On the link between Architectural Description Models and Modelica Analyses Models

Damien Chapon Guillaume Bouchez
Airbus France
316 Route de Bayonne 31060 Toulouse
{damien.chapon,guillaume.bouchez}@airbus.com

Abstract

When designing complex systems such as Aircraft, harmonizing the way we describe and analyze systems physical architectures is important in order to reduce costs, lead-time, and to increase systems maturity at entry into service. As Modelica has interesting multi-domain modeling capabilities, we define a harmonization approach that is based on the use of Modelica in an Integrated Development Environment.

Keywords: Modelica; UML; Domain specific language; Topcased; Physical Architecture;

1 Introduction

Airbus is a designer and integrator of systems. Airbus imagines the system' concepts, designs, and specifies the system so that suppliers can manufacture it. Those systems have the granularity levels as those considered within the ATA aeronautical classification of aircraft systems (e.g. ATA 27: flight control system, ATA 24: Electrical power generation system, ATA 32: Landing gear). Following a System Engineering Process (SEP), the stakeholders' needs, requirements, and constraints are transformed into a system architecture solution. For purpose of illustration, a generic SEP, the IEEE 1220 standard [1] is used in this document. The figure 1 depicts the sub processes of the IEEE 1220 SEP and shows how they iterate to produce a consistent set of requirements, functional arrangements, and design solutions. As depicted in this figure, one of the main outputs of this SEP is the verified physical architecture. Indeed during the synthesis and design verification phases, the functional architecture is translated into a physical architecture that provides an arrangement of system elements, their decomposition, interfaces (internal and external), and design constraints, to satisfy stakeholder expectations as defined in the requirements baseline.

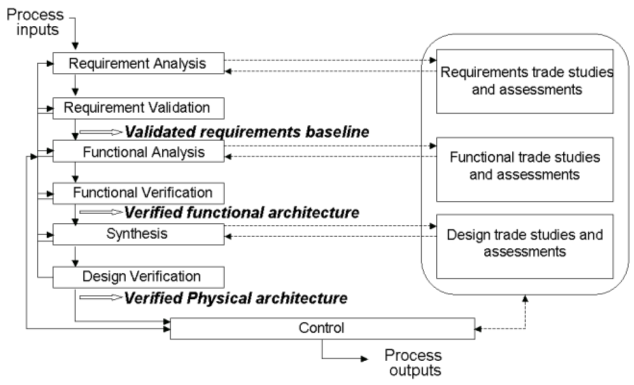
Process inputs
Requirement Analysis
Requirement Validation
Validated requirements baseline
Functional Analysis
Functional Verification
Verified functional architecture
Synthesis
Design Verification
Verified Physical architecture
Control
Process outputs
Requirements trade studies and assessments
Functional trade studies and assessments
Design trade studies and assessments

Figure 1 - IEEE 1220 Systems Engineering Process

Harmonizing the way we define and analyze system physical architectures is obviously a recurrent preoccupation when designing complex and critical systems because this is a mean to reduce costs, lead-time, and to increase systems maturity at entry into service. Modelica [2, 3] is a multi-domain and multi-paradigm modeling language for component-oriented modeling of complex systems that is well suited for physical and multi-domain physical analyses.

In this article we want to define recommendations in order to bring harmonization to the system physical architecture description and analyses process by promoting a model-based approach. Therefore this document is organized as follows. The second section is devoted to a deep explanation of the system physical architecture description and analyses phases in order to understand their particularities and to define recommendations for our model-based approach. In the third part we present some related works within the Modelica community and comment them with regards to our recommendations. Then in the fourth section we present the capabilities that need to be part of an Integrated Development Environment (IDE) in order to support our harmonization initiative. We also introduce Topcased, an integrated open source System/Software engineering toolkit that is a good candidate to support our work. Finally in the fifth section we present an example of a domain specific UML profile that we have created with the meta-modeling capabilities of Topcased.

2 System physical architecture description and analyses process

The figure 2 depicts more precisely the system physical architecture description and analyses process and exhibits the relations between the main products of this process.

