Product line engineering methods to support model reuse

Jean-Luc Marty
System Engineering Expert
Airbus Defence and Space

Alexandre Cortier
MOFLT champion
Airbus Defence and Space

Jonathan Ly
Intern/Student
Airbus Defence and Space/ IPSA

Abstract

This paper proposes a solution to cope with the re-use of models and requirements from one project to another. The proposed solution relies on the modelling of a generic spacecraft platform and its main variability points. This approach allows initializing a new model and its associated requirements as a variant of the 150% generic platform model. It contributes to objectives O3 and O5 of the MBSE 2022 workshop

Introduction

This paper proposes a solution to the need expressed by several system architects to automate the reuse of engineering information and definitions from one project to another by taking into account the specificities of their own project.

To reach this objective, we first present the need to reuse. In a second time we detail why and what is the constraint starting point, followed by a description of our proposition to reach the objectives, and to finish we present an example of usage in a project.

Need to reuse

Today on a process based document production a lot of engineering information is reused: chapters, schemas, requirements, that are adapted to a new project. This avoids reuse by capitalization and the blank page when a project start. It is a fact and is well appreciated by architects. So if we expect to push those actors to agree to a model oriented process, we shall provide the same capability: propose a set of reusable models, easy to integrate in a project and which could be easily adapted to its context. Reusable models can also bring other advantages like capitalizing formal definition, increasing design reactivity… But the first perimeter to cover is to at minima cover what is done today.

Airbus defined a modelling framework named R-MOFLT© (**R**equirement, **M**ission, **O**perational, **F**unctional, **L**ogical and **T**echnical architectures) to support system engineering. This framework is now used on several operational projects (PILOT, ARIEL, MSR-ERO). In addition to capturing the system architecture, the framework allows to perform traceability between model artefacts and textual requirements. We can provide a quick summary of R-MOFLT©

Constraint starting point: R-MOFLT© approach and usage

The R-MOFLT© Architecture Framework is a generic MBSE Architecture Framework at Airbus Group level explaining how to apply SE Technical processes based on the DDMS SECAM data model. This framework supports Systems Architects & Engineers to define a system architecture. The MBSE approach is based on a progressive, logical and structured approach in several layers, enabling a progressive and consistent definition of the system, from the problem space definition to the solution space definition. In more details, the R-MOFLT© approach covers the design of System of Interest through the Mission & Operational Analysis and Functional, Logical and Technical Architectures

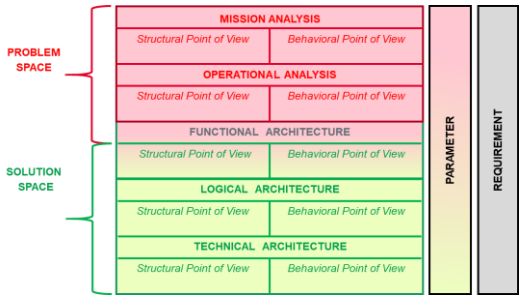

Figure 1: MOFLT layers

- ✓ **Mission Analysis**: WHAT is the problem that we need to solve, and WHAT are the potential ways of solving it?
- ✓ **Operational Analysis:** WHAT the System of Interest will do to contribute to the mission? What is the context of the System of Interest?
- ✓ **Functional Architecture**: HOW the System of Interest will work to meet expectations?
- ✓ **Logical Architecture**: HOW the System of Interest is organized? (Abstract Component)
- ✓ **Technical Architecture**: HOW the System of Interest will be implemented?

Solution proposed

Our objective is to propose a way to easily reuse models or parts of models in operational projects, based on MOFLT©. The main question to answer is then: "how to reuse models **easily**". Copy/paste could be one solution but with huge adaptation for each specific concept. Defining what could be reused and how to perform the reuse was studied with project actors and we extracted three main ideas:

1. If we consider the platform, we have the capability to define a generic one **with variability depending on performance constraints linked to the project**
2. The platform has a set of **functions systematically reused** with communality between projects and a limited variation perimeter by **staying at the right level of functional decomposition**. The specificities appear when we deal with technical components and performances
3. By staying at the good level of definition it could be easy to reuse some generic definitions. Namely, the functional and logical level of description seem appropriate if and only if we don't dive too deep in function decomposition.

