A Modelica-Based and Domain-Specific Framework for Electromechanical System Design

Damien Chapon*, Fabien Hospital†, Guillaume Bouchez*, Marc Budinger†
*Airbus Operation S.A.S.,
316 Route de Bayonne 31060 Toulouse,
†INSA de Toulouse
135 Av. de Rangueil 31500 Toulouse.
{damien.chapon,guillaume.bouchez}@airbus.com,
{fabien.hospital,marc.budinger}@insa-toulouse.fr

Abstract

A Modelica-Based Domain-Specific Framework for Electromechanical System Design was developed. The intended goal of this framework is to be used in early design phases in order to size physical architectures of electromechanical airbrake system. It has been developed using a generic methodology for the development of interoperable and model-driven system design frameworks. It is based on domain-specific modelling languages for the description of system architectures and relies on ModelicaML, a Modelica UML profile, to support system architecture analyses with the Modelica modelling language. Transitions between architectural description models and Modelica analysis models are realized through analyses-based model transformations.

Keywords: Modelica; Domain Specific Language; Model Driven Engineering, Electromechanical Actuator.

1 Introduction

To develop new generations of aircrafts which ensure safer flights with improved operations, new system architectures which encompass new technologies shall be developed. Moreover, aircraft development shall be realized in a shorter period and with a new complex industrial organization that enforces the links with the system suppliers. To face up to these technical and industrial challenges, more and more modelling and simulation are used during the aircraft development, from the preliminary and conceptual phase to the integration of systems, and at different levels from aircraft functional level, to detailed dynamical analyses of equipments. However, the use of modelling and simulation activities within such compartmented and distributed organization results in the application of several different and non-fully coordinated or optimized "model-driven" processes, methods, and tools to support the discipline of systems engineering.

In order to solve this problem, a generic methodology for the development of interoperable and model-driven system design frameworks has been created. In this paper we are not going to present the overall methodology, but rather its philosophy, and how Modelica is integrated and used in it. For the demonstration, we applied this methodology in order to develop a Modelica-based domain-specific framework for electromechanical system design. The intended goal of this framework is to be used in early design phases in order to size physical architectures of electromechanical airbrake system. This Framework is integrated in an Eclipse platform. It is based on domain specific modelling languages for the description of system architectures and relies on ModelicaML, a Modelica UML profile, to support system analyses with the Modelica modelling language. Transitions between architecture descriptive models and Modelica analysis models are realized through analyses-based model transformations.

2 Design Framework development methodology overview

2.1 Methodology's principles for collaborative and interoperable design activities

The design of a system physical architecture is an iterative process. It involves several interrelated sub processes to transform the system functional

architecture into a physical solution. The arrangement of these different sub processes are depicted in the following picture inspired from the IEEE 1220 standard [1].

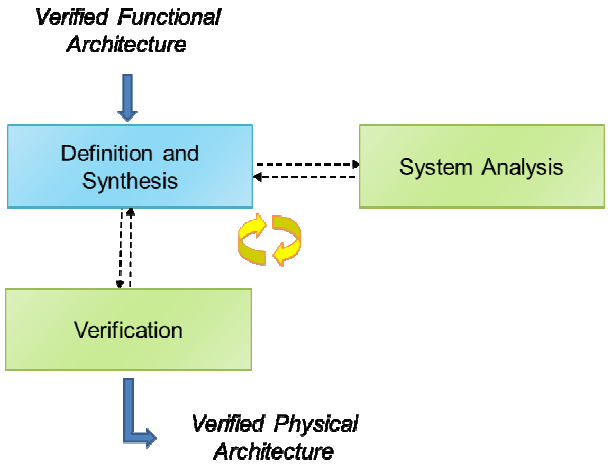

Figure 1 - System Physical Architecture design sub-processes

During the *Definition and Synthesis* sub-process, different alternatives of physical architecture are proposed and defined. The *system analysis* sub-process is used in support to evaluate and compare these proposed architectures. The selected architecture is then more precisely defined, optimised and sized in the *Definition and Synthesis* sub-process. Finally the *Verification* sub-process is used to verify and validate the chosen architecture in each of these domains of use. As illustrated in the Figure 1, the system physical architecture design process is an iterative set of back and forth between these different sub-processes.

Inside a system design team, the different activities of these sub-processes are distributed across different actors having different skills and roles. Depending on the system complexity and the organization these roles could be merged and their appellations could be different. However for complex systems these roles are clearly separated and it is therefore conceptually important to keep these roles separated in order to build optimized system design framework. To schematize, we can group these roles in three main categories:

- **System Architects**. They manage the overall system design activities. They define the different architecture alternatives. They ensure the consistence with the functional architecture and with other systems. They manage the detailed design and validate the choices and the results of the system designers and system analysts.
- **System Designers**. They define the architecture according concepts expressed by the system architect. They realize the detailed design, the optimization and sizing of the architecture;
- **System Analysts**. They realize the trade studies between the different architecture alternatives and/or the verification/validation analyses.