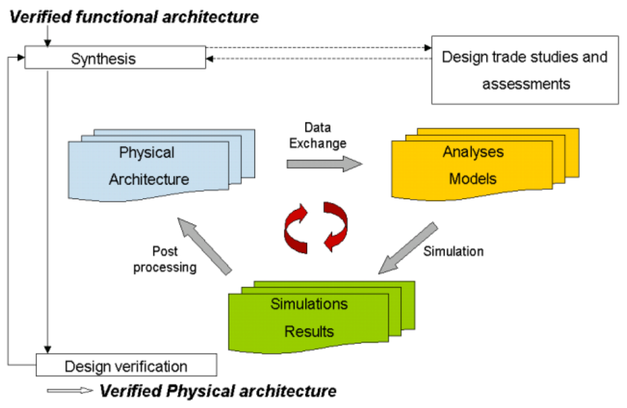

Figure 2 - System Physical Architecture description and analyses process

2.1 Separation between the system physical architecture description and analyses steps

As depicted in the figure 2 the physical architecture provides information (e.g. parameter, topology) to create the analyses models. Then simulations are run on the analyses models to get simulations results and exploit them to draw conclusions about the physical architecture. This process is iterated until the physical architecture satisfies all the requirements captured from the operational perspective (users' needs) and also defined in the functional architecture (e.g. safety, reliability, reuse, maintainability, performance, design constraints), with all the constraints imposed by the physical laws (e.g. aerodynamic forces, thermal dissipation, power consumption.) and with cost reduction constraints. Therefore the link between the architectural description step and the analyses step is very close. However it is better not to mix these two steps into the same modeling language or tools. Indeed their intended goals are not the same. The architectural description step defines the topological arrangement of the system components, how they are connected, and stores data, e.g. parameters, about the systems being designed whereas during the analyses step the information stored in the architecture is used to build analyses models in order to run simulations and to draw conclusions about the system physical architecture. This separation is sometimes very explicit; e.g. these two tasks are sometimes performed by different people or even subcontracted.

2.2 Physical architecture description

In order to identify good practices for the physical architecture description we use the IEEE Std 1471 [4]. This standard provides definitions and a metamodel for the description of architecture. The Figure 3 depicts the part of the metamodel that is relevant for our needs.

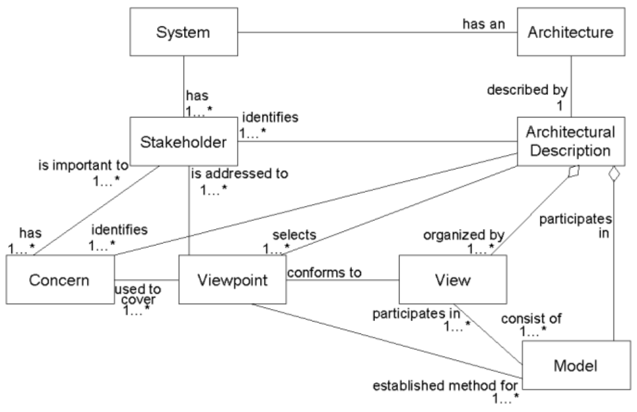

Figure 3 - IEEE 1471 conceptual model for architectural description

This metamodel state that a system architecture is described by an architectural description organized into one or more views and that each of them are conformed to a viewpoint. The central notion of viewpoint is defined as "the conventions by which a view is created, depicted, and analyzed. The viewpoint determines the languages (including notations, model, or product types) to be used to describe the view, and any associated modeling methods or analyses techniques to be applied to these representations of the view. These languages and techniques are used to yield results relevant to the concerns addressed by the viewpoints". This standard also recommends using the considerations and the concerns of the stakeholders as the inputs for the viewpoints selection. Obviously the systems designers are among the system stakeholders and one of their considerations is to have languages for describing the views that are customized for their job. A way of doing so is to encapsulate within the language some part of the domain knowledge. The domain knowledge is the good rules, good practices, and the experiences of the system designers in a technological field, a domain (e.g. a domain specific iconography).

2.3 Modeling languages for describing systems physical architectures

As explained above, modeling languages are needed in order to describe systems physical architectures. Depending on the systems being designed, the stakeholders and their concerns, different alternatives are available:

1. To use existing modeling languages. Among the most popular is UML, [5] (Unified Modeling Language), a standardized general-purpose modeling language in the field of software engineering. UML includes a set of graphical notation techniques to create abstract models of specific systems. Another solution when designing complex system is SysML[6] (Systems Modeling Language), a UML profile for systems engineering.
2. To customize existing modeling languages. When a specific syntax or a more precise semantic is needed for a specific system being designed, a solution is to use the extension mechanism of UML to create a domain specific UML profile. A profile is used to extend the UML metamodel by using three basic mechanisms: stereotypes, tagged values and constraints added in OCL (Object Constraint Language).
3. To create new Domain Specific Languages (DSLs). This solution allows customizing the modeling languages to cope totally with the needed specificity of each viewpoint. Creating a DSL can be worthwhile if the language allows expressing a particular type of problems or solutions more clearly than pre-existing languages would allow, and if the type of problem in question reappears sufficiently often.