In the context of variability management, Product Line approach seems the natural way to cope with re-use. This paper does not intend to talk about the constraints and added values of a product line approach but rather to identify how product line concepts (variability management) could bring added value for model reuse.

The proposed solution for model re-use based on variability is implemented in two main steps:

1) We chose to use only the functional and logical levels of R-MOFLT© to Model a generic spacecraft platform at a high level of granularity because:
    a. The M (for Mission) layer managing mission concept, Mission capabilities, and mission phases is really specific to the spacecraft mission so cannot be considered as generic.
    b. The O could be considered as generic at platform level but for only for specific types of Spacecraft (LEO or GEO activities). This layer manages the definition of operational capabilities and operational phases defined through a behaviour composed of operational tasks associated to a SOI to support mission concepts.
    c. For the T, it is possible to propose a reusable technical view based on a catalogue of components. This last proposition can be done only if, at company level, a real product line engineering approach based on physical architecture reuse is defined. At the moment, the technical level is very close to the physical components selected by the project. The system model at technical level shall thus be implemented by the project, based on the functional and logical definitions extracted from the generic platform model.
    d. Requirement reuse is taken into account in our approach but not mature enough to be presented in this paper.
2) Variability usage to allow the definition of a model which encompasses the classical architecture solutions used and capabilities to extract the expected solution for a project (Main principle of product line approach but limited to engineering model).

With the selection of the layers, Functional and Logical architecture definition, we have clarified with end users the need of expression requested through the model (model is done to cover objectives).

The following needs for function usage have been captured based on concepts managed by functional layer (Function, Functional exchanges):

- ✓ Define functions independently of the physical implementation. The functional architecture aims to :
    - Identify what the system shall do
    - Identify all functional exchanges between functions
    - Provide means to allocate functions in different ways depending on the technical architecture
- ✓ Hierarchical approach which allows to:
    - Manage complexity
    - Stop functional decomposition as soon as sub-functions can be allocated to logical components
- ✓ Consider all functions used on our design
    - Functional description shall have the right level to be generic
    - Functional description shall take into account all possible usages
    - Functional description shall cover all common needs

And the following needs for logical architecture usage based on concepts managed by the logical Layer (logical component function allocated to component, logical connections supporting functional exchanges):

- ✓ Define logical components with the main target being the end users' interests. The components shall express a concept manipulated by the system architects. These components also shall embed a set of functions (ex : Latching current limiter (LCL))
- ✓ The model shall represent functions hosted by components with their "interface end" supporting functional exchanges
- ✓ The Logical architecture describes end to end exchanges and shall be agnostic to the communication means
- ✓ Consider all components and variability used in our design
    - Alternatives between logical components hosting the same functions with different ways of implementation
    - All necessary logical components for the system designs

To cover the first point (Modelling of a generic spacecraft platform), the generic platform model covering all possible functions has been structured following the main functional chains of a spacecraft design that generally maps with engineering disciplines. For each functional chain, a model has been developed with architects and experts of the disciplines. Each of these sub-models is managed in configuration as well as the global generic platform model which is used by all of them.

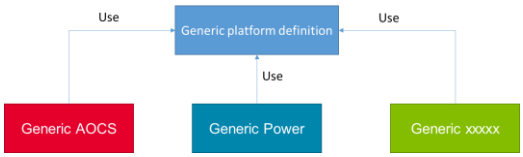

Figure 2: Generic platform composition

The generic platform model contains all the common artifacts and particularly the types used between models at interface level. The first level of functional decomposition is the one used to define the discipline model. (Figure 2)

