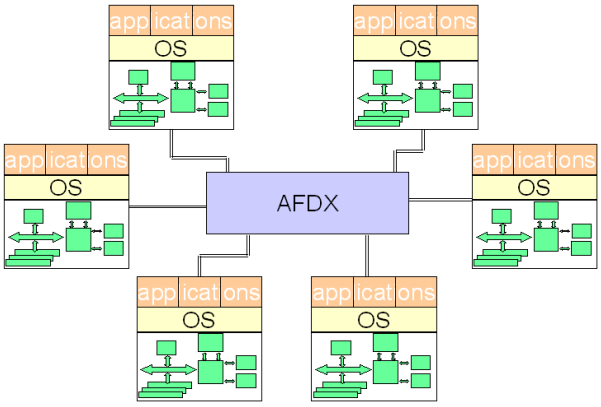

Figure 1. IMA platform

A processing module can host one or some applications at different criticality levels, it is then necessary to respect safety constraints. That's why OS ARINC 653 standard was defined, which specifies space and time partitioning. Figure 2 gives an overview of an IMA module embedding an ARINC 653 operating system. To ensure space partitioning, each application is enclosed in one or some partitions, isolating it from each other. Each partition is bound to a part of memory, so it only access its memory area. This partitioning prevents from failure propagation. Intra and inter-partitions communications are also defined by the standard to prevent failure propagation. To ensure time partitioning, each partition has its own execution time window, during which the application has access to all resources dedicated (processing, memory, dedicated I/O etc.).

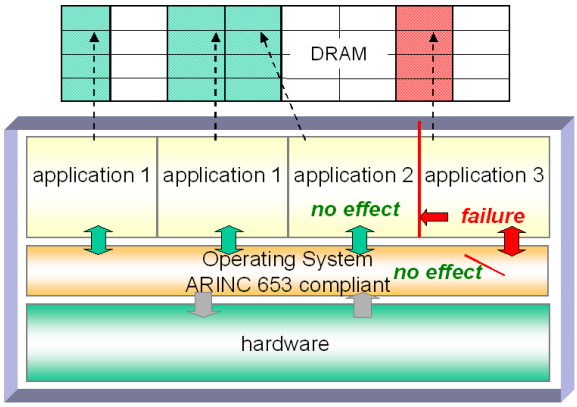

Figure 2. ARINC 653 spatial partitioning

3. IMA platform modeling

3.1 Overview

Model-Driven Engineering approaches are now mature enough to serve as a basis for building embedded systems and perform early validation. They are especially suitable for modeling the high-level architecture, that are the functional architecture (description of the functionalities offered by the system) and logical architecture (description of how the system is structured into logical components cooperating by communications) [10,11]. But at platform architecture level, these approaches describe both hardware and software as static blackbox elements with some properties.

Some projects [1,2,3] aim at building more accurate platform models, but they mainly focus on the software behavior, and model hardware as one or a few blackbox components without behavior information. They after simulate this latter statically. There is so no method to retrieve dynamic performances from the hardware to validate it according to the applications requirements.

Our method develops a new approach for avionics platform modeling. It aims at refining architecture description at platform decomposition level, specially the hardware. In order to refine this latter, our approach models and sizes it according to applications requirements. As we can see in figure 3, this approach consists of different tasks:

- system modeling;
- applications characteristics extraction;
- platform generation according to requirements;
- platform simulation with simulation scenario made with extracted stimuli;
- platform performances analyses.

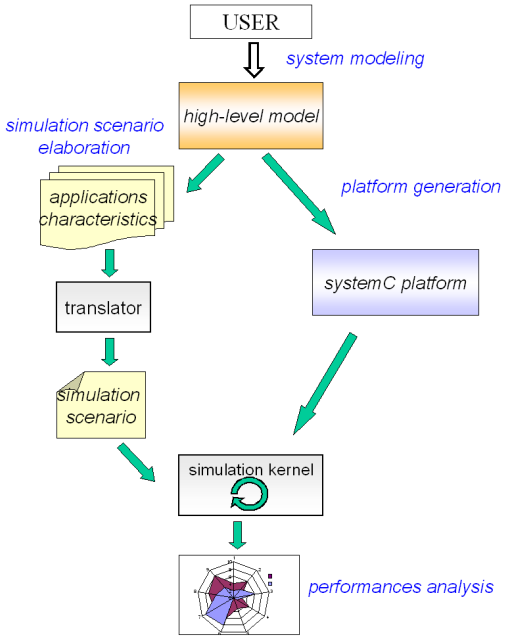

Figure 3. Modeling approach.

3.2 Application modeling

The description of the application consists of a set of characteristics (parallel code percentage, hit cache rate etc.), that will give the future platform requirements, and a set of instructions which will be translated into hardware stimuli. Figure 5 shows the two ways we can use to extract these characteristics from the application, and build a simulation scenario to simulate the platform.

