Co-Simulation of Complex Multi-Physics Systems Aeronautical example

Romaric Guillerm, Hamid Demmou, Alexandre Nketsa, Jean-Jacques Carrillo

Co-Simulation of Complex Multi-Physics Systems

Aeronautical example

Romaric Guillerm[1], Hamid Demmou[1, 2] and Alexandre Nkesta[1, 2]
[1] CNRS, LAAS, 7 av. du colonel Roche, F-31400 Toulouse, France
[2] Université de Toulouse, UPS, LAAS, F-31400 Toulouse, France
guillerm@laas.fr, demmou@laas.fr and nkesta@laas.fr

Jean-Jacques Carrillo
EDISON WAYS, CEEI Novalia 82, 20 place Prax Paris,
CS 80435, F-82000 Montauban, France
contact@edisonways.com

Abstract – Co-simulation is part of the current techniques to simulate multi-physics systems, which has a number of advantages compared to a complete simulation within the same multi-physics simulator. The principle is to connect existing dedicated simulators and to run in parallel these simulators allowing them to exchange data. A concrete example, based on the modeling of a new system of electric power distribution, illustrates this approach.

Keywords – Co-simulation, complex system, multi-physic system, electric distribution.

I. INTRODUCTION

Complex systems associate many components and several phenomena of different natures (Bar-Yam 2003). They involve different domain or scientific disciplines, each of which deals with one physical aspect of the system. For example, these physical aspects correspond to the science of solid mechanic, fluid mechanic, chemistry, electromagnetism, thermal, automation or computing.
Meanwhile, since several years the cost and time constraints imposed by the market on industrial projects induce a very restrictive prototyping. Less expensive and faster to implement, computer simulations progressively replaces this step. Many software tools exist for deep analysis and simulations in every scientific domain.
However, all the existing underlying interactions between phenomena make essential a simultaneous and common simulation of all fields. Various solutions exist today, including tools for multi-physics simulation (Istardi and Triwinarko 2013). Their principle is to integrate within a single software all phenomena, all possible physics. This approach provides excellent results. However, for our point of view it presents a number of disadvantages:

- Tools become very heavy because each tool has to deal with all the existing physics in the systems.
- They are not ergonomically suited to the working mode of a specific domain, because they have one common interface that integrates all the aspects.
- They force engineers to use new working tools, while they have specific tools well adapted to their own respective areas.

Addressing the problem in a different way, a second approach is possible: the co-simulation (Schmerler et al. 1995) (Glass et al. 2012), which allow engineers to keep their own specific simulation tools provided with new interfaces - software meaning (not graphical) - allowing them to communicate. The communication between different simulators allows to simulate the different models of the system, potentially related to different physics, as parts of the entire system. This approach is referred to co-simulation.

After this short introduction, the objective of this paper is to introduce the approach of co-simulation, followed by the presentation of the general principle in the second part. Then, a number of tools that we use are presented in the third part. The fourth part is devoted to the presentation of an industrial example with an application of co-simulation. It is about a new electrical power distribution system for aircrafts developed by the Edison Ways Compagny. The fifth and final section concludes the paper and presents some possible perspectives.

II. PRINCIPLE OF CO-SIMULATION

1. Co-Simulation - Definition

The first objective of the co-simulation was to do a mixed simulation between software and hardware systems (Yoo and Jerraya 2005). Now it is possible to simulate systems increasingly sophisticated and composed of different subsystems of various natures (electrical, mechanical, hydraulic, ...).
Used during the design, the co-simulation allows the validation of the complete system before its implementation, helping to correct design errors earlier.

From a global point of view, the principle of the co-simulation is the parallel execution of multiple simulators. Each simulator executes a system model established in a language specific to the concerned area. Then, all the models of the different simulators form the complete modeling of the system.

2. Communication between models

The co-simulation requires therefore an exchange of information between different simulators. To do this, a dedicated standard exists: the HLA (High Level Architecture) (Dahmann et al. 1997).

The HLA is a standard of interoperability for distributed simulation. It formalises the architecture and rules of interaction and it is based around a RTI (Run-Time Infrastructure). However, this solution is not the one that we have adopted.

In our work, we prefer to use the principle of co-simulation with data exchanges through a bus that is provided by a specialized tool: Cosimate, presented below. An illustration of this principle is shown in Figure 1.

