PROPERTY MODEL METHODOLOGY: A CASE STUDY WITH MODELICA

Romain Pinquié[(1)], **Patrice Micouin**[(1)], **Philippe Véron**[(1)], **Frédéric Segonds**[(2)]
[(1)] Laboratoire des Sciences de l'Information et des Systèmes UMR CNRS 7296 – Aix-en-Provence, France
[(2)] Laboratoire Conception de Produits et Innovation – Paris, France
Arts et Métiers ParisTech
{romain.pinquie, philippe.veron, frederic.segonds}@ensam.eu
patrice.micouin@incose.org

ABSTRACT

The aim of this paper is twofold. Firstly, it is intended to demonstrate the relevance of the Property Model Methodology (PMM) to specify, validate, design and verify continuous multi-physics systems. Secondly, it aims at verifying the compatibility of PMM concepts with the Modelica simulation language. We will be using the case study of an aircraft landing gear to show how to translate the theoretical concepts of PMM into executable Modelica models. This article proves the fundamental concepts of PMM and provides a starting point for further research so as to not only model other types of engineered systems such as discrete and hybrid systems, but also support additional systems engineering activities, such as safety-reliability.

KEYWORDS

Model-Based Systems Engineering, Validation, Verification, Modelica, Property-Based Requirement, Property Model Methodology.

1. INTRODUCTION

It is broadly accepted that there is a crisis in the classic systems engineering [1], as there was one in software engineering starting from the nineties. Whether we consider the energy or the automotive and transportation or the space industry, the symptoms of the crisis are the same: delivery delays, cost overruns and a lack of maturity during the system's infancy. A cumbersome document centric approach, the technical challenges assigned to the systems, and the large, multicultural geographically dispersed teams through whom systems are developed are among the causes of this crisis.

Although there is a shared understanding of the classic systems engineering crisis, the solutions proposed for resolving it diverge. First, there are the pragmatists who will put forward minimum corrective actions to obtain the presumed greatest improvements. The definition of best practice guides is a pragmatist's alternative solution. We call the second category inter-subjectivists who focus on more *agile* methods to reduce the misunderstandings arising within the development teams - "*people rather than processes*". Finally, there are those who see rigorous formal solutions. Although each approach contains a grain of truth and deserves to be explored, we undeniably side with the formal one. Indeed, we claim that a formal system development process including validated specification models and verified design models can solve the classic systems engineering crisis, the principles of which were established decades ago [2].

In this paper, (1) we shall present the Property Model Methodology (PMM), then (2) we will offer some insight into its utilisation through the case of an aircraft landing gear. Lastly, (3) we intend to show how to translate PMM models into Modelica models so as to support the validation of requirements and the verification of designs by using simulation.

2. PROPERTY MODEL METHODOLOGY

PMM is a new Model-Based Systems Engineering (MBSE) method [3]. Two main characteristics are at the origin of its compound name: (1) the formulation of requirements based on the concept of property – Property-Based Requirements (PBRs) –, and (2) the adoption of a MBSE approach. PMM is a top-down approach that authorises the reuse of pre-existing blocks at any hierarchical level. Additionally, PMM complies with current industrial development standards, specifically ARP4754A [4] and EIA632 [5]. Finally, the third pillar of PMM is simulation, which is the main technique to validate specification models and verify design models.
In this paper we are distinguishing three levels in a system hierarchy: the system level that is the highest level, the building blocks level(s) corresponding to the intermediate hierarchical levels resulting from the recursive breakdown process, and the elementary building blocks level that is the atomic level.