The different actors of the system design activities have to interact with the central point of all these activities, i.e. the system physical architecture being designed. Firstly to get the information that they need for the activities they are responsible for, and then to send back the results to the architecture definition after performing their activities. In order to get a collaborative design it is therefore crucial to give a central role to the system architecture and to allow the different actors accessing to the architectural data they need. In a classical document-centric process, the architectural data is stored in documents and is therefore not formalized and has to be interpreted by the different actors to realize their activities. Thus, with the growing importance of models and model-driven technologies in the system engineering processes, it could be useful to have a formal representation of this central architecture description in order to automate the access to the architectural data for the satellites design or analysis activities.

The Figure 3 illustrates the modular organization of the frameworks in order to get a collaborative design with dedicated environment for the different actors.

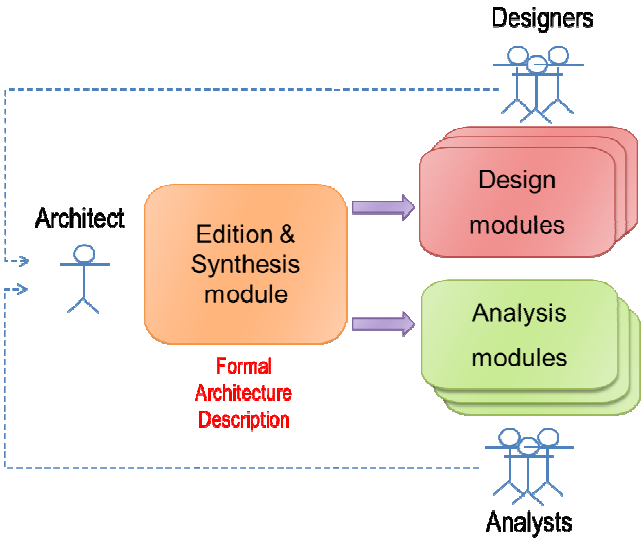

Figure 2 - Actors and modules interactions

Moreover, in order to get a functional and multi-system optimisation of the designed aircraft, the different teams responsible of systems architectures have to access to the architectural data of the others

systems their own system is related with. The architectural data is stored in the system architecture description, which should also be the link between the different teams. To support this data exchange we can imagine a common architectural database where every team could access to the others systems architectures descriptions through its own dedicated framework. The development of this database is out of the scope of the methodology. However this idea illustrates the need to build interoperable system design frameworks. We illustrated this idea in the Figure 3. This figure shows two dedicated design frameworks for two different system design teams. Each framework got a multi-view architectural editor for edition and consultation of system architectural data. Each team can access and edit the data of its system architecture through its own framework. Architectural data coming from others system can only be consulted. The consulting and editing views are therefore different for each team depending on their different representation needs. The design and analyses modules are plugged to the formal architectural data to access the data in order for the system designer or analyst to perform their activities. Then the results of these activities are used to mature the system architecture.

2.2 Model-driven development methods

The framework development methodology is model-driven. It concerns the two following main activities:

- Creation of architectural multi-views editors with dedicated Domain Specific Languages ;
- Development of gateway from system architecture description to design or analysis activities through model transformations.

With Domain-Specific Languages every actor of every system design activities could have access to the system architecture description in a view adapted to its needs. The creation of graphical Domain Specific Languages is composed of two main sub activities:

1. Creation and customization of domain specific languages with their own meta-models to capture the right knowledge and concerns of specific engineering domains;
2. Creation of customized graphical modelling editors. These graphical modelling editors bring the views that are needed by system architects to graphically create models conform to their engineering domain's meta-model.

Then, model transformations could be used to automate the exchange of data from architecture descriptive models to analyses models. If these satellite activities are not model-based, others models transformations such as automatic code generation or document generation are used to help the realization of these activities.