3. Synchronization modes

Several methods of synchronization between models are possible for the co-simulation. Mainly, we distinguish the two following modes: event driven or synchronous.

The principle of the event driven mode is asynchronous communication between models. Each model can have its own clock, independent of those of the other models. Messages are sent or read by each model only when this one decide to do so. The overall behaviour of the co-simulation is then provided by the implemented communication protocols and their consistencies.

With the synchronous mode, the clocks of the different simulators evolve simultaneously: they are synchronized. Every data is exchanged at each co-simulation step. This mode is particularly suitable for co-simulations implementing various physical phenomena evolving in different simulators and possibly coupled. It is this second method that we implement in our case study.

III. TOOLS PRESENTATION

This part presents the different tools, simulators or languages that we will use in the fourth part to treat an industrial example. The first one is essential to our approach: it is Cosimate.

1. The bandmaster: Cosimate

Cosimate is a software tool developed by the Chiastek Compagny, allowing the co-simulation of a model set by establishing communications between several tool simulators or languages (Colenbrander et al. 2008) (Mitts et al. 2009). These simulators are either on a single machine or multiple machines distributed across a local or global network. The current version of Cosimate (2014.02-v7.0.0) supports many simulators (including Simulink and OpenModelica) and several languages (including C and Java).

As stated above, Cosimate therefore adopts the principle of communication by exchanging data through a co-simulation bus. All the models of the system which is desired to achieve co-simulation will be connected to this bus (see Figure 1).

2. OpenModelica

OpenModelica is an open-source tool for modeling and simulation, developed by the OSMC (Open Source Modelica Consortium). The models created with this tool are defined into Modelica language (hence the name of the tool), which describes a system as a set of equations. It allows modeling of complex systems, including mechanical, electrical, hydraulic or thermal. OpenModelica can itself be considered as a multi-physics simulator.
In our case, we use OpenModelica for its ability to model an electrical system.

3. Simulink and LabView

In order to control and view system information during the simulation, we initially implement a HMI (Human-Machine Interface) with LabView. This tool from National Instruments enables the creation or the quick prototyping of graphical user interfaces.

However, this tool is not directly supported by Cosimate. To use it with our co-simulation based around Cosimate, we used the feasible connections between:

- Simulink and LabView, through the SIT (Simulation Interface Toolkit) which is an add-on provided by National Instruments.
- Cosimate and Simulink.

4. C language

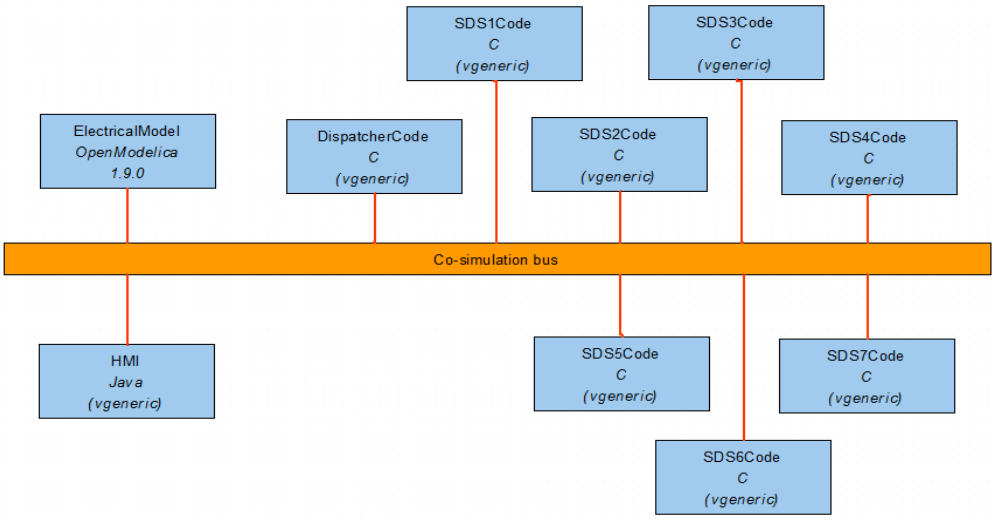

Figure 1: Co-simulation bus connecting different models (Cosimate view)

The C language is an imperative programming language widely used, especially for programming microcontrollers. We use the C in our case study for system management and

control software programs precisly supposed to be implemented on microcontrollers.