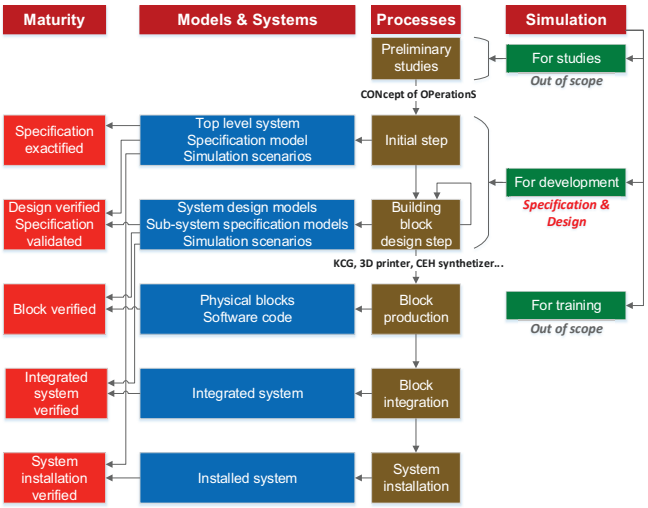

Figure 1 PMM Process

As Fig. 1 roughly describes, for an innovative system, PMM usually starts after the validation of a Concept of Operations [6] and is made up of 4 activities[1]:

1. The first activity, which is carried out at the system level, is the development of the system specification model associated with simulation scenarios defined to validate the system specification model.
2. Then, we continue with a recursive process that consists in breaking down the system into a finite set of building blocks. The set of building blocks and their connections constitute the system architecture. Each building block is made up of a specification model and one or more design model alternatives. Thus, a collection of competing system architectures is considered, but only the preferred one is selected. At each hierarchical level, the building block specification models are validated together against higher level building block specification models. A building block is qualified as elementary when it does not require any further decomposition. A library stores the building blocks that might be reused in future developments. From the point of view of its acquirer, any building block that is picked from a library is also an elementary building block. The design process ends when all the elementary building blocks have been identified, and implemented or acquired.
3. The third activity is a bottom-up process that consists in recursively integrating the elementary building blocks and the building blocks so as to finally end up with the system design model. For each integration level, design models are integrated and verified against their own specification models.
4. The final steps that deal with the production of the physical building blocks, their integration, and the final installation of the system in its environment are outside the scope of this study, but we encourage readers to read [3, chap. 11] for further details.

2.1. PMM: Specification

Engineered systems are goal oriented; therefore, they must embody the intents of their designers to be correctly developed. This is the reason why the first activity in system development is the development of the system specification. PMM proposes a Goal Oriented Requirement Engineering (GORE) approach such as KAOS [7]. According to PMM rules, the system specification process starts with the identification and definition of goals, that is, the intended effects. Goals are modelled as outputs of the system specification model.

The next step is to elicit the occurrence conditions of the identified goals. The modelling of these conditions of actualisation enables analysts to identify and model the expected inputs and additional outputs including observable states, secondary[2] (undesired) outputs, secondary (undesired) inputs and system failures. Fig. 2 illustrates the inputs and outputs of a specification model in PMM.

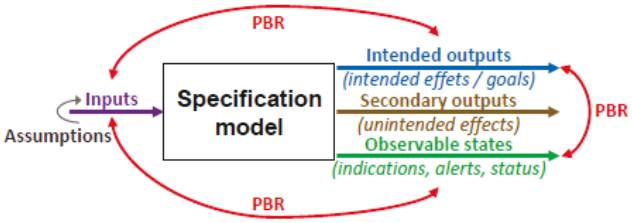

Figure 2 PMM specification model

Then, the results of the analysis are formalized as one or several PBRs [9]. PBRs are predicates that link goals, secondary outputs, observable states and in-

[1] It also includes recovery points to reengineer the system when a goal is not met.

[2] According to a terminology due to Hubka [8].

puts to specify the actualisation conditions of system's properties.

The basic form of a PBR is as follows:

[when C →] val(O.P) ∈ D

This formal statement means: "*When* ***condition*** $\mathcal{C}$ *is true,* ***property*** $\mathcal{P}$ *of* ***object*** $\mathcal{O}$ *is actual and its value* ***shall*** *belong to* ***domain*** $\mathcal{D}$*".*

In a PBR, $\mathcal{C}$ is a relevant condition of the system or its environment: a functioning mode, a system state, an (undesirable) event, a delay or a combination of such features. On the other hand, domain $\mathcal{D}$ is a finite or infinite set such as {0,1} or R^n (possibly linked to a frame and a physical unit).

Because they are outside the system's developer control and only presumed, *assumptions* are specific PBRs limited to input properties.