However as defined in the IEEE 1471, the stakeholders to whom the architectural description is addressed are responsible for the choice of the modeling languages to be used. Whatever this choice is, what is really important is to have the modeling and meta-modeling framework to create, customize and/or use the chosen architectural modeling languages.

2.4 Languages and tools to support the analyses steps

Different analyses are performed during this process. We can classify these analyses into two categories:

- Trade studies: to evaluate different architectural alternatives against performance, cost, schedule, and risk implications. These analyses include parametric sensitivity analyses, Monte-Carlo analyses or optimizations.
- Verification activities: to ensure the conformity of the design with respects to functional architecture and the higher-level requirements. These analyses include direct simulations, formal proofs or robustness analyses.

In order to support all the analyses performed during the design of all the ATA systems of an aircraft, there exists a huge variety of system modeling languages and tools. We consider Modelica only as one of these modeling languages. We want to use the multi-domain modeling capabilities of Modelica everywhere it is possible in order to bring some harmonization to the wide variety of languages and tools. However let's remark that Modelica cannot obviously replace all the system modeling languages and tools used when designing complex and critical systems because it cannot support all the different analyses required when designing complex systems.

2.5 Recommendations for harmonizing this process

During the four previous subsections we exhibit some features that allow us to identify recommendations in order to define a model-based approach in an Integrated Development Environment (IDE) for harmonizing the physical architecture description and analyses process. The IDE should include modeling and meta-modeling capabilities to create, customize and/or use the chosen architectural modeling languages. Then, as Modelica can bring some harmonization, the IDE should include the capabilities to use Modelica model, either by including a textual editor, a graphical editor, and a simulator into the IDE or by allowing the use of an external Modelica tool. The IDE should also include the capabilities to use other analyses languages and tools that are relevant for the system being designed, or in other words the IDE should not be only Modelica-centric. Finally the physical architecture description and analyses steps should not be mixed. The IDE should rather provide or allow defining some links (e.g. Model transformation, data exchange) between the physical architecture description step and analyses step.

3 Related Work

Several attempts to integrate or link Modelica with UML or SysML have been launch within the Modelica community:

- ModelicaML by A. Pop and P. Fritzson [7];
- A mapping between SysML and Modelica by T. Johnson and C. Paredis [8];
- UmlH by C. Nytsch-Geusen [9].

These are three very interesting and promising works but from our viewpoint they don't yet satisfy all the needed features that we exhibit in the section 2.5. Moreover the authors of ModelicaML argue in the following paper [10], for a development direction of ModelicaML that creates a small core with well-defined semantics, instead of the current version that is based on an extension of SysML. We agree with the fact that UML is too big and semantically unsound to be used to describe efficiently systems physical architectures. We differ from the vision of ModelicaML because we do not think that an Integrated Whole Product Modeling based on Modelica is the right solution. Modelica should be considered as an analyses tool among others and we propose rather to have an IDE that contains the capabilities to define customized physical architectures languages (e.g. notations, data model, semantics, and domain rules) that could be formally link with analyses activities based on models expressed with Modelica and other analyses tools, in a model-based fashion.

4 Using the model-based capabilities and services of Topcased

Topcased [11] is an integrated open source System/Software engineering toolkit compliant with the requirements of critical and embedded applications. It covers the stages from requirements analysis to implementation, as well as some transversal activities like anomaly management, version control, and requirements traceability. Topcased principles are based on Model Driven Engineering and therefore provide:

- Meta-modeling capabilities to describe all the modeling languages in a common framework;
- A model bus to access easily to the various tools;
- Model transformations capabilities to relate the various models and adapt models to the various tools involved in a project;
- Generative programming capability to easily produce both textual and graphical model editors.

We focus in this article on Topcased because it owns all the model-based capabilities and services that are expected to deploy our harmonization initiative. What is important is the use of these capabilities and services, not the use of Topcased. However Topcased is a good solution because it provides a system-engineering platform that owns expected services for critical embedded systems, like configuration management and traceability.