5. Java language

The Java language is inspired by the C++ object oriented language. We used it only for the implementation of a more accomplished HMI dedicated to the visualization and control of the studied system, using the Swing standard library.

VI. INDUSTRIAL EXAMPLE: NEW ELECTRICAL POWER DISTRIBUTION SYSTEM

1. Presentation of the System

We illustrate the approach of co-simulation with the use of various tools/simulators on a new electric power distribution system for airliners. This system completely rethink the architecture of the electrical distribution system and offers an innovative, lighter, more economical, safer, and more maintainable solution. Many patents protect this new concept named "Captain". They are held by Jean-Jacques Carrillo, founder of the Edison Ways Compagny.

***Note**: For privacy reasons, this article will not present in detail all the different parts of the system. However, it presents a general view of the functional principle, enough to understand the main features of the system.*

The Figure 2a shows the traditional architecture of the electrical system of airliners.

The Figure 2b exhibits the new system architecture using the "Captain" network. The general principle is to pool the electrical distribution in multiwire networks sized relatively to the sources (ie the generators). The primary distribution systems disappear and leave space to dispatchers, located in the center of the aircraft, and a set of secondary distribution systems scattered along networks.

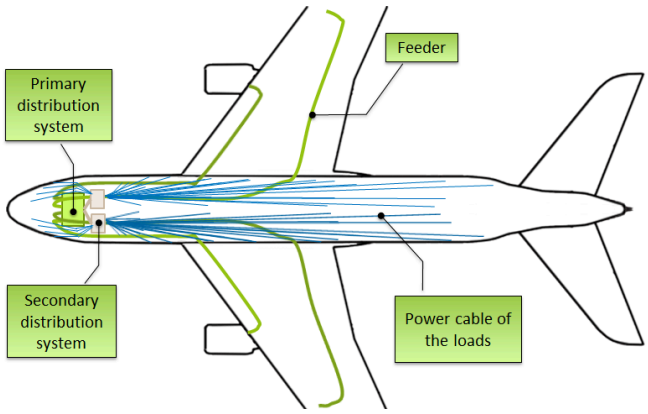

Figure 2a: Traditional architecture

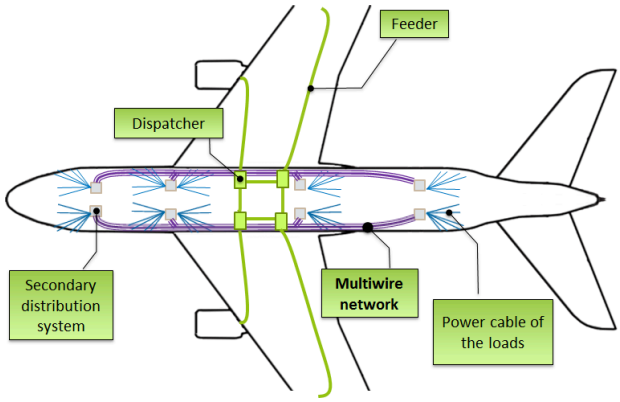

Figure 2b: "Captain" architecture

Dispatchers are responsible for monitoring the generator and network loads and for the power supply of the networks. The secondary distribution systems locally distribute electrical power to loads at around.

The figure 3 shows a simplified schematic view of the multiwire network. It shows, among other things, the dynamic allocation of loads over subnetworks linked to different sources. A network is actually composed of two subnetworks.

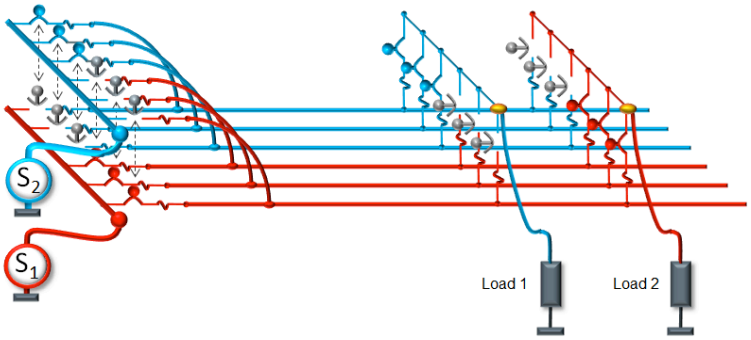

Figure 3: Multiwire network with dynamic allocation

The largest gain for the aeronautical industry is undoubtedly on mass. Indeed, the use of this system allows a gain of more than 60% of the wiring mass, which represent a relief of more than 2 tonnes for an aircraft like the A380.