As shown in [9], several PBRs can be combined thanks to the conjunction operator "∧" and result in a composite PBR. The partial order relationships "≤" and "≥" enable analysts to compare two PBRs. For instance, the expression "PBR1 ∧ PBR2" is the conjunction of PBR1 and PBR2 and is itself a PBR. Moreover, the statement "PBR1 ≤ PBR2" means that PBR1 is less constraining than PBR2.

The concept of a PBR can be directly implemented as a conditional assertion - i.e. a Boolean function - that is available in most simulation languages such as VHDL-AMS [10, 11], Modelica [12] or Simulink [13] usually used by subject-matter experts – e.g. mechanics, electrics, electronics, hydraulics, control, etc. The need for a dedicated grammar and requirement modelling language to express PBRs is therefore inessential so far. Indeed, PMM adheres to the principles of simplicity and parsimony avoiding the introduction of supplementary complex features for end users when these features could lead to limitations compared to the expressiveness of the simulation languages targeted by PMM. Conversely, Form-L [14] or Lucid/Lutin-Stimulus [15] appear to be sub-optimal. The former requires Modelica extensions, whereas the latter is a standalone requirement language isolated from both the standard simulation languages and the subsequent design modelling and derivation activities described in §2.2.

A specification model is a formal model that includes: (1) system requirements, (2) system interface requirements, and (3) system assumptions. By using a simulation language like Modelica, VHDL-AMS [16] or Simulink one can express specification models as simulation models so as to make sure that specification models and interfaces are coherent. Furthermore, the simulation of a specification model linked with the corresponding equation design model [3, p 139] provides analysts with various advantages. On the one hand, it helps to guarantee the completeness and correctness of a particular system specification model for a given set of validation scenarios. On the other hand, simulation provides capabilities for validating building block specification models against higher level building block specification models up to the system specification model.

2.2. PMM: Design & Derivation of PBRs

Once the system specification model is complete and correct, the second system development activity consists in developing a system design model. For a very simple system, the system design model is usually an equation design model. However, for a more complicated system, the system design model is a structural design model [3, p 145]. A structural design model is a formal definition of a system architecture $\mathcal{A}$ that is made up of three elements: (1) building blocks; (2) an endo-structure linking together the building blocks that belong to the system; and (3) an exo-structure linking together system building blocks with objects that belong to the environment through its interfaces.

For each candidate structural design model, the third system development activity consists in deriving the system requirements {PBRs} into building-block requirements {PBR_1, …, PBR_n}. To be valid, for a given system structural design model $\mathcal{A}$ and for a set of assumptions on the properties of objects that belong to the environment $\mathcal{EA}$, the conjunction of the derived building-blocks PBR_1, …, PBR_n must be more constraining than the system PBRs.

Derivation:

When $\mathcal{A} \wedge \mathcal{EA} \rightarrow PBR \leq PBR_1 \wedge ... \wedge PBR_n$

This validity condition of PBR derivation leads to the "prime contractor" theorem:

"*Prime contractor*" theorem: A sufficient condition for a system to comply with its PBRs is that its building blocks comply with the PBRs validly derived from the system PBRs, provided the design choices and assumptions made about the environment driving the derivation remain valid.

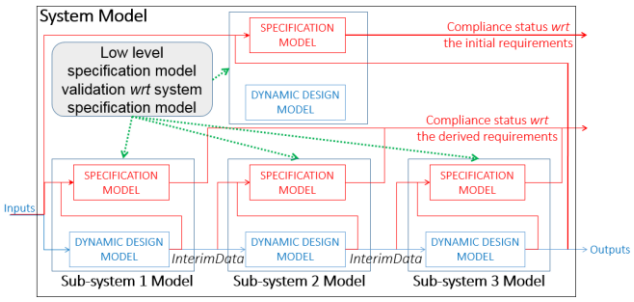

Figure 3 Validation process of subsystem specification models

PMM uses simulation as the main method to validate the derivation of system PBRs into a set of derived building block PBRs with respect to a sufficient set of validation scenarios.

