Towards Automating Interface Control Documents Elaboration and Management

Hassna Louadah[1], Roger Champagne[1] and Yvan Labiche[2]

[1] Ecole de technologie supérieure (ETS), Montréal, Canada
hassna.louadah.1@ens.etsmtl.ca
roger.champagne@etsmtl.ca
[2] Carleton University, Ottawa, Canada
labiche@sce.carleton.ca

Abstract. Avionic systems have been migrating from the legacy federated architecture towards an integrated modular architecture (IMA). The IMA architecture replaces the equipment principle by a set of interoperable components (hardware and software). The interoperability between the integrated components requires a detailed specification and description of their interfaces, which, in the avionic domain, is usually written in Interface Control Documents (ICD). However, ICD creation and usage during the integration process is challenging. In fact, the two main problems with the usage of ICDs are the lack of a commonly accepted language to define and use them on the one hand, and the lack of tool support in their production and consumption. In this paper, we present our approach and methodology to overcome these limitations.

Keywords: Interface, Interface Control Documents, Modeling, DSL.

1 Introduction

Avionic systems are one of the major parts of an aircraft that leads to the growing increase of its cost (35 to 40% for civil aircraft, and over 50% for military aircraft) [1]. Up to the 90s, avionic systems followed a classical federated architecture where each equipment has its own avionic resources. However, due to the increasing complexity of such systems, unprecedented technological progress, and economic concerns, an innovative solution, developed and documented in ARINC-651 ”Design Guidance for Integrated Modular Avionics” [2] has been proposed.

As the legacy approach has reached its limits, the implementation of avionic systems for modern aircrafts converges towards the adoption of the Integrated Modular Avionic (IMA) architecture [1]. Thus, the aerospace industry is in the middle of a mutation, abandoning traditional federated architectures in favour of Integrated Modular Avionics. An IMA architecture replaces the equipment principle by a set of interoperable components (hardware and software). The interoperability between the integrated components requires a detailed specification and description of their interfaces (e.g. their boundaries and points of interaction). To this end, the specification of

an interface should include the description, at different levels of abstraction, of its physical and electrical characteristics, the communication protocols it uses as well as the data exchanged with it. The detailed specification of an interface in the avionic domain is usually referred to as an Interface Control Document (ICD).

The Aerospace industry is facing a real problem while building IMA systems using ICDs. This problem is related to (1) the manual creation of ICDs, (2) the manual use of heterogeneous ICDs by system engineers/integrators to create an IMA system, and (3) the manual verification and validation, especially verification of appropriate composition of components described by ICDs. These largely manual activities are tedious, very expensive, and error-prone. This is mainly due to the lack of a formal and common language and/or a standardized format or method to enable the creation in an unambiguous, complete, verifiable, consistent, and traceable manner of those ICDs, and their use during integration.

This research work aims to develop reliable and cost-efficient mechanisms to create and manage ICDs. The ultimate goal of the project is to provide innovative tools to system engineers to allow them to efficiently integrate equipments from different suppliers, described by their ICDs, to provide avionic systems in commercial aircrafts. To do so, our main idea consists in leveraging the strengths of model-driven engineering to the development, use and verification of ICDs, in order to: help unambiguous description and representation of interfaces and ICDs; enable automatic verification and analysis of interfaces; enable the automatic generation of human-readable ICDs in different formats.

The rest of the paper is structured as follows. In section 2, we provide a snapshot of recent works dealing with ICDs management. Afterwards, we present the proposed approach and methodology to achieve our goals (section 3). Finally, conclusions and future research directions are discussed in section 4.

2 Related Work

Despite the major role of the ICDs in the process of building avionic systems, only a few recent research works have addressed the problems related to their manual management. In this paper, we restrict our discussion to the most significant contributions which seem to be close to the solution we are looking for.

Qian Chen defined a taxonomy of interfaces under the form of an inheritance hierarchy [3]. The different dimensions that are accounted for in this taxonomy can be interesting for multilevel abstraction representation and complexity handling.

Rahmani and Thomson proposed a systematic methodology for modeling interfaces [4,5]. They have reused the principle of interfaces categorization and hierarchization to provide a unique interface architecture topology for two interacting subsystems. Thus, they defined a standard model for ICDs based on class diagrams. Based on this standard, the ICDs of the interfacing sub-systems can be defined separately, and then the consistency can be checked using the assembling rules defined manually for this purpose. Furthermore, the authors have proposed a conflict detection approach which enables the verification of interface definitions correctness and

consistency [6]. However, interface requirements specification, which constitute one of the main challenges of interfaces modeling and management, are not taken into account in their work. In fact, an interface specification should include the set of assumptions provided by the interfaced components and required for an efficient functioning of the component, and the set of guarantees that the component promises to other components via this interface.