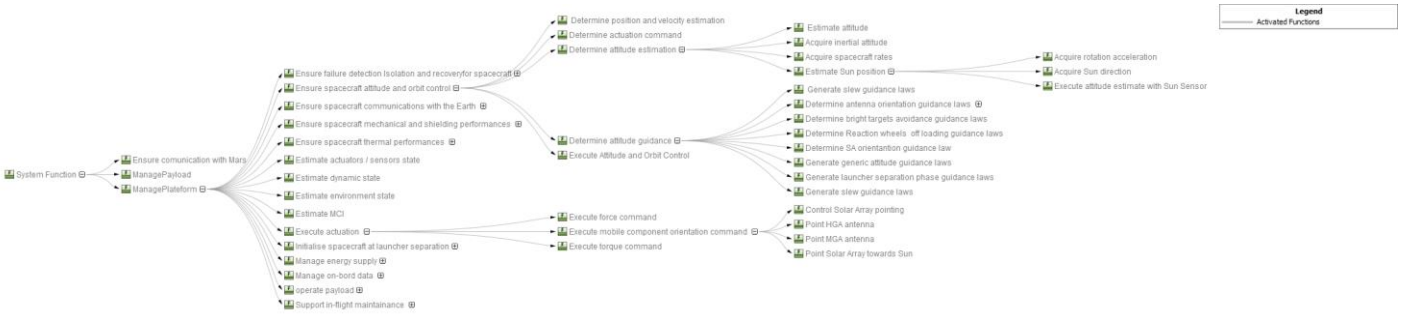

Figure 3: Functional decomposition

Note that the proposed functional decomposition comes from an arbitrary choice (Figure 3). The functional decomposition could be different as the concrete and relevant engineering information comes from the leaf functions (i.e. the elementary functions identified as leaf in the decomposition tree). These leaf functions are then allocated to logical components (Figure 4).

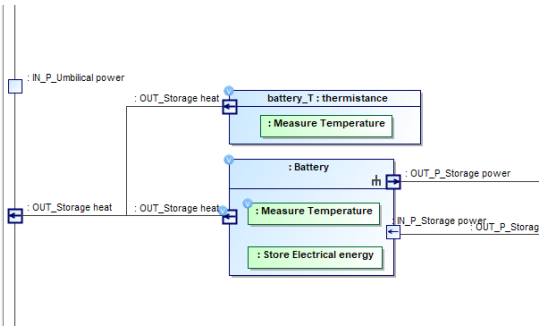

Figure 4: Functional to logical definition

In our model, the main function “manage energy supply” has a leaf function “store electrical energy” and the main function “ensure spacecraft thermal performance” has a leaf function “measure Temperature”. The first leaf function (store electrical energy) is allocated to the logical component “Battery” and the second one (measure Temperature) to the component “thermistor”. This last function is also allocated to “Battery” to cover two possibilities: battery with internal or external monitoring. (Figure 4).

The model containing every possible solution is called 150% model. It defines functions and logical elements in an “over definition” of the system which cannot have a real existence. It is the merging of all functional and logical architectures possibilities in a single model.

To use the same vocabulary as PLE (Product Line Engineering), 150% models for the functional and logical views of a system are proposed as well as a variability model based on features as reference. When necessary, a configuration with the variability model is defined to extract a 100% model for the functional and logical definitions of the system solution for the targeted project.

Variability Management

Associated to the 150% generic model, a feature model is defined to identify the expected variation points.

The two main concepts are core assets defining common elements of all products available in a product line (such as solar array), and variant assets defining elements having different instances for specific products of the product line (a battery with internal temperature monitoring or not). Different variant asset types define the main variabilities (Figure 5):

- ✓ Optional feature: can be selected or not.
- ✓ Mandatory feature: shall have associated functions but one in front of N must be chosen. Is applicable to a structure.
- ✓ Alternative feature: only one is selected in front of a choice of N.
- ✓ Or Feature: at least one is selected in front of a choice of N
- ✓ Parameter: an asset can change due to property value.

It is necessary to deal with offering a catalogue of possible options and possible choices on mandatory assets. It is done through the usage of a feature model providing all possible choices. With this, the capability to make a selection is offered to define specific configuration answering dedicated needs. Each configuration is called a “variant”.

In variability management, **constraint definition between features** can be added to limit the possible choices in the configuration. Two constraints are expressed:

- ✓ Requires : It is possible to select this feature only if the other one is selected
- ✓ Conflicts: If one feature is selected, the other one cannot be selected.