Simulation also provides capabilities for verifying design models. While the simulation is running, for all submitted simulation scenarios, Fig. 3 and Fig. 4 show that the specification models monitor the interacting design models so as to check whether any requirement is violated. If no violation has been detected during the verification of a building block design model, then the building block design model is verified.

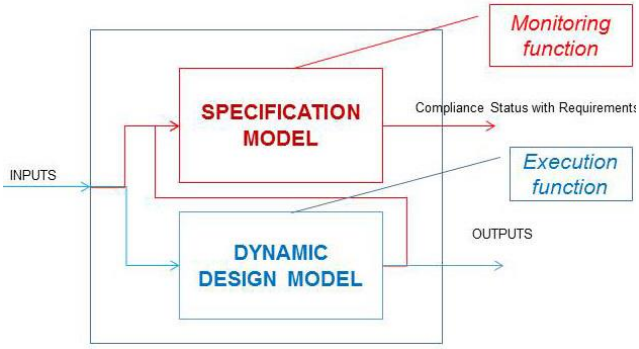

Figure 4 Specification and design model roles during a verification process

The design process – i.e. design, derivation, derivation validation – is applicable at any level within the system architecture, down to the lowest hierarchical level corresponding to the elementary building blocks. Lastly, the design verification is performed according to § 2 (3).

3. CASE STUDY

Here we will be showing how to model a landing gear system (LGS) with PMM. The modelling of a system-of-interest and its building blocks includes two main phases: (1) the specification phase whose output is a single specification model; and (2) the design phase whose output is one or several design models.

Borrowed from Sadraey [17], the case study focuses on the design of a LGS for a two turboprop subsonic civil transport aircraft that can carry up to 18 passengers. The reference gives all the required characteristics for the design of the LGS and can be supplemented with aeronautical [18] and mechanical [19] knowledge. The LGS design activities start after an initial design of the aircraft [17, p. 481] because the main aircraft design parameters – e.g. the maximum mass at take-off and landing, the inertia moments, the forward and aft positions of the centre of gravity, the lift and drag coefficients and the take-off speed – have to be known before going through the LGS design.

Let's consider the following initial PBR:

- **PBR. "Boarding attitude"** – For small commercial aircraft, when passengers are boarding, the pitch angle of the aircraft shall belong to the intervall [-1°, 1°].

This PBR has a direct impact on the LGS design. Indeed, a tricycle configuration with a nose landing gear (NLG) and two main landing gears (MLG) is the cheapest solution satisfying this requirement.

According to PMM, the modelling process in general, and in particular the LGS one, is a "zigzagging[3]" process. For a given hierarchical level, we have to "zig" from the specification model to the design model. Once the design model is defined, we have to "zag" from the design model of the current hierarchical level to the specification model of the next lower hierarchical level. The "zigzagging" process ends when the elementary building block design models are fully defined. The output of the process is a coherent tree, the LGS system model, of specification models associated with design models.

3.1. Specification

In this section we are focusing on the specification phase by introducing the specification process that consists in specifying goals in terms of formal PBR(s).

As Currey [19, p.13] reminds us, the LGS is "*the essential intermediary between the aeroplane and*

[3] This term is borrowed from Nam Pyo Suh ([20], p. 29).

catastrophe". Such a very synthetic statement leads to a set of goals [17, p.481] that the LGS must fulfil. Among the set of goals there is:

- **Goal. "Ground clearance"** – To provide a clearance between the aircraft structure and the ground for protecting the aircraft structure when on ground.

This "Ground clearance" goal is specified into a collection of PBRs that are stored in a specification model with potentially some assumptions. One example of a PBR is given below:

- **PBR. "Ground clearance"** – When the aircraft is on ground and its ground speed is lower than V_r, the clearance between the lowest point of the aircraft and the ground shall be equal or greater than Hprop_tip_min = 18 cm (Far Part 23 CS25.925).

There are many other PBRs expressed in other aircraft conditions contributing to reach the "ground clearance" goal such as the:

- **PBR. "Take-off rotation clearance" –** When the aircraft is taking-off and after the nose up rotation, the fuselage rear upsweep point shall be at a ground height $H_{fus\_upsw\_pt}$ greater than a desired value $H_{clearance}$ to prevent the rear fuselage from-striking the ground at take-off.

