Refinement of AADL models using early-stage analysis methods

Guillaume Brau[1] Jérôme Hugues[2] Nicolas Navet[1]

[1]University of Luxembourg, Laboratory of Advanced Software Systems,
6 rue Richard Coudenhove-Kalergi, L-1359 Luxembourg, Luxembourg
{guillaume.brau, nicolas.navet}@uni.lu

[2]Université de Toulouse – ISAE, 10 avenue E. Belin, 31055 Toulouse, France
jerome.hugues@isae.fr

Abstract

Model-Driven Engineering (MDE) is a relevant approach to support the engineering of distributed embedded systems with performance and dependability constraints. MDE involves models definitions and transformations to cover most of the system life-cycle: design, implementation and Verification & Validation activities towards system qualification. Still, few works evaluate the early integration of performance evaluation based on architectural models. In this paper, we investigate the early-stage use of analysis in AADL modeling. Precisely, we exemplify on an avionics case study how to dimension the data flows for an application distributed over an AFDX network. Based on the insight from this study, we suggest a simple framework and associated techniques to efficiently support analysis activities in the early-stage design phases.

Keywords AADL, ARINC653, AFDX, analysis combination, WCTT evaluation, Network Calculus.

For more details and experimental results, an extended version of this paper is available as a technical report [1].

1. Introduction

Context of the paper. Distributed Real-time Embedded (DRE) systems are present in safety-critical domains such as transportation, telecommunications, health services, military or space. These systems have to meet both the *functional* and *non-functional* requirements. Hence, DRE systems encompass specific technologies to realize the required service with the expected performance metrics (*e.g.* time, security or safety) through dedicated networks, processors or real-time operating systems. In addition, the engineering process has to address efficiently system modeling and evaluation of all metrics.

The 4th Analytic Virtual Integration of Cyber-Physical Systems Workshop
December 3, 2013, Vancouver, Canada.

Definition of the problem. Many experiments indicate that the distance between the activities steps in a classical V-cycle is detrimental and usually slows down the development process [2]. In practice, a significant part of errors is injected at early-stage of the engineering process, while being detected during later integration phases. As a consequence, regressions and rework activities have an important weight on the overall project costs.

We believe that this problem can be lifted considering models with sufficient power of expression to guide the conception phases (e.g. using interface or behavioral models on which one can both reason and iterate) and supporting early-stage analysis. This "integrate, then build" approach, also known as *virtual integration* is promising to support the design of complex systems.

Contributions and objectives. In this paper, we use the Architecture Analysis & Design Language (AADL) [3] as the pivot to capture the system architecture and derive its performances. AADL is an Architecture Description Language (ADL) suitable to describe systems, capturing the *functional* and *non-functional* concerns together with the *operational platform*. As the AADL provides modeling elements with a precise *syntax* and well-defined *semantics*, it is possible to : 1) perform analysis and 2) derive implementations thanks to code generation [4].

The focus of this paper is twofold. First, taking a specific example coming from the avionics, we show how to accurately seize important parameters in the AADL model. As the system that is modeled includes technologies not supported within the core language, we build on and extend the work in [5] in order to include in the AADL model the networking elements and capture the overall DRE system.

Secondly, based on lessons learned during the modeling experience, we propose to jointly use AADL models and analysis methods so as to gradually refine and verify the model. We investigate and examplify this strategy on the avionics case study. Taking as example the network communications and the expressed timing constraints, we show that using complementary analysis methods allows to deduce missing parameters while maintaining the consistency of the model (*i.e.* chosen parameters guarantee that non-functional constraints are met).

The paper is organized as follows: we first introduce the avionics case study and give elements to model it with AADL (Section 2). In Section 3, we underline relationships between modeling concerns, and, using analysis, we explain how to solve those dependencies for some important communication parameters. Finally, in Section 4, we draw conclusions from this case study and discuss future work.

2. Modeling avionics systems with AADL

In this section we describe how to jointly capture with AADL the functional description of an avionics system, its temporal constraints and the target execution platform.