2. The different models in interaction

In our case study, the configuration of the system that we chose is composed of 3 generators (twin engine configuration with APU) and 3 networks. Two networks are oriented to the front of the aircraft and are connected to two secondary distribution systems each. The third network is localized in the back of the aircraft and is connected to three secondary distribution systems. The figure 4 provides a representation of this configuration.

We connect 10 models to achieve a co-simulation of this system:

- 1 electrical model, made in OpenModelica,
- 1 C code for the management and control software of the dispatchers,
- 7 similar C codes for the management and control software of the secondary distribution systems (1 code for each one)
- 1 Simulink gateway linked to one LabView interface, or - depending on the choice of the GUI - 1 Java tool, for the user interface.

The figure 1 shows this set of models connected to the Cosimate bus in its Java tool version for the choice of the HMI.

a. Electrical model

The highest level of the electrical model is visible in figure 5. This model matches to the configuration fixed for our study ; it models all the electrical components and the connections from the generators to the loads, through dispatchers, multiwire networks and secondary distribution system.

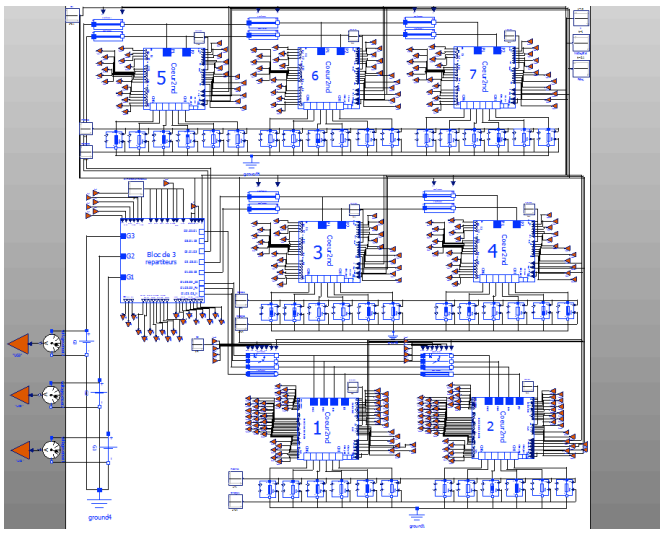

Figure 5: Electrical model

Considering the number of components of the system, the compositional aspect of the Modelica language has been widely praised here and used to define classes of objects: slices of circuit breakers, secondary distribution system, sub-dispatchers, dispatchers, etc.

Towards the co-simulation bus, data exchanged are:

- For those sent on the bus: the values of currents or voltages, mainly,
- For those from the bus: the states of the circuit breakers in order to control them, electrical parameters, or information relating to faults injected into the system.