Another goal of the LGS is:

- **Goal. "Touch-down energy absorption"** – To absorb and to dissipate the kinetic energy of the aircraft when the aircraft is landing.

This "touch-down energy absorption" goal is specified in a collection of PBRs that are recorded in the LGS specification model with potentially some assumptions. One example of a PBR is given below

- **PBR. "Touch-down requirement"** – When touching down and for a vertical speed V_z that is equal or less than V_{z_max}, the LGS shall deflect to absorb and to dissipate the aircraft kinetic energy without an excessive reaction load to the aircraft or without bottoming out.
- **ASSUMP$_1$. "Maximum vertical speed"** – When wheels touch down, the aircraft vertical speed is assumed to be less than V_{z_max} = 3 m.s^{-1} at design landing weight and 2 m.s^{-1} at maximum gross weight
- **ASSUMP$_2$. "Max reaction factor"** – when landing, the reaction factor N shall not exceed 1.5.

After landing, there may be undesired reactions in the LGS such as an excessive rebound and vertical oscillations. To prevent such secondary outcomes, an additional PBR has to be specified.

- **PBR. "Rebound requirement"** – When touching-down with a vertical speed V_z that is equal or less than 3 m.s^{-1}, the LGS shall not bounce 10% above its statically deflected position*"*.

Finally, we are considering the following goal:

- **Goal. "Aircraft deceleration"** – To stop the aircraft when touching-down at landing.
- **PBR. "Braking requirement"** – When touching down at any acceptable longitudinal ground speed V_x, the LGS shall be able to absorb and to dissipate the aircraft kinetic energy.
- **ASSUMP$_1$. "Longitudinal speed"** – When wheels touch down, the aircraft longitudinal speed is assumed to be less than 3 m.s^{-1}.

3.2. Design

Hereunder we will be focusing on the design process consisting in breaking down the system into building blocks.

According to PMM rules, the preferred design model, which has been selected among several design model alternatives, implements the aircraft specification model. PMM defines four types of design models, each one addressing a different purpose. Dynamic models include Equation Design Models (EDM) and Behavioural Design Models (BDM). Static models include Structural Design Models (SDM) and Reliability Design Models (RDM).

In our case study, the aircraft design model is an SDM that is decomposed into two building blocks: the body and the Landing Gear System (LGS). Each building block has its own specification model (prescriptive part) and design model (descriptive part).

Similarly, the LGS structural design model is broken down into three Landing Gear (LG) building blocks: the Nose Landing Gear (NLG), the left Main Landing Gear (MLG), and the right MLG. Each LG building block design model is also a structural design model that consists of several building blocks including an articulated leg, a shock absorber, one or two

wheels, an extension/retraction mechanism and a braking system for both MLGs.

The design modelling activities of PMM can rely on the mathematical modelling and analysis of systems. For instance, PMM can benefit from system dynamics techniques [21], e.g. free body diagrams and equation-based representations of physical systems. Such mathematical representations do not usually strictly mimic the empirical reality. Additionally, they can lead to the merging or separation of properties owned by different objects, as well as to the creation of fictitious objects. For instance, Fig. 5 shows a typical mechanical representation of a landing gear that is defined so as to derive its equations of motion. Ideal, massless springs (K_1, K_2) and damper represent wheels and shock absorbers with their masses concentrated on the M_{LG}.