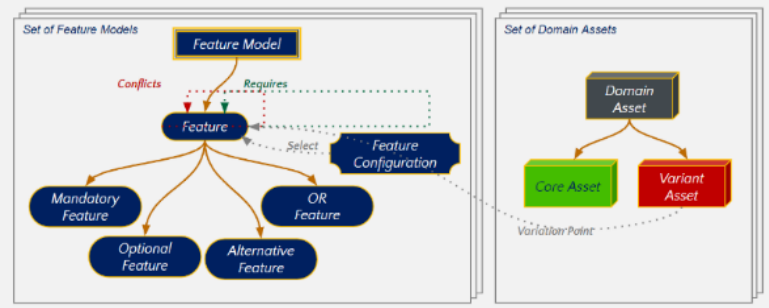

Figure 5: Variability management summary

MOFLT© and PLE concepts usage for model Reuse with CAMEO

We then have on one hand a 150 % model based on R-MOFLT and on the other hand a description of the variability defined with a feature model. The next step is to make the link between the two models and it could be done by defining variation points in the 150% model based on the features definition. It is a classical approach to manage product line definition. Based on the generic and feature models, it is then possible to generate a variant answering to the project's needs. The expected features are selected to support the mission needs. The association of the generic model with the feature one (also called feature tree), is performed using the standard MBPLE's plugin of Cameo System Modeler® (CSM). Extraction of the variant is performed in five main steps described hereafter, illustrated with some elements of the generic power system definition.

First, a 150% model of the system is defined. In the Figure 6, a generic power system model is defined. This power system can have a specific solar array regulation, an internal-temperature-monitored battery or not, a regulated or unregulated bus and a distribution where the type and quantity of protection lines is selected.

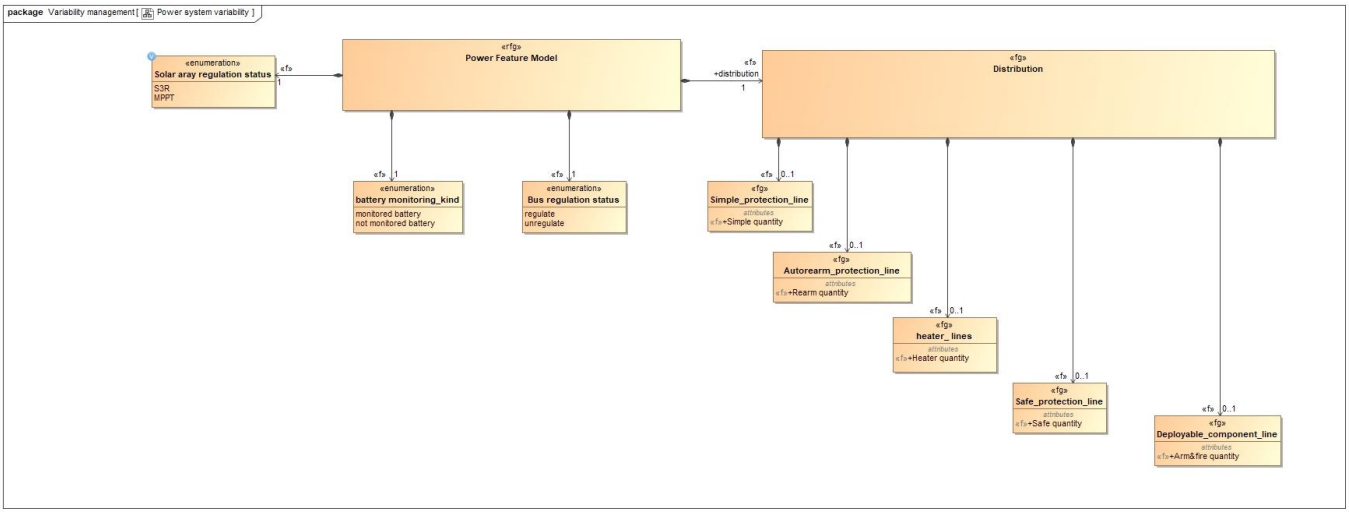

Figure 6: Power system feature model

The second step is to associate variation points to the model (using MBPLE plugin), particularly on variant assets. Three kinds of variation point can be added: the "*Existence*" to define that an asset existence depends on a feature (e.g. Figure 8: re-arm protection line usage or not: R-LCL), the "*Element property*" when a particular instance of element type is selected (e.g. Figure 7: "*S3R*" and "*MPPT*" are defined as instances of "solar array regulator" using SysML inheritance properties) and the "*Primitive property*". The "*Primitive property*" could be used to associate a value to an element property. In the proposed example, this variation point defines the variability of the number of protected lines needed by changing the multiplicity of the associated logical component (Figure 8).