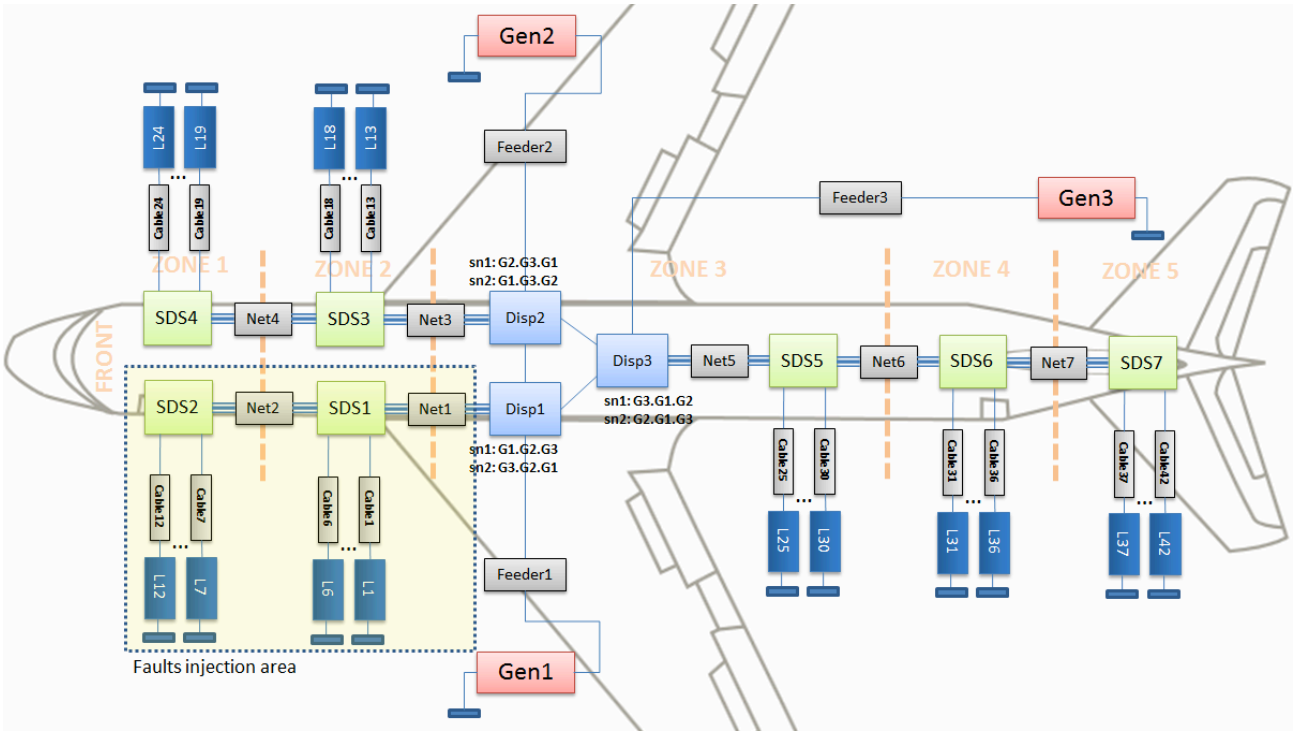

Figure 4: Electrical system configuration

b. Management and control software of the dispatchers

The level of abstraction chosen led us to encode a single management software to govern all of the dispatchers. The role of this program is mainly to monitor the loads of generators and networks. If an overload is detected, several strategies have been defined to return to an acceptable situation. This potentially involves dynamic reallocations or unballastings of loads that are not essential to a safe flight.

Details of these strategies are not described in this article, both for reasons of pages limitation and confidentiality.

This management software operates on data from the electrical model, passing through the co-simulation bus. Then, it makes decisions that determine the state of the circuit breakers at the dispatchers level or control orders sent to the secondary distribution systems.

c. Management and control software of the secondary distribution system

Each secondary distribution system is provided with a management software. This program is designed to manage the connection or disconnection of the loads associated to the secondary distribution system. It works autonomously but also receives orders from the management software of the dispatchers that must be treated. In normal operation, its role is mainly to balance consumption between the two subnetworks to which it is attached.

To work, this software uses also values provided by the electrical model transmitted trough the co-simulation bus, in addition to the orders from the software of the dispatchers that it receives as input. The decisions concern the status of the circuit breakers at the secondary distribution system level.

d. Human-Machine Interface (HMI)

As previously announced, two versions of the HMI were implemented.

The first one was created with the LabView software, dedicated specially to the fast development of user interface (see figure 6). The advantage here is the extremely short time and ease of development with such tool.

However, disadvantages of this solution are:

- The fact that LabView is a commercial tool,
- The need to use Matlab/Simulink - another commercial tool - to connect the interface to the co-simulation bus of Cosimate,
- The opportunities in terms of HMI and ergonomics, less widespread and sophisticated than with a pure programming language.

These disadvantages have led us to perform a second interface in Java. Nevertheless, the interest of using LabView in the early stages of the project was still not negligible, as it allowed to have very quickly a functional interface that was used for the first versions of the simulator. The work effort was then turned to the electrical modeling and the algorithms implemented by the different software.