2.1 The Flight Management System

The avionics system to model, coming from Lauer et al. [6], is part of an aircraft's Flight Management System (FMS). It interacts with the crew and provides static and dynamic information about the flight plan (*i.e.* the predefined path between departure and arrival points) : current location, remaining distance and estimated arrival time.

Functional description. The system is made up of five main functions as depicted in Figure 1. The *Keyboard and cursor control Unit (KU)* reads data inputted by the pilot (or copilot) through the keyboards while the *Multi Functional Display (MFD)* refreshes the displays consecutively. From the KU function, crew requests are forwarded to the *Flight Manager (FM)* function which computes the response about the flight plan and returns it to the MFD. For this, it requests static data to the *Navigation Data Base (NDB)* function and also relies on dynamic data from the *Air Data Inertial Reference Unit (ADIRU)*, computed based on sensors measurements.

Temporal constraints. In the avionics context, the system has to conserve a predictable behavior. Several temporal constraints may be expressed upon different "locations". Typically, temporal constraints concern : 1) response times which are the delays needed to carry out the functions, 2) traversal times refering to communication delays between functions and 3) latencies along functional chains that encompass a succession of response times and traversal times.

Execution and communication platform. The *functions* are executed in an *Integrated Modular Avionics (IMA)* environment. In particular, this execution and communication platform supports two standards, used in the case study, that defines the use of the shared hardware and software resources in a deterministic way.

The ***ARINC653*** [7] standard defines management of the functions hosted by a same hardware/software platform (referred as an execution *module* in the following). In this environment, each function is located in a different *partition* with a strict access to processing and memory resources.

The ***ARINC664*** [8] standard defines a deterministic communication network called *Avionics Full Duplex-Switched Ethernet* (AFDX). AFDX implements the core concept of *Virtual Link* (VL) which is an unidirectional logical connection from one sender to one or several receiver(s). Each VL has a limited bandwidth according to its *Bandwidth Allocation Gap* (BAG in the following) – the minimum time elapsed between two frames sending – and a *maximal allowed packet size* (s_{max} in the remainder).

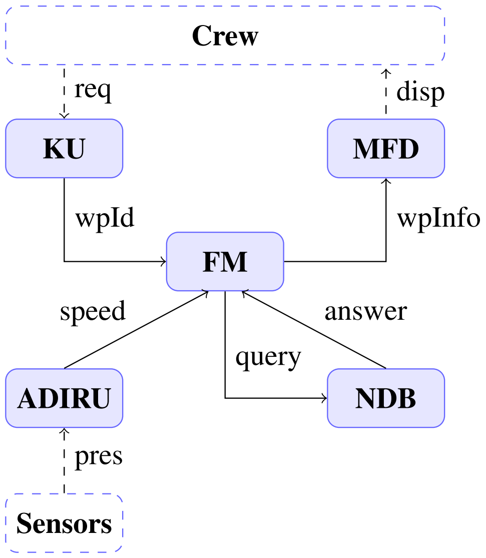

Figure 1. The Flight Management System functional architecture depicts the functions and the data exchanged as well as the interactions with the actors.

2.2 Modeling the FMS in AADL

The Architecture Analysis & Design Language (AADL) [9] is an international standard by the *Society of Automotive Engineers (SAE)*, defining the basics of an architecture description language dedicated to the design of DRE systems. AADL is component-centric and allows to specify both software and hardware parts of a system. See [3] for a complete presentation of AADL.

The full model of the FMS uses AADLv2 core specifications and the ARINC653 Annex. The model is made up of AADL *components* that describe the ARINC653 execution modules hosting the avionics functions and the AFDX network that supports the data exchanges. The full AADLv2 textual model – that spans over 770 SLOCs – is part of the AADLib project and is available at `http://www.openaadl.org`.