The same authors proposed a computer aided methodology for defining and controlling subsystem interfaces [7], enabling a formal expression of interface requirements and mating rules of two subsystems. However, the interface is considered as a connection between two ports, and thus, could exist only by having knowledge about the two ends of such a connection. To overcome this issue, the authors have defined a domain ontology, however it is more suitable for hardware systems interconnections rather than software ones. Moreover, in avionic systems, we need to specify both hardware and software interfaces.

Pajares et al. [10] proposed a tool for ICD Management for embedded avionic systems. They defined a set of meta-models (data definition, data coding and communication architecture) for defining and managing ICDs in a formal way, capturing only a subset of the information that one typically requires in an ICD. In a similar way, Tapp defined a language to describe system interfaces and the various aspects surrounding their data exchanges [11], though without mechanisms to specify constraints on the interfaces. Luca de Alfaro et al. [12] on the other hand, focused only on constraints, defining sets of assumptions and guarantees on an interface's inputs and outputs variables respectively. In fact, the authors proposed a stateless interface language dubbed assume/guarantee and particularly, the notion of interfaces composability, formally verifiable, to check the interfaces compatibility of two components designed separately. Other works such as [9], [13] advocate the use of some tools but don't bring significant help to integrators in the ICD management process. Other authors proposed to use SysML (Systems Modeling Language) in the context of interfaces modeling, but, their approaches and consideration of interfaces do not meet our needs [14, 15]. In fact, they are useful for defining interfaces of an already specified sub-system's interactions and certification concerns, respectively.

In summary, existing works propose interesting, partial solutions to our problem but fail to provide a complete adequate solution due to one or more of the following reasons:

- Partial consideration. Some authors either overlooked the expression of interface requirements and constraints or restrict their definition of these requirements neglecting the interface characteristics and properties.
- Partial definition of the interfaces domain aspects such as hardware interconnection and/or data concepts only.
- Pairwise definition of interfaces. An interface is considered as a connection between two interacting components, whereas, in our project, we need to define each component interface separately to be able to verify interfaces compatibility when integrating these components.

Our solution will provide the following:

- Include hardware and software aspects of interfaces, requirements (as assumptions and guarantees) and all characteristics (physical and electrical) and constraints of both IMA and federated architecture's interfaces, since the aerospace industry is progressively migrating from federated to IMA architectures.
- Separate and independent definition of component interfaces to enable semi-automatic compatibility verification when integrating components.
- The interface specification will include the description, at different levels of abstraction, of its characteristics, the communication protocols used as well as the data exchanged through it.

3 Our Research Vision

Avionic systems and their hardware and software components interfaces must be well defined and have good specifications (i.e. unambiguous, complete, verifiable, consistent, and traceable). Interface specifications include different levels of abstraction, from different perspectives: e.g., physical and electrical, data messaging, and communication protocols. Capturing these various characteristics can be done by defining a Domain Specific Language (DSL).

A promising approach to address these concerns is the use of the Model-Driven Engineering (MDE) approach. In this context, the RTCA DO-331 standard [16] provides guidance for using model-based development and verification when designing avionic software (and systems). In fact, MDE promotes the definition of DSLs described using meta-models and enables model transformation which can be useful for verification and simulation [8]. The project will define a Domain Specific Language (DSL) for modeling interfaces and ICDs, and develop methods and tools enabling automatic verification and analysis of interfaces.

The first scientific challenge involves the understanding of what is needed when modeling components with ICDs. This can be performed in collaboration with our industrial partner which has extensive experience with the various types and formats of ICDs. Moreover, a domain analysis enables us to identify the relevant concepts (properties and meta-properties of ICDs) that should be considered in the context of ICD modeling. Based on the acquired knowledge, we will be able to create a meta-model, which is the most important ingredient for building a DSL, and define the relationship between the domain defined concepts. Such a meta-model will describe the abstract syntax of the DSL under construction [8]. The second scientific challenge is to build the DSL by reusing/combining existing modeling languages based on their closest concepts to DSL domain concepts. This should be done while following good meta-modeling practices while providing sufficient expressiveness, taking into account the possibility for future extensions. The third challenge consists in defining/selecting an appropriate concrete syntax for the DSL. As the concrete syntax expresses the user’s perception of the DSL, its definition should enable end users to easily read, write and understand the models [8]. The fourth challenge consists in building the DSL in such a way that allows us to implement and apply control mecha-

nisms on the interface and ICD models and so, enables semi-automatic control and verification of the interfaces.