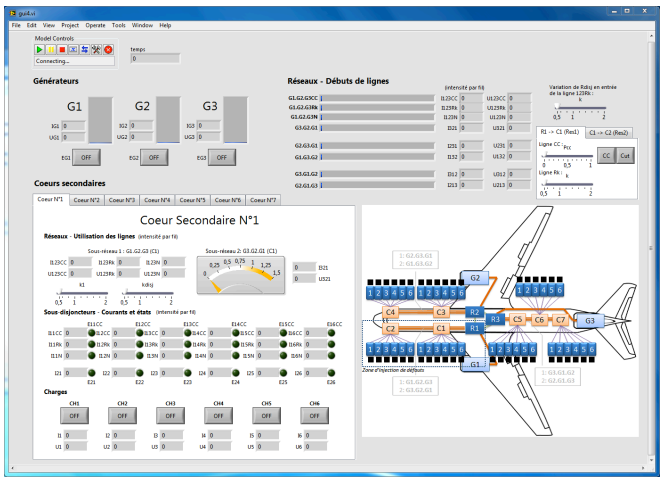

Figure 6: HMI implemented with LabView

Therefore, the second interface has been implemented entirely in Java. It includes many more features, for example the ability to define in advance scenarios (including both dated events or random events associated with probability distribution), to save them, or to execute them. This Java interface also provides an animated and interactive aircraft view, which brings a better understanding of the system and its operation. The figure 7 shows the main view of this interface, under the "aircraft" tab.

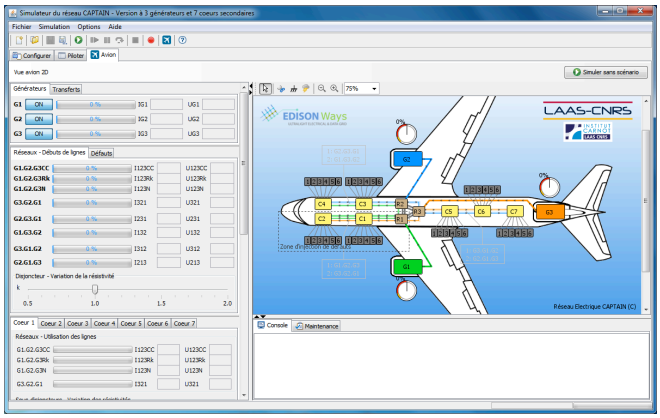

Figure 7: HMI implemented in Java

3. Co-Simulation and Results

Different scenarios have already been tested using this simulator around the co-simulation bus of Cosimate.

a. Running one scenario

To start a co-simulation with Cosimate, the procedure is as follows:

- Open the project in Cosimate (see figure 1), and start the co-simulation,
- Then, Cosimate will, for each model and based on the project parameters, run models automatically or prompt the user to do so. In this step, it is therefore to run models on Cosimate requests.

In our case study, for example, we set the co-simulation so that the various C codes start automatically. It remains the

responsibility of the user to run the simulation with OpenModelica and under the Java HMI when requested by Cosimate.

In order to run a predefined scenario, simply select it before in our Java HMI. This scenario will be used for the co-simulation, without preventing a possible interaction of the user who can inject new events on the fly.

b. Results analysis

Once the co-simulation is finished, many results can be analyzed:

- Traces of the Cosimate tool,
- Data of the simulators,
- Log files of the codes.

For example, the OpenModelica simulator allows to plot the curves of the evolution over time of any parameter of the model, like the current in a wire of a network (figure 8) or the one delivered by a generator (figure 9). The management software have been coded to keep track of their states and their decisions in files. These traces are very useful to test and validate the developed and implemented strategies, and also to validate the functional aspects of the new architecture of the electrical distribution system inside an aircraft.