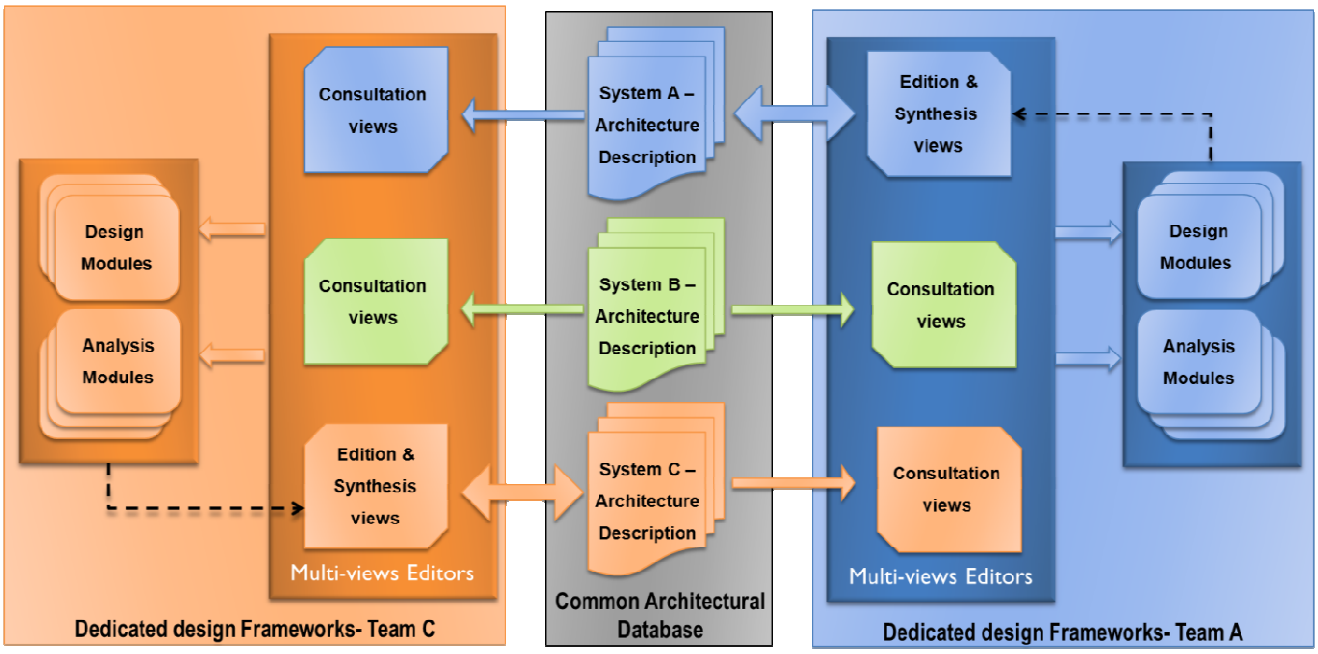

Figure 3 - Example of possible relationships between dedicated design frameworks through a Common Architectural Database

2.3 Model-driven technologies

Thus, the methodology relies on the creation of domain specific modelling languages in order to capture the specificities of engineering domains' knowledge. In this perspective it uses the Eclipse Modelling Framework (EMF) [2]. EMF is an Eclipse-based modelling framework and code generation facility for building tools and other applications based on a structured data model that can be specified in the Ecore language. Ecore is a variant of the EMOF, a subset of the OMG's MOF [3] standard, that is used to define simple meta-models using simple concepts. Moreover, EMF provides the foundation for interoperability with other EMF-based tools and applications. Interoperability between tools is a key driver in our methodology, so we selected ATL[4] (ATLAS Transformation Language), a model transformation language and toolkit, for the model transformations inside the use case presented in the next section. We selected Acceleo[5] for model to text transformation. For the creation of graphical editors we selected Obeo Designer [6]. Obeo Designer is an open workbench fully integrated with Eclipse. It is based on GMF [7] (Graphical Modelling Framework), that provides a generative component and runtime infrastructure for developing graphical editors based on EMF. Obeo Designer hides the complexity of GMF and offers the capacity to build quickly and easily customized graphical editors.

3 Use Case

3.1 Airbrake system presentation

In the present case study, an electromechanical actuator equivalent to a currently operating hydraulic airbrake actuator of a commercial single aisle aircraft is studied (Figure 4). The kinematics of the electromechanical airbrake is assumed to remain identical to the hydraulic one. This kinematics is based on a three rod mechanism, where the extension/retraction of the actuator linear jack drives the angular movement of the airbrake control surface. Accordingly, the transformations of motion are rotation (motor, gear), translation (screw, nut) and rotation (of the control surface).The airbrake motion ranges from 0 °to 50°. The moment of inertia of the airbrake introduces dynamic efforts that are not significant compared to the aerodynamic efforts. Specification imposes matching the dimensions of the current hydraulic actuator. Therefore, the variation of hard point positions was not addressed in this study. Furthermore, in power sizing phase, these points will be considered as perfect mechanical transmissions (e.g., no friction and no backlash). But with the sizing progress, tools are more and more detailed, and friction, stiffness and backlash will be taken into account to implement the virtual prototype.

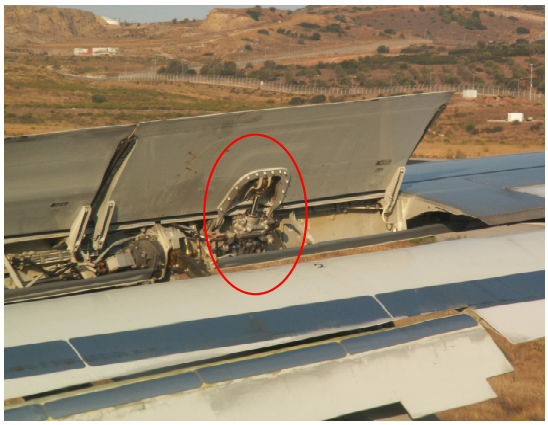

Figure 4 - Airbrake and its hydrulic actuator

3.2 Overview of the developed Domain-Specific Design Framework

For this use case, the development methodology introduced in section 2 has been used. A Modelica-Based Domain-Specific Framework for Electromechanical Actuators (EMAs) System Design was developed. The Figure 5 gives an overview of this Framework.