4.1 Topcased as a Meta-modeling tool

Topcased relies on the Eclipse platform [12]. With regard to the four-layer meta-model architecture of the OMG's Meta-Object Facility [13], the M3 meta-modeling language used in Topcased is Ecore, provided by the EMF [14] (Eclipse Modeling Framework) project. Topcased is therefore strongly model-oriented. Indeed TOPCASED provides model editors, model checkers and model transformations capabilities, but is also itself based on modeling and code generation. With Topcased it is possible to use existing modeling language such as UML or SysML, customize domain specific UML profile, or create new DSL with their graphical editors using the EMF and the Graphical Modeling Framework (GMF)[15]. For illustrating the meta-modeling capabilities of Topcased, an example of a domain specific profile for the description of aircraft on-board power systems architecture is given in the section five of this article.

4.2 Providing a Modelica enabler and system physical analyses capabilities in Topcased

The meta-modeling capabilities can also be applied to provide editors for some of the analyses tools needed to perform trade studies, verification activities or other specific analyses on the systems physical architectures. Providing an enabler to Modelica in Topcased with graphical and textual editors and a Modelica simulator is a way to bring some harmonization to the physical architecture description and analyses process. In this perspective an interesting work has been done for the Modelica Development Tooling (MDT) [16]. MDT is a collection of plug-ins for Eclipse and can therefore be plugged into Topcased. It provides an environment for working with Modelica projects and integrates the OpenModelica compiler to provide support for various features, for example package and class browsing and code completion. However the MDT lacks for the moment a graphical editor for Modelica models.

4.3 Linking the physical architectural models to the analyses models

As defined earlier, we do not recommend mixing the physical architecture description step and the analyses step because these are separated tasks with different objectives. However as depicted in the figure 2 these two tasks are closely linked because the physical architecture provides information (e.g. parameter, topology) to create the analyses models and in return, simulations are run on analyses models during the analyses step to get simulations results and draw conclusions about the physical architecture. This process is iterative and therefore the link between the architectural models and analyses models is crucial. In this perspective Topcased provides a model bus that could be useful to access easily to the various analyses tools. Another potential idea is close to the mapping between SysML and Modelica realized by T. Johnson and C. Paredis. But as we mentioned earlier we don't want to impose in our framework a predefined architectural modeling language such as SysML. So the idea is rather to use the meta-modeling capabilities of Topcased in order to create domain specific UML profile. Then a mapping between these architectural modeling languages and analyses languages (e.g. Modelica) could be realized and finally the model transformation capabilities of Topcased could be used to link the physical architecture description step and the analyses step.

5 Illustrating the meta-modeling capabilities of Topcased - A UML profile for the description of aircraft on-board electrical power systems

In order to illustrate the meta-modeling capabilities of Topcased we have customized a UML profile for the description of aircraft on-board power systems architecture. The role of this profile is to store topological and parametric data that are relevant for the system stakeholders. This domain specific UML profile encapsulates a small part of the needed domain specific knowledge. Please remark that this profile is not intended to be usable as it is. It has only been created in order to illustrate how to customize a domain specific UML profile with the meta-modeling capabilities of Topcased and should be improved greatly to be usable. Our profile extends the UML metamodel by using three basic mechanisms:

- Stereotypes. These are extensions of already existing elements in order to define new types specific to the domain.
- Tagged values. These are extra properties that can be added to UML elements in order to specify additional information.
- Constraints. These enable to specify well-formedness rules and restrictions on model elements.

We define new types for the elements of the aircraft on-board power systems architecture by stereotyping the Class metaclass. Following the same idea we define new types for the topological connections between the elements of the architecture by stereotyping the Association metaclass. The figure 4 depicts the stereotypes used in our profile.