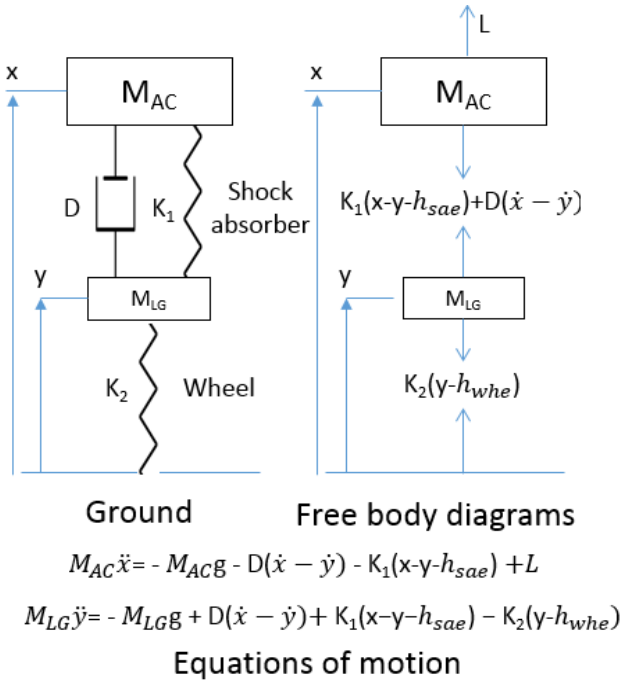

Figure 5 Mechanical model of a landing gear

A PMM candidate design model includes a shock absorber modelled as the parallel association of a spring (K_1) and a damper (D), a wheel modelled as a spring (K_2), and a virtual component [3, page 204] named LG_Assembly.

According to the definition of a virtual component, the building block "LG_Assembly" does not correspond to any real, tangible object. On the one hand, it represents the emergent property of the MLG system emerging from the behavioural properties of its components. On the other hand, it is the approximately true model of the no less real designatum of the Newton's second law applied to the MLG:

$$M_{LG}\ddot{y} = -M_{LG}g + D(\dot{x} - \dot{y}) + K_1(x - y - h_{sae}) - K_2(y - h_{whe}) \quad (1)$$

These notions not only show how PMM fills the gap between the operative rules of systems engineering and the substantive rules of engineering disciplines. They also illustrate how PMM exploits scientific bodies of knowledge.

3.3. Requirement derivation

According to PMM rules, the aircraft requirements have to be derived into body requirements and LGS requirements. For instance, the "Take-off rotation clearance" requirement derives into two requirements. The first one is allocated to the body and constrains the MLG attachment location on the body. The second one is allocated to the MLG and constrains the MLG height. As previously shown, assuming the derivation is valid, the satisfaction of both derived building block requirements implies the satisfaction of the initial "Take-off rotation clearance" system requirement.

The derivation process of a PBR continues as long as the satisfaction of this PBR requires one or several additional building blocks. Nevertheless, it is not an endless process. It ends when the requirement matches with a design pattern or an architectural characteristic – e.g. redundant blocks, block arrangement, etc. – that fully satisfies it. For instance, the derivation of the "Boarding attitude" PBR ends at the LGS level with the selection of the tricycle configuration that consists of a nose landing gear (NLG) and two symmetrical main landing gears (MLG). For a small commercial aircraft, this LGS design pattern fulfils the "Boarding attitude" PBR.

The structural design parameters of the LGS – i.e. the height, the attachment location on the fuselage and/or the wing, the wheel base corresponding to the distance between the MLGs and the NLG, and the wheel track corresponding to the distance between the left and right MLG – are defined according to business rules [17, chap 9] and shall satisfy all the related requirements, including in the worst conditions, that is, when the current centre of gravity (CoG) matches with the forward or the aft CoG.

As an example, let us consider the deflection of a MLG when the aircraft lands. This deflection shall be sufficient to reduce the landing shock energy at an

acceptable level for the aircraft structure and for its passengers.

According to [18, p. 275], the total energy (E) of the aircraft at touchdown is expressed by the formula:

$$E = \frac{W * v^2}{2g} + (W - L) * S \quad (2)$$

W is the aircraft weight W, v is the vertical speed, L is the lift at touchdown, g is the gravitational acceleration and S is the MLG deflection.

Moreover, the aircraft total energy upper bound E_{max} is given by:

$$E_{max} = \frac{W_{max} * v_{max}^2}{2g} + W_{max} * S_{max} \quad (3)$$

W_{max} is the aircraft maximum landing weight, v_{max} is the maximum landing vertical speed at touchdown and S_{max} is the MLG deflection.

The kinetic energy capacity of the MLG shall be equal to E_{max} so as to be absorbed at touchdown.

$$E_{max} = \eta_{MLG} * S_{max} * N * W_{max} \quad (4)$$

E_{max} is the aircraft total energy, W_{max} is the aircraft weight at touchdown, N is the reaction factor, η_{MLG} is the MLG efficiency factor, and S_{max} is the MLG maximum deflection.