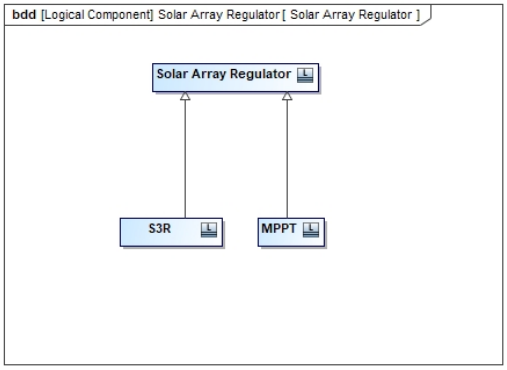

Figure 7: SysML inheritance to manage variability

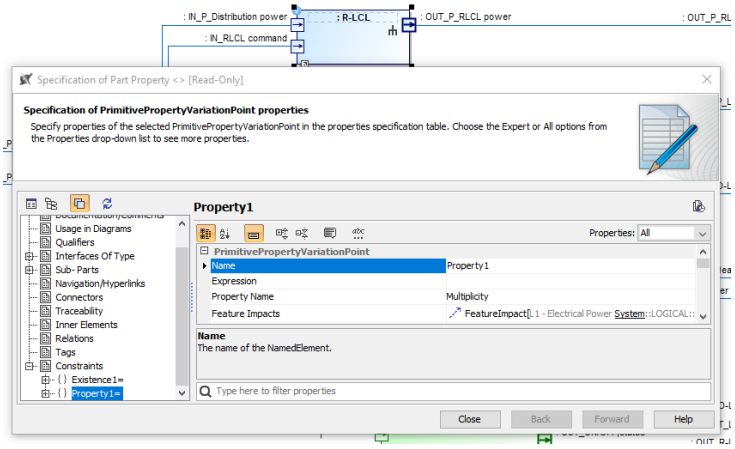

Figure 8: Existence and multiplicity for R-LCL

The third step is to define a specific variant. To do so, a configuration table is set to define choices according to the defined feature model. This configuration consists in selecting the existence and the value of the corresponding variation points parameters. It is possible to set several tables using sub-tables to define configurations of a feature model. In our example (Figure 9) the global configuration is composed of a sub configuration associated to the distribution. The global configuration "solar array regulation only" is composed of an unregulated bus, MPPT SA regulation, a not monitored battery and 3HG distribution (using LCL, R-LCL and heater line).

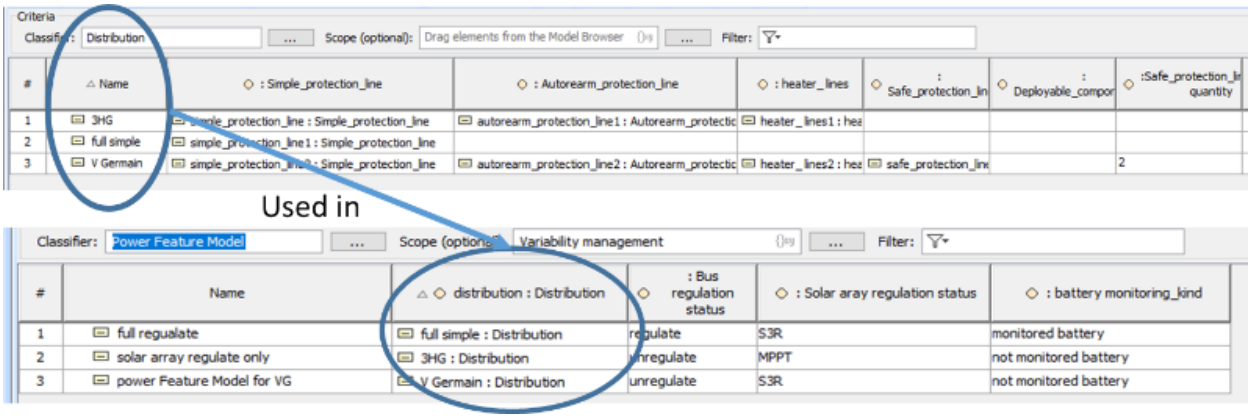

Figure 9: Variant configuration table

The fourth step is to control the model to be extracted for the project by applying the defined configuration to the 150% model. Removed, conserved and instantiated model objects are colored for result evaluation.