In order to achieve our project objectives and overcome the challenges raised, we propose the methodology illustrated in Fig. 1. In Fig. 1 (a), we present our approach and planned methodology to achieve our research project goals, while Fig. 1 (b) depicts the DSL construction steps using the planned methodology. Our approach intends to enable the specification of an interface without any knowledge of how it will be used by other interfaces.

1. **Domain analysis**: to define concepts of "aspects of interest" related to the ICDs and interfaces. Considering a separate and independent definition of component interfaces, the domain concepts will include the hardware/software aspects of interfaces, the communication protocols used, the data exchanged through it as well as the requirements and all physical/electrical of both IMA and federated architecture's interfaces.
2. **Modeling languages study:** to select a set of candidate languages able to model the DSL domain concepts defined in the first step. The set of the studied languages includes, so far, AADL [17], SysML [18], EAST-ADL [19], HRC (Heterogeneous Rich Components) of SPEEDS project [20] and AUTOSAR [21].
3. **Case study modeling:** a feasibility study to check the possibility of reusing or extending existing modeling languages to build the DSL. The case study will be constructed in collaboration with our industrial partners and will include all relevant domain concepts defined in the domain analysis phase.
4. **Define a meta-model for the DSL:** by defining the DSL abstract syntax and its well-formedness rules. Based on results of the previous step, we will decide to either reuse or combine the existing modeling languages to define our DSL abstract syntax, and select the most appropriate ones to be considered. The definition of the DSL meta-model in the case of combination can be performed by transforming each modeling language's meta-model to the chosen one. This will be restricted to the relevant meta-models' parts only.
5. **Concrete syntax:** define/select an appropriate concrete syntax for the DSL and its semantics, which allows defining the meaning of the DSL elements and therefore enables automatic transformations [8].
6. **Verification and validation:** on the one hand, to continuously validate the meta-model with the domain experts, to validate models against their meta-model's prescribed constraints and rules via automated checks, and, on the other hand, to establish interface control and verification mechanisms to provide a semi-automatic control and verification of interfaces. The latter will be based on the separate and independent definition of component interfaces. This will provide an efficient way to detect any incompatibility between component interfaces during the integration process.

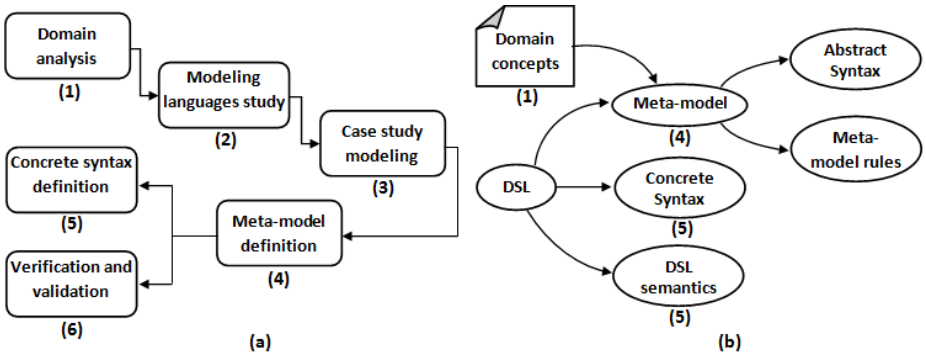

Fig. 1. (a) Methodology phases, (b) DSL construction applying our methodology

We are currently starting step 3. We have studied a set of modeling languages based on their meta-model and domain concepts, and then performed a high level selection of the potential candidates. Our study shows that the closest languages to our domain concepts are EAST-ADL and HRC, because of their hardware/software ports and data descriptions as well as requirements specification abilities. We are preparing the case study to verify the feasibility of reusing/combining the selected modeling languages. This case study will be constructed in such a way that all domain concepts (or at least the most important ones) will be taken into account.

4 Conclusion and Future Work

In this paper, we highlighted the main problems faced by the aerospace industry during the components integration process, based on Interface Control Documents (ICDs), to build avionic systems using an Integrated Modular Avionic (IMA) architecture. These problems are mainly related to the elaboration of ICDs and interfaces in the form of documents and their manual use and management. We have also outlined the drawbacks of the actual solutions and practices. Moreover, we have introduced our proposed approach which addresses these issues by using the MDE technology to efficiently create and manage the ICDs. Our solution promises to automate the elaboration and use of ICDs, and to provide mechanisms to verify, in a semi-automatic manner, the compatibility of the interfaces of the equipments to be integrated. Our future work will focus on the realisation and validation of the proposed approach.