We deduce from the equation (4) that if the MLG efficiency factor η_{MLG} is assumed to be greater than a threshold such as $\eta_{MLG} \geq 0.75$, then the minimum MLG deflection S required to absorb the maximum aircraft energy at touchdown is prescribed by the following the $PBR_{MLG\_Deflection}$ as follows:

- $PBR_{MLG\_Deflection}$: When $(W \leq W_{max}) \wedge (v \leq v_{max}) \wedge (\eta_{MLG} \geq 0.75) \rightarrow MLG.S \geq S_{max}$, where S_{max} is the solution of equation Eq 3.

The shock absorber (SA) and the main wheel tyre (T) building blocks support the MLG deflection. Consequently, the derivation of the $PBR_{MLG\_Deflection}$ results into two PBRs. The PBR_{SA_Stroke} relates to the stroke property of the shock absorber, whereas $PBR_{T\_Deflection}$ constrains the deflection property of the tyre.

Assuming that the shock absorber efficiency factor (η_{SA}) and the tyre efficiency factor (η_T) are greater than 0.8 and 0.47 respectively, then the $PBR_{MLG\_Deflection}$ may be derived as follows:

- PBR_{SA_stroke}: When $(\eta_{MLG} \geq 0.75) \wedge (\eta_{SA} \geq 0.8) \wedge (\eta_T \geq 0.47) \rightarrow SA.Stroke \geq 0.85 * S_{max}$
- $PBR_{T\_deflection}$: When $(\eta_{MLG} \geq 0.75) \wedge (\eta_{SA} \geq 0.8) \wedge (\eta_T \geq 0.47) \rightarrow T.Deflection \geq 0.15 * S_{max.}$

The conjunction of the derived PBRs is more constraining than their parent PBR:

- $PBR_{SA\_Stroke} \wedge PBR_{T\_Deflection} \geq PBR_{MLG\_Deflection}$

The previous PBR derivation dealt with the same deflection[4] property.

Similarly, the braking capability of the MLGs deals with different properties of MLG parts. The aircraft landing phase involves the braking capability (braking torque) of brakes, the capacity of brakes' heat sinks (mass and specific heat), the wheel radius and the tyre friction coefficient. These properties are constrained by PBRs derived from the MLG braking performance PBR and by making assumptions on both the runway and the MLG architecture.

3.4. Implementation with Modelica

The Modelica Association website[5] states that "*Modelica is a non-proprietary, object-oriented, equation-based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents*". From the point of view of its expressiveness, Modelica is similar to VHDL-AMS.

Numerous proprietary and open source integrated development environments that support Modelica are available: CATIA Systems, Dymola, LMS AMESim, MapleSim, OpenModelica, SCICOS, Wolfram SystemModeler etc. This is an advantage compared to the environments supporting VHDL-AMS.

However, Modelica does not follow the systems engineering philosophy. Indeed, systems engineering relies upon a Cartesian top-down approach. Conversely, the modelling process of Modelica is a bottom-up approach that starts with the design or reuse of elementary building blocks that are sequentially integrated so as to reach the system level.

Furthermore, systems engineering, which adheres to the principles of scientific methods [22, p 10-11], prescribes a development process that (1) starts with the specification of the requirements to be met – the problem's statement –, before (2) developing one or

[4] For the shock absorber (SA), the terms *stroke* and *deflection* are synonyms.

[5] https://www.modelica.org/

several design models – the hypothetical solutions –, and to finally (3) verify the conformity of the design models and their physical implementation against the requirements – the evaluation of hypotheses and evidence. Among these development stages, the Modelica community usually only focuses on the design phase by developing unspecified model libraries. The lack of explicit specifications is the major weakness of Modelica Libraries [6].

There have been some recent attempts to adapt Modelica to the systems engineering development process, especially for requirement engineering [23, 24, 25]. However, to the best of our knowledge, all these attempts require language extensions to model requirements. Moreover, the top-down development process that is not only promoted by a system engineering approach, but also in harmony with the requirement derivation process has not been addressed yet.