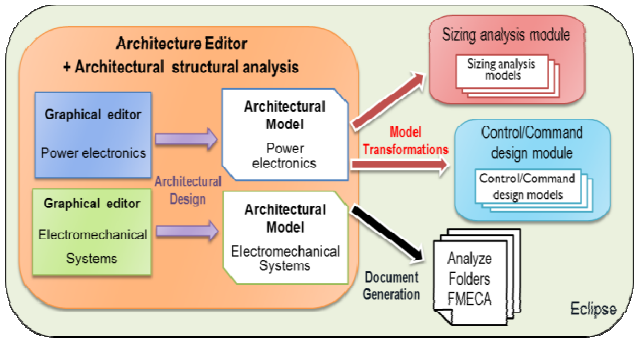

Figure 5 - General overview of the developed system design framework

As illustrated in the figure, the framework is composed of two system architecture graphical editors, one for each of the main domains collaborating for EMAs system design activities:

1. Power Electronics Systems, to control and make the correct conversion of the electric power coming from the electrical network to the electromechanical motor;

2. Electromechanical Systems, to transform the electrical power in mechanical power and to adapt the mechanical power to the application.

Then, model transformations and document generation capabilities have been developed to support partially or totally the following activities:

- Parametrical and structural analyses;
- Sizing analyses;
- Failure Mode, Effects, and Criticality Analyses (FMECA);
- Airbrake position control synthesis.

3.3 Domain Specific Modelling Languages for Electromechanical System Design

For the development of the two system architecture graphical editors, two meta-models were developed, one for electromechanical actuating system architecture and one for power electronics system architecture. Then, the domain specific graphical editors for each domain have been realised in Obeo Designer so that the system architects can build graphically the architectural models. In Obeo Designer we specified the graphical representations of each required concepts (components, ports and connections) of the two meta-models. We assigned a domain specific icon to each component and used generic graphical representations for ports and connections. Then we created the palette of components and connections, and we specified the way the model elements are created when using the palette. The figure 6 and 7 presents two architectures that have been realized with these graphical editors.

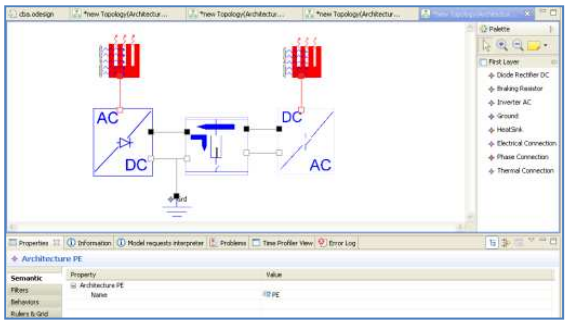

Figure 6 - Power electronics system architecture graphical editor

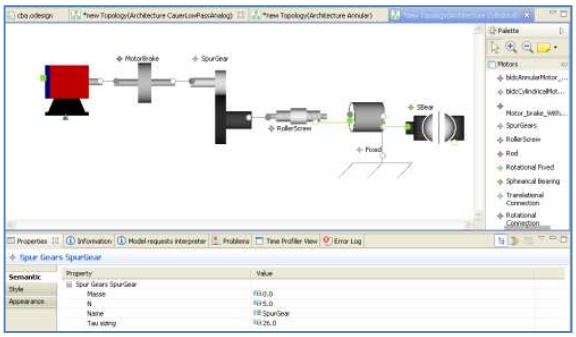

Figure 7 – Electromechanical system architecture graphical editor

As can be seen, the graphical editors propose a palette of components and connections that can be disposed on the workbench. The components' attributes can be changed in the properties view. Obeo Designer keep updated instantaneously the graphical view of the model and the model itself.

3.4 Integration of ModelicaML

Modelica [8] is a multi-domain modelling language for efficient component-oriented modelling of complex systems. The Modelica formalism can be used by several domains to perform physical analyses. Modelica is therefore well suited for multi-domain physical analyses and consequently we add it as an analyses module in order to add virtual analyses capabilities. The link between system architecture descriptive models and Modelica analyses models as already been studied in a previous work [9] and this integration is the concretisation of this work.

The Modelica module used in the framework is the ModelicaML [10] eclipse plug-in. Actually ModelicaML is a UML [11] profile. It extends a subset of UML in order to graphically define new Modelica models by using UML diagrams. TheseUML diagrams allow presenting the composition, connection, inheritance or behaviour of classes. Thus, it brings Modelica modelling capabilities into the framework. Further it relies on the OMG's UML, which is conform to the Meta-Object-Facility and therefore the model transformations between the system architecture descriptive models and the analyses models can be easily defined in ATL.