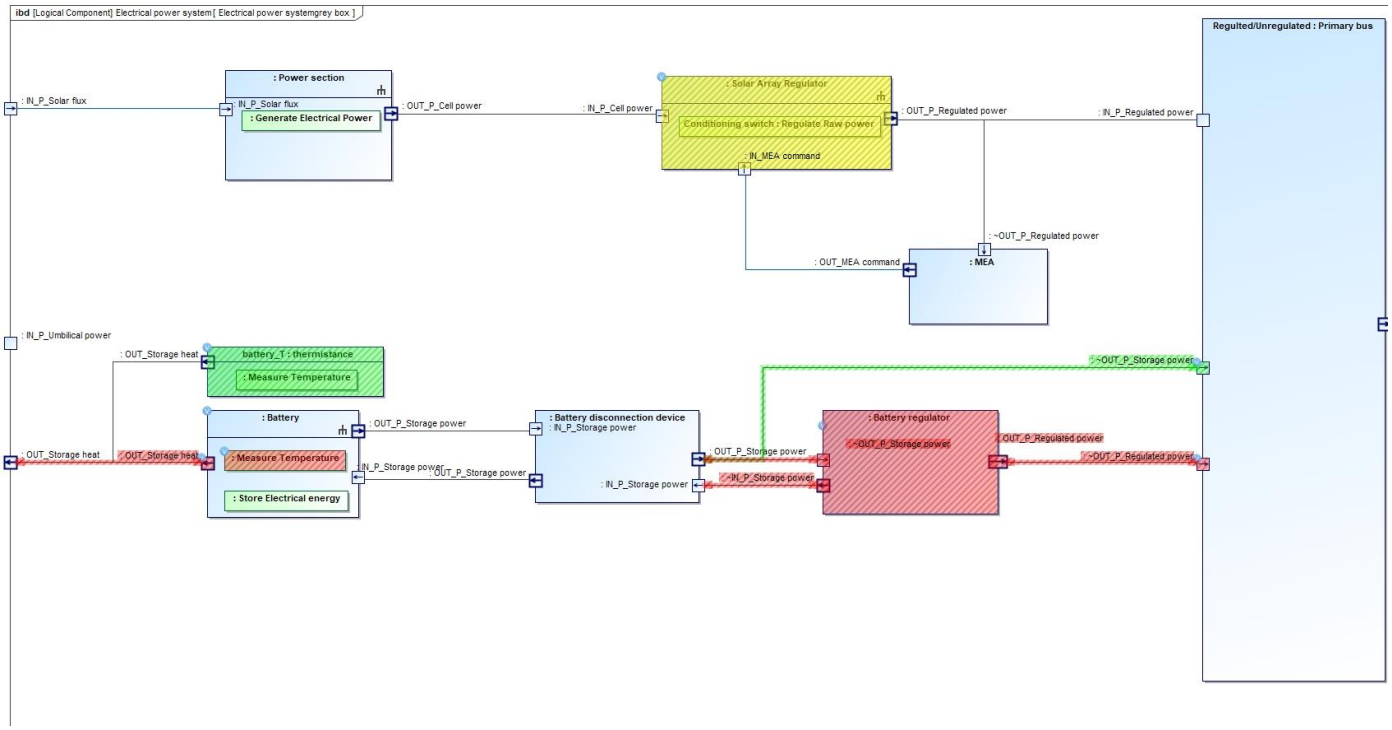

Figure 10: Variant preparation

For example (Figure 10), a power system variant defined by the configuration, “solar array regulation only” (Figure 9), “Measure temperature” function will be removed of the component “Battery”. The logical component ‘Battery regulator” will also be removed (red elements are deleted). A thermistor is kept to support temperature monitoring (green elements are kept). Multiplicity of “LCL” and Solar array regulator type will be valuated (yellow model objects will be instantiated).