The model follows the initial specifications and AADL guidelines for ARINC653 systems : a module is a distinct `system` (containing a global `memory` and a `processor`) that hosts partitions (each is a `process`) bound to separate `memory` segments and `virtual processors` (representing spatial and temporal partitioning). `thread` components contained in partitions realize the avionics functions. Thanks to annex guidelines, we can model precisely the ARINC653 components and associated parameters.

AADL does not provide specific guidelines for modeling AFDX networks. The AADL concept of `virtual bus` defines a connection supported in a `bus`. We use this concept to define AFDX virtual links. Switches are represented by `device` components bound to the virtual links. A dedicated property set has been defined to model parameters attached to virtual links, end systems and switches.

3. Analysis as part of the design process

We discussed how AADL allows to capture both the FMS functional and non-functional aspects as well as the IMA platform description. From this model, we may now consider further analysis of the full architecture. The current design could be used to validate a given architecture. Yet, the most challenging part is actually guiding the designer in finding a suitable definition of the architecture parameters in order to respect the constraints expressed in the model.

3.1 Discussion : there are dependencies between modeling concerns

Figure 2 summarizes the three traits caught in the architecture model : the functions to realize, the hardware and software platform hosting the functions and the constraints to comply with. It implies that the architecture model components and the attached properties have to integrate and solve the dependencies between these views.

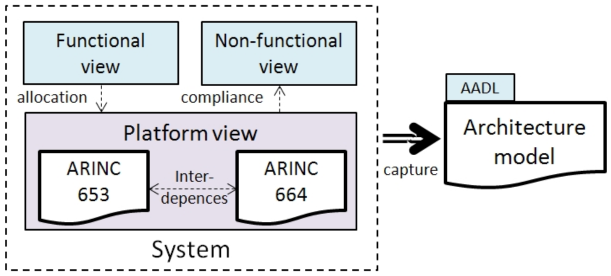

Figure 2. The architecture model captures jointly the system functional, non-functional and platform concerns and has to integrate the dependencies between these aspects.

For instance, let us consider the design of virtual links in AFDX networks. The virtual links characterization (***platform*** view) depends on :

- the data flows needed to realize the functions (***functional*** view) and their features, *e.g* the number (n) of messages sent by a function and their size (m),
- the constraints expressed onto the data flows (***non-functional*** view), *e.g* a communication between two specific tasks can be subject to timing constraints (noted L_C in the sequel).

Moreover, finding a suitable design for the virtual links in AFDX networks can be a difficult problem because of the interferences between VLs definitions. For further information, the issue is addressed in a more comprehensive setting in [10].

Proposed approach. We believe it is possible to deal with the dependencies between the modeling concerns by executing relevant analysis methods onto the model under construction. In the sequel, we show how to define progressively the VLs parameters taking into account information in the model such as the constraints expressed upon the communications. From an evaluation perspective, it implies : 1) isolating model input parameters that can be combined to 2) propose a feasible solution which is 3) later assessed. Some parameters are mandatory, while others can be assumed. This is discussed in the following paragraphs.

3.2 Example : integration of the Bandwidth Allocation Gap (BAG) parameter into an incomplete AADL model

Let us consider a partially completed AADL model. This initial model contains basic information : the functions hosted in the modules and their properties as well as the data exchanges between them. We also know the constraints of the system expressed onto the communications.

At this stage, dimensioning the BAG, which has a direct impact on the respect of timing constraints and the network load, may be a difficult task because the design space can be huge. Indeed, as this parameter ought to respect the formula BAG= 2^k [ms] with k integer in range 0 to 7 (see [8]), if we take as assumption that one VL is dedicated to one data flow, then there are $sol_{bag} = 8^f$ conceivable solutions, with f the number of flows.

To overcome this problem and complete the model, we execute the process pictured on Figure 3. We propose to:

1. use a *pivotal* Worst-Case Traversal Time evaluation (WCTT in the following) in order to identify the set of suitable BAGs,
2. use a *complementary* analysis method, relying on Network Calculus (NC in the following), to improve the results of the main WCTT evaluation.