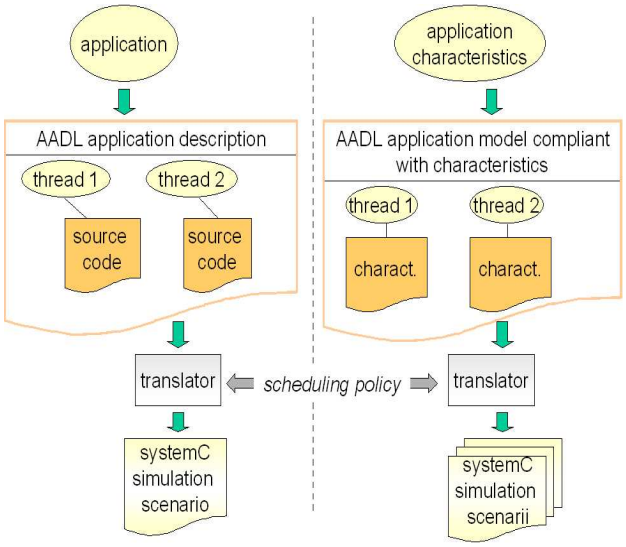

Figure 4. application modeling.

In the first way, -left path in the figure 4-, we have access to the application source code. In this case, the application is modeled with the AADL threads, which represent the ARINC 653 processes of the application. They are configured according to this processes (deadline, priority etc.). Threads are bound to a processor (or partition) and a memory (or part of memory), and ordered according to the scheduling policy defined for the application. Each thread as a reference to a part of the source code, from which we extract characteristics giving the requirements the platform has to match.

To elaborate the simulation scenario, we use a code profiling and application decomposition method. The figure 5 gives an overview of an application decomposition: it is decomposed into a logical function sequence (encoder, decoder…), refined into basic functions instructions (FFT, FIR…). Each basic function is composed of simple instructions (operator and operand(s)), that can be simple operations (add, div etc.) or memory access. The code attached to the AADL thread is parsed and instructions are extracted. We translate these latter into corresponding systemC instructions. For example, a "load" gives a systemC READ_COMMAND instruction.

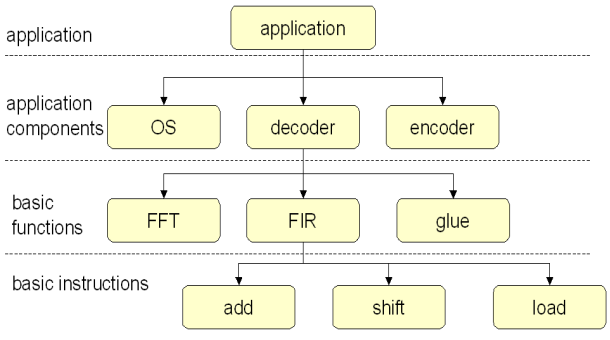

Figure 5. Application decomposition example

On the other way -right path on the figure 4-, we have not the source code of the application, so we can not extract basic instructions. However, some main application characteristics (sequential code percentage, scheduling policy etc) are given, and allow the elaboration of several constraint-random simulation scenario which fulfill application requirements. This method allows a hardware platform early validation without access to the application, but only with representative characteristics. It is less accurate than the first way, but is easier and faster.

3.3 High level system modeling with AADL

As we saw in the previous section, the application is described with AADL threads.
The real-time operating system is defined by some properties dispatched in the different hardware components. For example, scheduling policy is set in the processor module, partition security level is defined in the virtual processor, etc. To model an ARINC 653 operating system, we use the AADL 653 annex, and the method described in this article [8].

Each hardware component is modeled with the AADL corresponding component, or with the device element. Some components more complex, like network, are modeled as a sub-system containing some components. We model hardware component as a pseudo blackbox element, where behavior is not defined. We define interface information (ports, bus required if needed) and a few properties (memory size, bus transfer latency etc.). In order to refine those hardware properties, we created DRAM and cache component (that inherit memory element) to refine their read and write latencies, and we defined or completed some AADL property sets. We introduced behavior and specific properties, like cache hit rate for cache module, or refresh time for DRAM component.

The user models with the AADL the system corresponding to the platform, and choose which viewpoint(s) will be set when analyzing the platform. Viewpoints can be for example timing performance, power consumption or safety, and enable the platform investigation under different angles. We then extract the main characteristics of the application, and parse this AADL platform to retrieve properties, connections and deployment information, that will serve for systemC platform generation.

3.4 Platform integration by generation

In order to generate the systemC platform, we developed a database of configurable systemC components. These component have three parts: behavior, main properties and interface. For each of them, we identified the main characteristics and defined a configurable automata. At each state of this one are attached parameters corresponding to viewpoints and/or global parameters. Figure 6 shows an example of DRAM automata with some timing parameters (tRefresh, tRDC etc.) and one global parameters (burst_cpt).

For each platform element of the AADL model, we retrieve its information (properties, connections etc.) and configure the corresponding systemC behavioral model. Then, we connect all the modules to elaborate the refined systemC platform.