The fifth step is to generate the variant by a simple request, to obtain the final model (Figure 11). The variant model can then be stored as a branch of the system definition, (the configuration set could also be stored to support future changes). It could be reused in a project or several projects with the same needs, depending on the chosen process. The choice made shall take into account the global process of configuration management (not discussed in the paper). We can see on our example Figure 11 that Solar Array regulator, yellow on the previous step, is now instantiated to “MPPT”

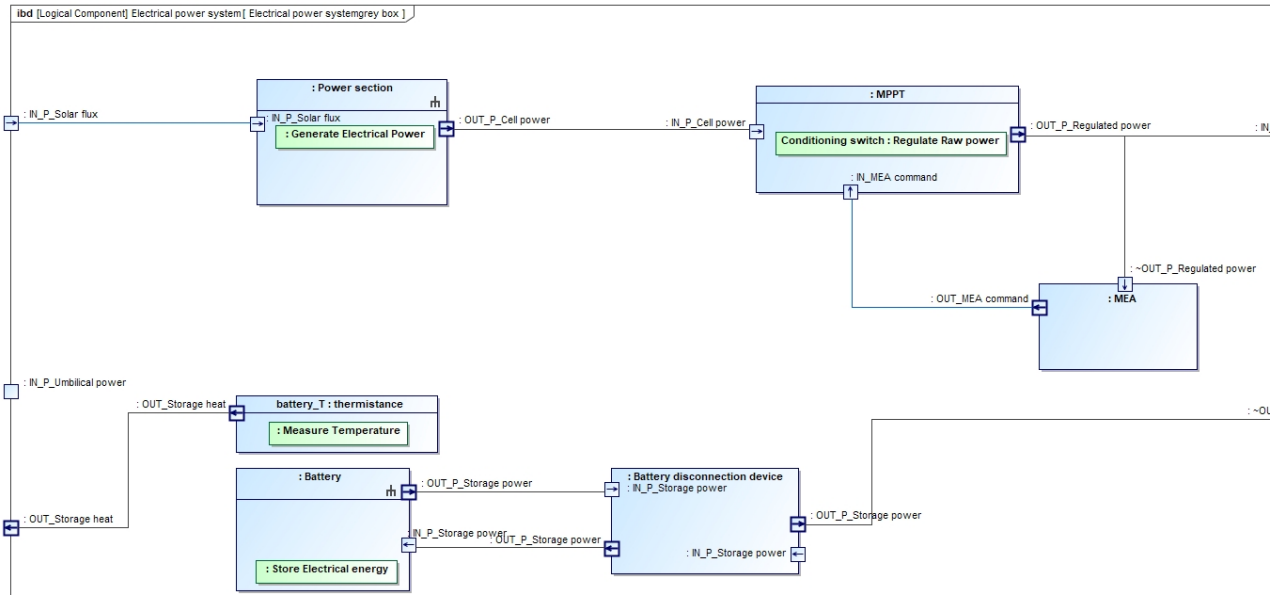

Figure 11: Generated Model of the variant

Conclusion

The experiment led in Airbus shows the capability to perform reuse of models using variability management concepts. A model is a part of system design data. We have applied this approach in a first step to cover a need expressed by power architects. The extension was done to AOCS, thermal regulation system and it is in progress for propulsion system, launcher separation management and communication system. To improve this way of working we have made a first application on Ariel project (side exercise) to validate the feasibility and the interest for the project team. The result is very positive and shows that it is easy to rapidly obtain the parts to define a design. Based on this experience we expect to switch from a purely opportunistic reuse from a project to another to an approach which proposes reusable functions based on model usages. In the end, the generic platform definition offers the capability to initiate a project with the global platform model.

The activity which enables the reuse is: *the study of what can be considered as generic with possible variation points on a design*. It explains why the generic platform is used as a starting point and concentrate the knowledge on all main possible solutions. For the technical aspect to consider to extract a model to define a variant, we have demonstrated that existing technologies cover our need.

Some limitations appeared during the study and it seems that the usage of a specialized tool to support feature model definition could solve those points. It is why the usage of Pure::Variant coupled with CAMEO (the plugin exists) is envisaged for future studies.

Some studies could be launched to enlarge the perimeter and try to identify generic definitions of operations for a dedicated mission perimeter (e.g. Earth observation).The results shall be to provide the capability to reuse the operational layer, based on the “O” of “R-MOFLT©”. Another study shall be done to offer the capability to manage a catalogue of technical products (T of “MOFLT©”) with variability thus increasing the level of reuse.