3.5 Analysis-based model transformations to Modelica Model

With the integration of a ModelicaML analyses module, the architecture descriptive models can be used as inputs for model transformations in order to create Modelica analyses models. In the system

engineering process, analyses are performed for specific purposes and therefore domain experts use adapted modelling languages to create the adequate analyses models.. And finally the domain experts dispose and connect the required components according the known system architecture to be analysed. In the same way, the developed framework relies on model transformations in order to automate as much as possible these analyses models creation steps.

According to the analysis to be performed, a specific ATL model transformation is coded. The Model transformation is performed in three steps as illustrated in Figure 8:

1. Components of the architecture to be analysed are directly mapped to the right analyses components available in a library dedicated to the analysis to be performed.
2. Parameters of each component of the architecture are mapped to the right parameters of their corresponding analyses models.
3. Connections of the architecture models are treated by the model transformation to automatically connect the components of the analyses models.

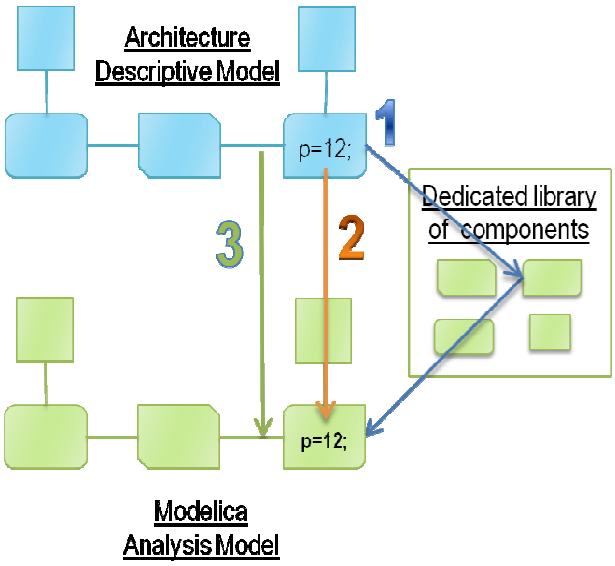

Figure 8 - The three steps of the model Transformations to get the Modelica analysis models

The Modelica analysis module is used in our use case to support sizing analyses and airbrake position control synthesis. As illustrated in the Figure 8, dedicated libraries for these two activities have been used. They will be presented in the next subsection. Then, a model transformation has been developed for each analysis type. Thus, from the same system architecture descriptive model different analyses models can be obtained as illustrated in the figure 9. This principle is called analysis-based model transformation.

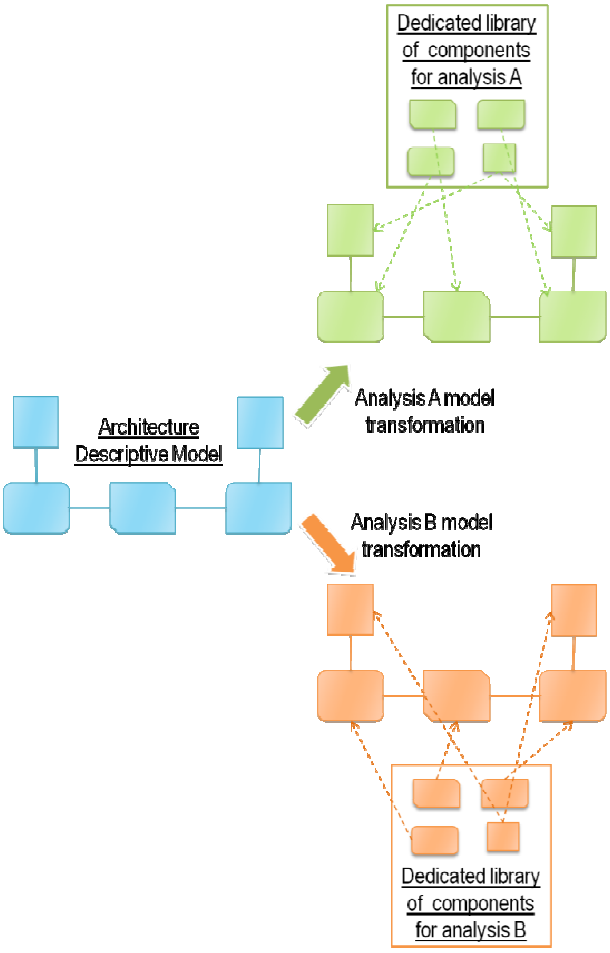

Figure 9 - Analysis-Based Model Transformations

3.6 Analysis-specific libraries

An in-house Modelica library for the preliminary design of electromechanical actuators is being developed since three years in the ICA laboratory of Toulouse [12], [13], [14], [15], [16]. This library uses non-causal models and inverse simulation. In fact, these Modelica models contain estimation, simulation and analysis models. Calculations of different parameters of simulation, sizing and comparison in function of definition parameters are separated in different sub-models. Moreover, according to the type of rated study, models own the necessary and sufficient characteristic sizes for analyze, in order to reduce the parameter number. So as to adapt itself to the various stages of the conception, each physical component has different analysis models according the analysis to be performed. The next figure shows the three analysis models of a component example, the spur gear.