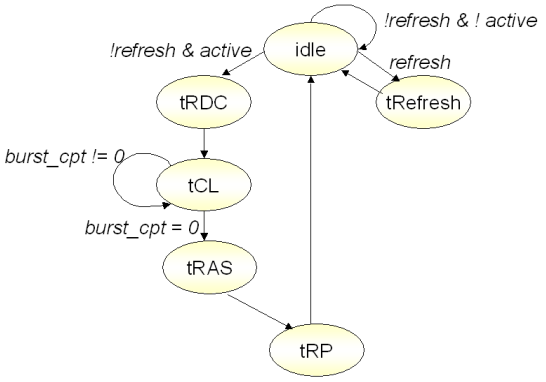

Figure 6. DRAM automata example with timing annotation

3.5 Platform simulation and results

Currently, this is a work in progress. However, we have already encouraging results. The AADL part of the process has been specified and is under development, while systemC main hardware components (processing unit, cache memory, dram and bus) have been developed (behavior, main properties and communication interface). In order to test and refine these elements, we implemented a minimum hardware platform with one or some instance of each of them. We also developed a systemC frame generator that simulates the platform, so we can observe the elements behavior and the platform communications.

Otherwise, to test our future platform, we defined a simulation and performances analysis method: to see if the hardware platform built is compliant with the application requirements, we perform a simulation using systemC kernel. It takes the platform generated, the viewpoint(s) set, and applies the simulation scenario. As we can see in the figure 7, the user can analyze the platform performances by examining performances graphs or simulation traces. Then, we can see if the platform matches the requirements corresponding to the viewpoint(s) set. If the hardware is not compliant with the applications requirements, we can investigate what is the problem, and try to refine or modify one or more components implementation.

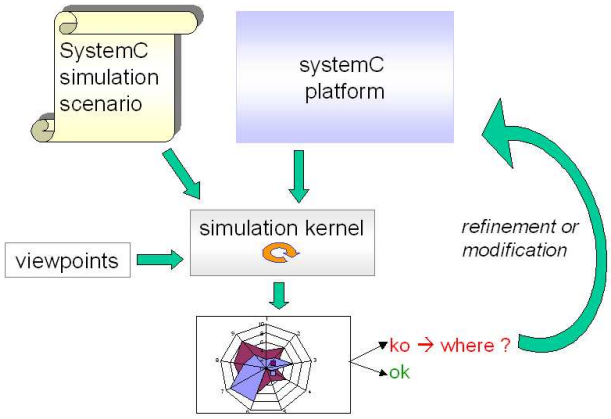

Figure 7. Platform simulation and performances analysis.

4. Conclusion

Current early platform validation methods center on software modeling, regarding the hardware as blackbox components which can't be dynamically simulated.

We have presented a new early validation approach, that aims at modeling a complete avionics platform, software and hardware (i.e. IMA modules and their interconnections as AFDX). Our method automatically generates hardware and simulation scenario to simulate it. It enables a dynamic simulation of the platform, and analyzes its performances according different viewpoints (timing, power consumption or safety). It takes advantages from the AADL, particularly adapted for software architecture modeling, and from systemC, industrial standard for hardware architecture description.

To validate the accuracy of our modelling methodology, we first model electronic evaluation boards. We will afterwards model a complete IMA platform to compare the model performances with the experimental results. Otherwise, we will connect with existing model-driven engineering methods and improve the platform development process.

5. REFERENCES

[1] Support for Predictable Integration of mission Critical Embedded Systems project (SPICES), 2009 http://www.spices-itea.org

[2] Model-Based Approach for Real-Time Embedded Systems development project (MARTES), 2007. http://www.martes-itea.org/

[3] C. Brunette, R. Delamare, A. Gamatié, T. Gautier, J-P. Talpin, "A Modeling Paradigm for Integrated Modular Avionics Design", IRISA report, 2005.

[4] Y. Ma, J-P. Talpin, T. Gautier, "Virtual prototyping AADL architectures in a polychronous model of computation", IRISA research report, 2007.

[5] AADL Portal at Telecom Paristech : http://aadl.telecom-paristech.fr/

[6] J. Hughes, F. Singhoff, "Développement de systèmes à l'aide d'Ocarina et Cheddar" *ETR09*, 2009.

[7] P. Dissaux, F. Signhoff, "the AADL as a Pivot Language for Analyzing Performances of Real Time Architectures", *4th European Congress ERTS Embedded Real Time Software*, 2008.

[8] J. Delange, L. Pautet, A. Plantec, M. Kerboeuf, F. Singhoff, F. Kordon, "Validate, Simulate and Implement ARINC 653 systems using the AADL", *CM SIGAda Ada Letters*, 2009.

[9] Open SystemC Initiative. IEEE 1666: systemC Language Reference Manual, 2005. www.systemC.org.

[10] J.A. Estefan. "Survey of model-based systems engineering (MBSE) methodologies". Technical report, *INCOSE MBSE Focus Group*, may 2007

[11 Bernhard Schätz, Manfred Broy, Franz Huber, Jan Philipps, Wolfgang Prenninger, Alexander Pretschner, Bernhard Rumpe, "Model-Based Software and Systems Development – a white paper", 2004