WCTT evaluation. The Worst-Case Traversal Time analysis aims to assess the delay experienced by each data flow in the AFDX network. Without too much details, the WCTT evaluation gives the formula of an upper-bounded delay (D_T^{wctt}) suffered by a frame as a function of several parameters. Hence, it is possible to compare the delay against the expressed constraint ($D_T^{wctt} \leq L_c$) and to calculate the suitable set of BAGs [1]. The WCTT evaluation gives a set of suitable BAGs for each VL ($BAG_{vl_i}^{wctt}$). Of course, the accuracy of the BAGs sets depends on the precision of the model and of the assumptions. In addition, at the first stage, there is no NC feedback, i.e. $D_{sw\_vl_i} = 0$.

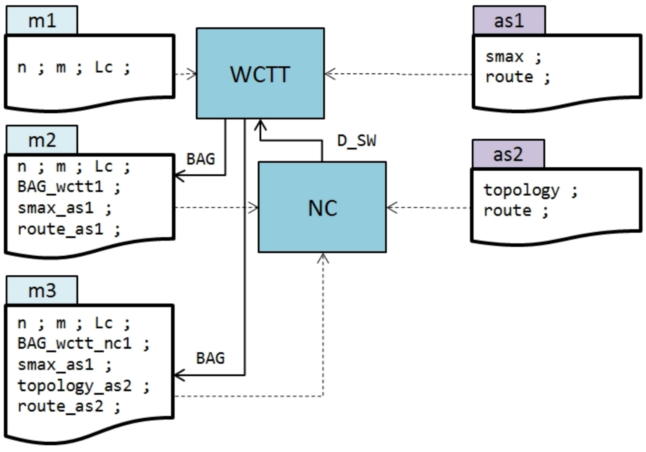

Figure 3. The BAG refinement process includes AADL models (blue-headed shapes), analysis methods (portrayed by green rectangles) and assumptions models (purple-headed shapes).

NC analysis. Network Calculus (NC) is an algebra for computing accurately the end-to-end delay of a data flow in the AFDX network. In the scope of this paper, this analysis is performed using RTaW-Pegase, which is a commercial product implementing a state-of-the-art network calculus AFDX timing analysis [11]. In our case, the NC analysis computes complementary information ($D^{nc}_{sw\_vl_i}$) and allows to refine results of the main WCTT evaluation.

Model refinement. We can see through the modeling and analysis flow (figure 3) that, as long as the required analysis inputs are present, the model is enhanced. The model refinements are done in line with : 1) the model evolution ($m1$, $m2$, $m3$), 2) the analysis methods outputs (BAG^{wctt} and D^{nc}_{sw}) and 3) the feasible assumptions ($as1, as2$).

The first execution of the pivotal analysis method takes as input the data of the incomplete model ($m1$) and the deduced assumptions ($as1$). The first coarse-grained WCTT evaluation reduces the space of BAGs solutions for the VLs ($BAG^{wctt_1}_{vl_i}$) attached to the data flows in order to verify the latency constraints ($L^{m_1}_{C\_vl_i}$).

Thanks to the first WCTT computation, the initial model (m_1) is enriched with a crucial missing parameter : the BAG (m_2). We are then able to perform the complementary NC analysis that aims to evaluate the upper delay suffered in the switches for each VL ($D^{nc_1}_{sw\_vl_i}$). To execute the complementary analysis, the assumption model ($as2$) contains additional information about the network topology and the VLs routes. The NC computed latency is passed as a refinement parameter to the WCTT method so as to precise the BAGs sets. Taking into account the calculated $D^{nc_1}_{sw\_vl_i}$ reduces the set of eligible BAGs ($BAG^{wctt\_nc_1}_{vl_i}$) : solutions that do not meet the initial $L^{m_1}_{C\_vl_i}$ constraints are discarded.