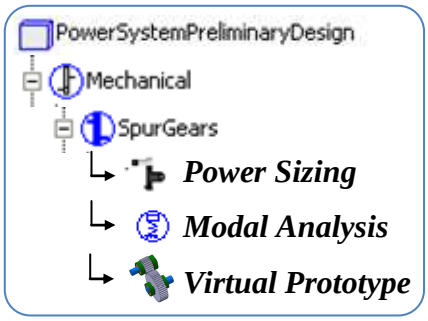

Figure 10 - Example of the three analysis models of the spur gear component

3.6.1 Power Sizing analysis

For the electromechanical actuating system sizing, each component is simulated and estimated with respect to the transmitted power in a backwards way starting from the load and its mission profile (effort and position time histories). This library takes into consideration operating areas, fatigue and reliability of components. Scaling laws approaches are implemented as a fast and efficient strategy in order to reduce the number of design parameters from the numerous model parameters [12]. These models require only inertia and efficiency as details.

3.6.2 Modal analysis for dynamic performances

For the control synthesis, the stability and the speed of the system can be studied with direct simulation. In this preliminary design stage, modal analysis gives the dynamic performance of the structure (pass-band and time response). Thus, more or less complex components models with or without linear friction and linear stiffness, allow the validation of components choice of the EMA..

3.6.3 Validation trough virtual prototype

In a preliminary design phase, components selection is finished and with the virtual prototyping, we start to go up in the V-cycle with more and more detailed models. CAD models of component allow the assembly of EMA elements and the analysis of complex components like carter. Non linearity is integrated in stiffness and in friction to introduce backlash and finer models. Finally, integrating components with complex characteristics, control system and 3D representation, a virtual prototyping is realized to analyse the virtual integration and to go far in synthesis of the global system. This kind of model is realized for a reducer box in the reference [17] where fine phenomena are modelled to implement a virtual prototype.

4 Design scenario and simulation results

A study was performed on the airbrake actuator use case. This study includes the following steps:

1. System architecture definition;
2. Power sizing analysis;
3. Modal analysis;
4. FMECA analysis (not presented).

The next two sections illustrate the analysis performed with Modelica.

4.1 Sizing analysis

The dimensions and mass estimation of one of architecture of the airbrake actuator are summarized in Table 1. Sizing of the mechanical parts and verification of the fatigue constraints was realized from the mission profile. For more precision in methodology refer to the Reference [15].

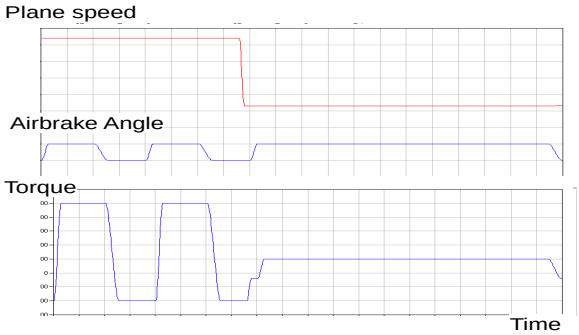

Figure 11: Mechanical mission profile

	Mass (kg)	***Length (mm)*** ***Diameter***
BLDC Cylindrical Motor	1.5	136 67
Brake	0.54	28 73
Spur Gear (ratio=5) + Ball Bearings	0.8	21 129
Roller Screw (pitch=4mm) + Thrust Bearing	1.38	172 53
Rod (hollow)	0.55	113 61
Spherical bearing (2 pieces)	0.14	70 35
Housing (Aluminium)	0.4	252 130
TOTAL MASS	5.3	
FINAL DIMENSIONS Distance between hard points Outer diameter		 312 129

Table 1: Components dimensions results

Indeed, components known in power view have modal analysis characteristics associated and a preliminary dynamic study can be realized. In direct simulation, with appropriated models of components, the system modal analysis can be done.

4.2 Modal analysis

For the airbrake actuator, specification imposes stability and time response. So, in a first time, the actuator slaving loop is simplified and composed to proportional controllers for speed slaving and position slaving, with constant disturbance which involves static deviation. Furthermore, the objective of this stage is analyzing natural performance of components in function of dynamic constraint applied.