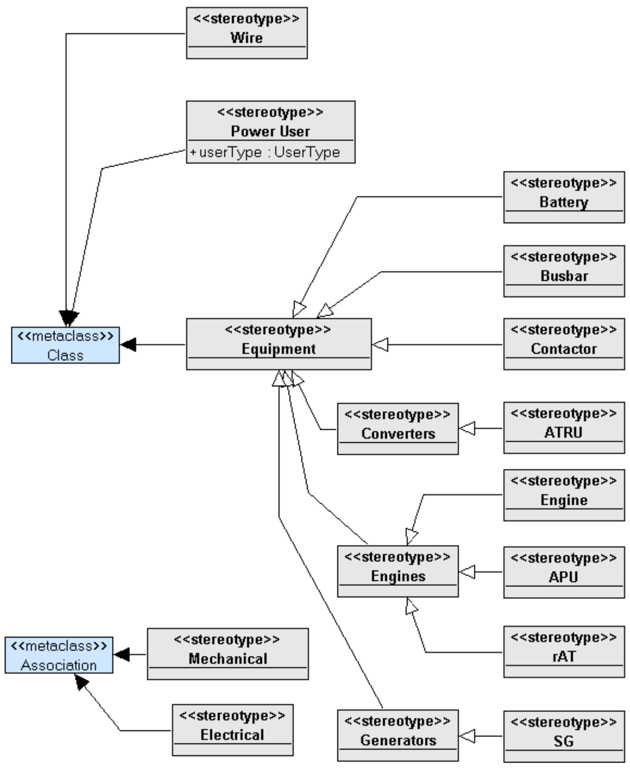

Figure 4 – New stereotype for the Aircraft On Board Electrical Power Systems architecture

Black arrows denote an extension of the targeted metaclass and white arrows denote inheritance between the stereotypes. A tagged value has been added to the Power User in order to add a new property for this new stereotype. Several others could have been added for the other stereotypes and shared by the inheritance mechanism between stereotypes. Then constraint written in OCL can be used to specify well-formedness rules. An example of well-formedness rule could be that you cannot draw an association between two classes of stereotype <<Engines>>. OCL can also be used in various ways to specify the stereotypes more precisely such as constraining property values or specifying dependencies between values of different properties of elements. Another mean in order to customize a domain specific UML profile is to adapt the visual aspects of the different elements with icon and symbol that are familiar and more intuitive for the systems stakeholders.

6 Conclusion

In this document we have focused on the system physical architecture description and analyses process. We have defined recommendations in order to build a model-based approach for harmonizing this process. As Modelica has interesting multi-domain modeling capabilities, this approach is based on the use of Modelica model in an Integrated Development Environment. However this IDE should also include the capabilities to use other analyses languages and tools that are relevant for the system being designed and should not been only Modelica-centric. We recommend also that the IDE include modeling and meta-modeling capabilities to create, customize and/or use architectural modeling languages in a model-based fashion with models transformations or model bus services in order to provide links between the physical architecture description step and the analyses step. As a consequence of these recommendations we have presented Topcased, an integrated open source System/Software engineering toolkit compliant with the requirements of critical and embedded systems. Indeed Topcased owns all the model-based capabilities and services that are expected to deploy our harmonization initiative. Finally we have illustrate the meta-modeling capabilities of Topcased with customized domain specific UML profile for the description of Aircraft On-Board Power Systems architecture.

References

[1] T. Doran. IEEE 1220: for practical systems engineering. IEEE Computer, Vol.39, No. 5, May 2006.

[2] P. Fritzson. Principles of Object-Oriented Modeling and Simulation with Modelica 2.1, 940 pp., Wiley-IEEE Press, 2004.

[3] The Modelica Association. http://www.modelica.org/

[4] R. Hilliard, IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems. IEEE. 2000.

[5] OMG. UML. http://www.uml.org/

[6] OMG. SysML. http://www.sysml.org/

[7] A. Pop, D. Akhlevidiani, and P. Fritzson. Towards Unified System Modeling with the ModelicaML UML Profile. EOOLT'2007. July 2007.

[8] T. Johnson, C.J.J. Paredis, R. Burkhart, Integrating Models and Simulations of Continuous Dynamics into SysML. Modelica 2008.

[9] Christoph Nytsch-Geusen. The use of the UML within the modelling process of Modelica-models. EOOLT.

[10] J. Guy Süß, P. Fritzson, A. Pop. The Impreciseness of UML and Implications for ModelicaML. EOOLT

[11] P. Farail, P. Gaufillet, A. Canals, C. Le Camus, D. Sciamma, P. Michel, X. Crégut, M. Pantel. The TOPCASED project: a Toolkit in Open source for Critical Aeronautic SystEms Design. ERTS 2006.

[12] Eclipse. http://www.eclipse.org/

[13] OMG's Meta Object Facility. http://www.omg.org/mof/

[14] Eclipse Modeling Framework Project. http://www.eclipse.org/modeling/emf/

[15] Graphical Modeling Framework. http://www.eclipse.org/modeling/gmf/

[16] H. Tummescheit. Design and Implementation of Object-Oriented Model Libraries using Modelica. Lund, Sweden: PhD thesis, Department of Automatic control, Lund Institute of Technology, 2002.