At the third iteration, the model ($m3$) contains the refined BAGs ($BAG^{wctt\_nc_1}_{vl_i}$). It is then necessary to calculate the delay suffered in the switches for each VL ($D^{nc_2}_{sw\_vl_i}$) and refine the BAGs sets ($BAG^{wctt\_nc_2}_{vl_i}$). In our example, a new combined execution of WCTT and NC shows that a fixed-point is reached : the model m_3 cannot be refined anymore against the Bandwidth Allocation Gap if the input parameters and assumptions stay stable.

4. Conclusions and perspectives

Our first contribution (Section 2) dealt with the architectural description of an avionics system with AADL, combining the *functional*, *non-functional* and *platform* concerns. We extended existing patterns dedicated to ARINC653 systems to also model AFDX networks.

We then addressed (Section 3) the definition and evaluation of the AADL model components and their properties. We showed that dealing with the architecture description amounts to solve the dependencies between the AADL model concerns. Starting from an incomplete AADL model, we implemented a process combining two analysis methods to evaluate and refine the Bandwidth Allocation Gap parameter. This process combines early and in-depth analysis to help the designer to narrow the design space.

We believe it is necessary to formalize the use of analysis methods along with modeling languages in order to tackle the early system architecture definition and evaluation. In a previous work, we defined REAL [12] – Requirement Enforcement and Analysis Language. It allows one to define a set of predicates, and check whether a system satisfies them. We plan to extend REAL to define a library of predicates for *tools*. Such predicates would 1) define conditions under which a given analysis becomes feasible and 2) detect and exploit relationships between analysis methods. This would guide the designer to trigger relevant analysis as early as possible in the design flow.

The definition of those predicates, in the context of the FMS, is currently under implementation.

References

[1] G. Brau, J. Hugues, and N. Navet. Refinement of AADL models using early-stage analysis methods. Technical Report TR-LASSY-13-06, LASSY, University of Luxembourg, 2013. Available at `http://orbilu.uni.lu/`.

[2] D. Redman, D. Ward, J. Chilenski, and G. Pollari. Virtual Integration for Improved System Design. In *Proceedings of The First Analytic Virtual Integration of Cyber-Physical Systems Workshop*, San Diego, California, USA, Nov. 2010.

[3] P. H. Feiler and D. P. Gluch. *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language*. Addison-Wesley Professional, 1st edition, 2012.

[4] G. Lasnier, B. Zalila, L. Pautet, and J. Hugues. Ocarina : An Environment for AADL Models Analysis and Automatic Code Generation for High Integrity Applications. In *Proceedings of the 14th Ada-Europe International Conference*, Brest, France, June 8-12 2009.

[5] J. Delange, L. Pautet, A. Plantec, M. Kerboeuf, F. Singhoff, and F. Kordon. Validate, simulate, and implement ARINC653 systems using the AADL. In *SIGAda annual international conference on Ada and related technologies*, SIGAda '09, pages 31–44, New York, NY, USA, 2009.

[6] M. Lauer. *Une méthode globale pour la vérification d'exigences temps réel - Application à l'Avionique Modulaire Intégrée*. Thèse de doctorat, Institut National Polytechnique de Toulouse, Toulouse, France, juin 2012.

[7] Aeronautical Radio Incorporated. *ARINC Report 653P0 Avionics Application Software Standard Interface, Part 0, Overview of ARINC 653*.

[8] Aeronautical Radio Incorporated. *ARINC Report 664P7-1 Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network.*

[9] SAE/AS2-C. Architecture Analysis & Design Language V2 (AS5506A), January 2009.

[10] A. Al Sheikh, O. Brun, M. Chéramy, and P.-E. Hladik. Optimal design of virtual links in AFDX networks. *Real-Time Systems*, 49(3):308–336, 2013.

[11] M. Boyer, J. Migge, and M. Fumey. PEGASE - A Robust and Efficient Tool for Worst-Case Network Traversal Time Evaluation on AFDX. In *SAE AeroTech Congress & Exhibition*, Toulouse, France, October 18-21 2011.

[12] O. Gilles and J. Hugues. Expressing and enforcing user-defined constraints of AADL models. In *Proceedings of the 5th UML& AADL Workshop*, Oxford, UK, 2010.