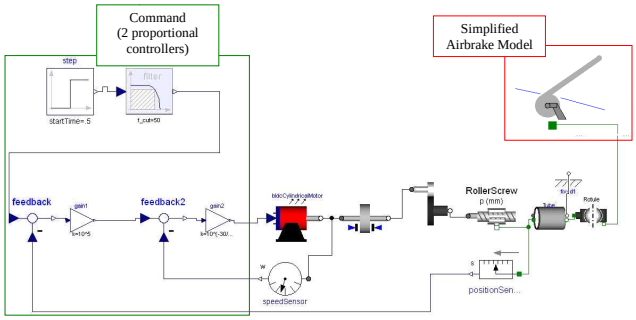

Figure 12 : Modelica slaving model actuator

First, the kinematic of the airbrake model was simplified changing 3D model by inertia and lever arm where aerodynamics load can be applied as a constant load. Secondly, the scaling laws were used to reduce parameter number [12], and assumptions were used to develop more or less complex electro-mechanical models:

- Motor: second order transfer function (cutoff frequency of 500Hz, damping coefficient of 0,7: typical of continuous current motor), torque source and inertia;
- Brake: only inertia (stiffness and backlash unknown);
- Spur Gear: inertia and transmission ratio;
- Roller/Screw: inertia, translation mass, pitch and take into account stiffness or not;
- Rod and Spherical bearing: translation mass and stiffness which can be take into account or not. Thus, anchorage stiffness can be included directly in the spherical bearing model.

The first stage consists in controllers' setting with frequency analysis in open loop. Thus, choice of controllers gain can be done in function of cutoff frequency of the system. Controllers can be adjusted after some iterations, the first on the speed loop, and then on the position loop.

Then, the answer to a step in input is studied in order to analyze the time response. The output and the input of the slaving are the position of the rod actuator. Of course, the static deviance is still present but the system is stable and time response is less than 1,5 seconds as specification requires (Figure 13).

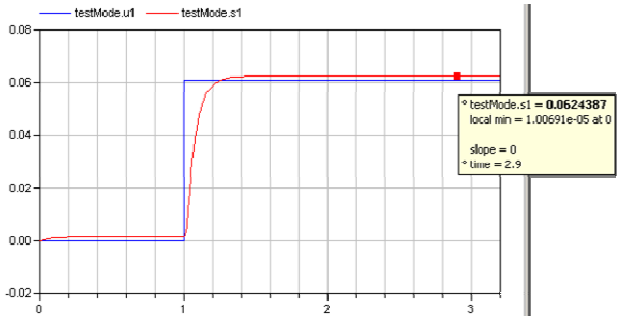

Figure 13: Position system answer to a step

5 Limitations

5.1 Modelica

For system design, Modelica is a very interesting modeling language because of its efficient physical modeling paradigm promoting:

- Object orientation, which allows the re-use of components in different projects with integration of elementary blocs.
- A-causality, thus different analyses are possible such as inverse simulation for power sizing and direct simulation for command synthesis.

However, a designer needs other tools and models to totally design and analyze system architecture, in particular in pre and post processing phases of the simulation:

- Estimation models, prerequisite in the simulation, inform the different necessary parameters for design. These models are static and algebraic. They are implemented in the previously presented library, with scaling laws in definition of some parameters [14] (power laws which reduce the number of definition parameters). For example, the addition of resolution's capacity of algebraic problems, as [18], would allow spreading the range of the treated problems of design.
- Post processing analysis models, which allow processing of results stemming from simulation, in particular in design way to validate components

choice. They are implemented for now, in the in-house library models, with means and integral calculation during the simulation that is weighing it down. It would be better if they could be realized at the end of dynamic simulation.

As an example, for the sizing of components of the system architecture performed in the previous use case the following models are needed. The different models of the spur gear component are separated and described in Modelica.

These aspects of Pre-processing and Post-Processing have to keep the oriented object logic of Modelica language to allow re-using of models. A solution could be the addition in Modelica norms of sections as *static model*, to develop calculation before issuing a dynamic simulation; and *post-processing*, to lead calculation after dynamic simulation; seemed to *equation* form on Modelica language.

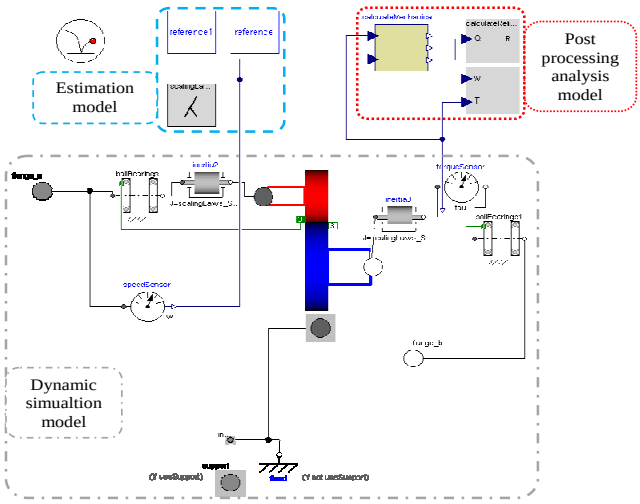

Figure 14: Spur Gear models

5.2 ModelicaML

Regarding the capabilities of the developed prototype, ModelicaML has some limitations:

- ModelicaML doesn't include yet a full simulation center, with an integrated GUI for launching of code generation, compilation, execution and displaying of simulation results on plots inside Eclipse. This means that the Modelica code generated in ModelicaML should at the moment be loaded inside a Modelica simulator outside the prototype. For this use case, Dymola has been used.
- ModelicaML does not allow the import of external Modelica code. This is a real problem to import and use existing Modelica libraries inside ModelicaML. For this use case we modelled directly in the ModelicaML graphical modelling language the libraries that were necessary.

However these limitations are planned to be removed in a near future by the ModelicaML developers.

6 Conclusions

The main principles of a generic methodology for the development of customized, interoperable and model-driven system design frameworks are illustrated in this paper. This methodology encourages the capitalization of engineering domains' knowledge in order to reuse it by promoting the use of analyses-based model transformations and domain specific modelling languages. It relies on a set of interoperable model-driven tools and languages including EMF, GMF, ATL, or Obeo Designer.

To illustrate this methodology, a domain-specific framework for electromechanical system design was developed. The intended goal of this framework is to be use d in early design phases in order to size physical architectures of electromechanical airbrake system. This framework uses the ModelicaML UML profile to support system architecture analyses with the Modelica modelling language. Model transformations from system architecture models to Modelica analysis models are performed through analysis-based model transformations. To this end, we used specific libraries dedicated to preliminary sizing and control/command of electromechanical system. However, the framework doesn't depend only on Modelica for system analysis. As an example document generation capability is implemented in Acceleo for Failure Mode, Effects, and Criticality Analyses. This documentation is not described in this paper but it represents a very important information source for designer as soon the start of design system. The developed framework is just a prototype and should be extended according the methodology principles with other architectural and analyses capabilities.

References

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

[2] Eclipse Modelling Framework Project, http://www.eclipse.org/modelling/emf/

[3] OMG: Meta Object Facility (MOF) 2.0 Core Specification, OMG Document formal/2006-01-01. (2006)

[4] Jouault, F., Kurtev, I., Transforming Models with ATL. In J.-M. Bruel (Ed.): MoDELS 2005 Workshops, LNCS 3844, p. 128 – 138, 2006

[5] Acceleo – http://www.eclipse.org/Acceleo/

[6] Juliot, E., Benois, J., La création de points de vue avec Obeo Designer, ou comment fabriquer des DSM Eclipse sans être un développeur expert? In Génie logiciel, GL & IS, p 49 - 54, 2009

[7] Graphical Modelling Framework., http://www.eclipse.org/modelling/gmf/

[8] Fritzson, P., Principles of Object-Oriented Modelling and Simulation with Modelica 2.1, Wiley-IEEE Press, 2004.

[9] Chapon, D., Bouchez, G., On the link between Architectural Description Models and Modelica Analyses Models. Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009

[10] Schamai, W., Fritzson, P., Paredis, C., Pop, A., Towards Unified System Modelling and Simulation with ModelicaML: Modelling of Executable Behavior Using Graphical Notations. Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009

[11] OMG: UML OCL 2.0 Specification, OMG Document ptc/2003-10-14. (2003)

[12] Liscouet, J., "Conception préliminaire des actionneurs électromécaniques - Approche hybride directe/inverse," PhD, Institut Clément Ader, INSA, Université de Toulouse, Chap.3, 2010.

[13] Budinger, M., Liscouet, J., Lefevre, Y., Fontchastagner, J., Abdelli, A., Allain, L.: Preliminary design of electromechanical actuators with Modelica. Proceedings of the Modelica 2009 Conference (2009)

[14] Budinger, M., Liscouet, J., Cong, Y., Maré, J.C.: Simulation based design of electromechanical actuators with Modelica. Proceedings of the ASME IDETC/CIE 2009 (2009)

[15] F. Hospital, M. Budinger, J. Liscouet, J-Ch Maré, “Model Based Methodologies for the Assessment of More Electric Flight Control Actuators“, 13th AIAA/ATIO Aviation Technology, Integration and Operation Conference, 13 - 15 Sep 2010 - Fort Worth, Texas.

[16] M. Budinger, A. Fraj, T. El Halabi, J-Ch. Maré, “Coupling CAD and system simulation framework for the preliminary design of electromechanical actuators”, IDMME Virtual Concept, 20-22 October 2010, Bordeaux, France.

[17] Angelika Peer, Physical-based Friction Identification of an Electro-Mechanical Actuator with Dymola/Modelica and MOPS, Modelica'2003 conference.

[18] GAMS, http://www.gams.com/