Acknowledgements

This work was performed under the umbrella of a NSERC-CRD grant. The authors would like to thank NSERC, CRIAQ, and CMC Electronics, for their financial support.

References

1. Bieber, P., Boniol, F., Boyer, M., Noulard, E., Pagetti, C.: New Challenges for Future Avionic Architectures, AerospaceLab journal, (2012). http://www.aerospacelab-journal.org/sites/www.aerospacelab-journal.org/files/AL04-11.pdf, last accessed (September 2014).
2. AEEC: ARINC-651: Design Guidance for Integrated Modular Avionic, Aeronautical Radio, (1997)
3. Chen, Q.: An Object Model Framework for Interface Management in Building Information Models, (2007). http://scholar.lib.vt.edu/theses/available/etd-07262007-145650/unrestricted/Dissertation_Qian_Chen.pdf, last accessed (September 2014).
4. Rahmani, K., Thomson, V.: New interface management tools and strategies for complex products, International Conference on Product Lifecycle Management, (2009). http://www.mcgill.ca/plm2-criaq/files/plm2-criaq/rahmani_thomson_plm09-final.pdf, last accessed (September 2014).
5. Rahmani, K., Thomson, V.: Managing subsystem interfaces of complex products, International Journal of Product Lifecycle Management, 73 - 83, 1743-5110, (2011)
6. Onur, H., Rahmani, K., Thomson, V.: A conflict detection approach for collaborative management of product interfaces, Proceedings of the ASME Design Engineering Technical Conference, 555-563 (2010)
7. Rahmani, K., Thomson, V.: Ontology based interface design and control1 methodology for collaborative product development, CAD Computer Aided Design, vol. 44, pp. 432-444 (2012)
8. Stahl, T., Volter, M.: Model Driven Software Development Technology, Engineering, Management (2006)
9. Specht, M.: Creating, maintaining, and publishing an interface control document (ICD), AHS Technical Specialists Meeting on Systems Engineering 2009, vol.2, pp. 910-935 (2009)
10. Pajares, M., ngel, M., Daz, C.M., Pastor, I.L., Hoz, C.F.: ICD Management (ICDM) tool for embedded systems on aircrafts, ERTS2 (2010)
11. Tapp, M.: Automating system-level data-interchange software through a system interface description language, École polytechnique de Montréal (2013). http://publications.polymtl.ca/1256/1/2013_MartinTapp.pdf, last accessed (September 2014).
12. de-Alfaro, L., Henzinger, T.A.: Interface-based Design, Springer-Verlag, (2005)
13. L.Sergent, T., L.Guennec, A.: Data-Based System Engineering: ICDs management with SysML, ERTS2, (2014)
14. Fosse, E., Delp, C.: Systems engineering interfaces: A model based approach, IEEE Aerospace Conference Proceedings (2013)
15. Sabetzadeh, M., Nejati, S., Briand, L., Evensen-Mills A.H.: Using SysML for Modeling of Safety-Critical Software-Hardware Interfaces: Guidelines and Industry Experience, IEEE HASE, (2011)
16. SC-205: DO-331: Model-based development and verification supplement to DO-178C and DO-278A, Radio Technical Commission for Aeronautics, (2011)
17. Feiler, P., Gluch, D., Hudak, J.: The Architecture Analysis & Design Language (AADL): An Introduction (CMU/SEI-2006-TN-011). Software Engineering Institute, Carnegie Mellon University, (2006)
18. OMG: Systems Modeling Language, version 1.3, (2012). http://www.omg.org/spec/SysML/1.3/PDF, last accessed (September 2014).

19. EAST-ADL Association: EAST-ADL domain model specification, version v2.1.12, (2013). http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf, last accessed (September 2014).
20. SPEEDS Consortium: Speeds l-1 meta-model, (2009). http://speeds.eu.com/downloads/SPEEDS_Meta-Model.pdf, last accessed (September 2014).
21. AUTOSAR: Specification of the Virtual Functional Bus, version 1.3.0, (2012). https://www.autosar.org/fileadmin/files/releases/3-2/main/auxiliary/AUTOSAR_SWS_VFB.pdf, last accessed (September 2014).