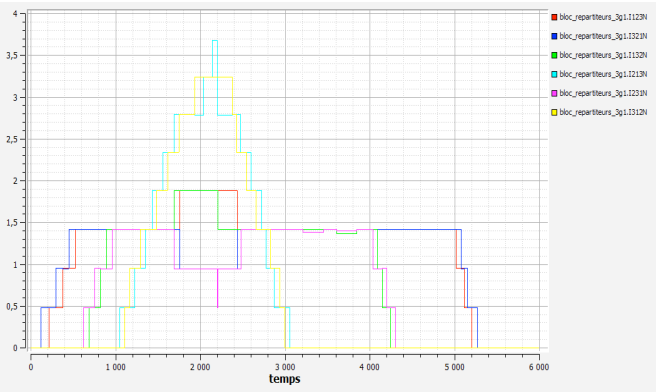

Figure 8 : Currents in the subnetworks

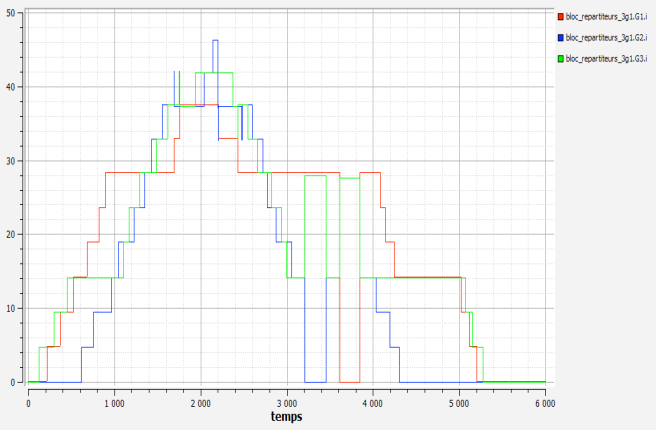

Figure 9 : Currents delivered by the generators

V. CONCLUSION

The objective of this article was to present a framework for the co-simulation of complex multi-physic systems. The principle here is the use of a co-simulation bus, using the Cosimate tool, alternative to an HLA-based approaches.
The use of this approach in an aeronautical project showed its feasibility and interest. The models are made with specific tools in the preferred environments of the engineers. All models aggregated around the co-simulation bus then allow a co-simulation to validate and verify the system. Moreover, it is possible to refine and reduce the level of abstraction of some models without jeopardizing the whole simulator and architecture.

REFERENCES

Bar-Yam, Y. 2003. “Dynamics of Complex Systems”. *The Advanced Book Studies in Nonlinearity series*, ISBN 0813341213.

Istardi, D. and A. Triwinarko. 2013. “Induction Heating Process Design Using COMSOL Multiphysics Software”. *TELKOMNIKA (Telecommunication Computing Electronics And Control)*, 9(2), pp.327-334.

Schmerler, S.; Y. Tanurhan; and K.D. Muller-Glaser. 1995. “A backplane approach for cosimulation in high-level system specification environments”. *Design Automation Conference, 1995, with EURO-VHDL, Proceedings EURO-DAC '95., European*, pp.262-267.

Glass, M.; J. Teich; and L. Zhang. 2012. “A co-simulation approach for system-level analysis of embedded control systems”. *Embedded Computer Systems (SAMOS), 2012 International Conference*, pp.355-362.

Dahmann, J.S.; R.M. Fujimoto; and R.M. Weatherly. 1997. “The Department of Defense High Level Architecture”. *In Proceedings of the 29th conference on Winter simulation (WSC '97), Sigrún Andradóttir, Kevin J. Healy, David H. Withers, and Barry L. Nelson (Eds.). IEEE Computer Society*, Washington, DC, USA, pp.142-149.

Yoo, S. and A.A. Jerraya. 2005. “Hardware/software cosimulation from interface perspective”. *Computers and Digital Techniques*, IEE Proceedings, vol.152, no.3, pp.369-379.

Colenbrander, R.R.; A.S. Damstra; C.W. Korevaar; C. A. Verhaar; and A. Molderink. 2008. “Co-design and Implementation of the H.264/AVC Motion Estimation Algorithm Using Co-simulation”. *Digital System Design Architectures, Methods and Tools, 2008*. DSD '08. 11th EUROMICRO Conference, pp.210-215.

Mitts, K.; K. Lang; T. Roudier; and D. Kiskis. 2009. “Using a Co-simulation Framework to Enable Software-in-the-Loop Powertrain System Development”. *SAE World Congress & Exhibition*, Detroit, Michigan, United States of America.

Behr, R.; J. Brown; and V. Baumbach. 2011. “Aircraft Thermal System Simulation - Hydraulic and Fuel”. *3rd International Workshop on Aircraft System Technologies*, Hamburg, Germany.