Despite this context, as we will show in the next section, the current version of Modelica [12] possesses all the right properties to support the MBSE PMM approach.

4. FROM PMM TO MODELICA

In this section we show how to translate PMM models into Modelica models. Fig. 6 gives the gist of Modelica models and their interactions. For convenience, a package AC_DP of inherited Aircraft Design Parameters is shared by the models of the System Model Tree thanks to the use of an "import" clause.

The system root is made of an Aircraft Specification Model extended by an Aircraft Structural Design Model. The Aircraft Structural Design Model consists of two building blocks: Body and LGS. According to PMM, each building block forms a pair that associates a specification model and a descriptive design model. The design model extends the specification model. For instance, in this case study, the inheritance is as follows: the Body Design Model extends the Body Specification Model, whereas the LGS Design Model extends the LGS Specification Model. The System Model Tree can be developed further by decomposing the LGS into two MLGs and a NLG which, in turn, can be decomposed into shock absorbers, wheels, tyres, brakes, LG assembly, etc.

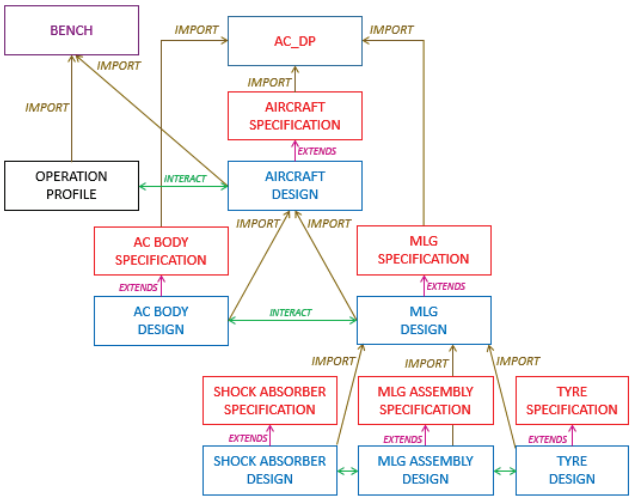

Figure 6 Architecture of PMM models with Modelica

The system under consideration or its building blocks are conceptual models made up of specification models linked with design models. Fig. 7 shows that a PMM specification model is a Modelica partial model, whereas a PMM design model is a Modelica model that extends the specification model.

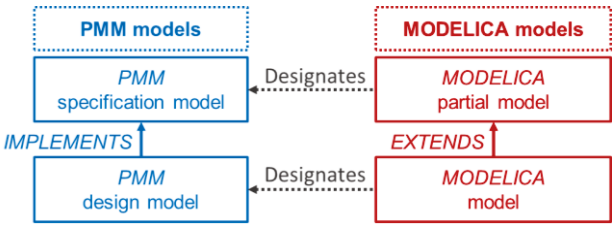

Figure 7 Translation of PMM models into Modelica models

Fig. 8 depicts an alternative to translate PMM models into Modelica models where each Modelica model is a combination of two interacting Modelica submodels. The main advantage of this PMM modelling alternative is its consistency with the building philosophy of existing Modelica libraries.

[6]. As Suh points out in [26], p. 451: "*Many designers deliberately leave their design goals implicit and then start working on design solutions even before they have clearly defined their design goals. They measure their success by comparing their design with the implicit design goals that they had in mind, which may or may not what the customer wanted. They spend a great deal of time improving and iterating the design solution and "what they had in mind" converge* ».

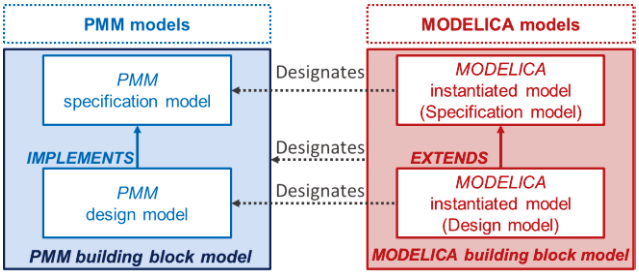

Figure 8 Alternate translation of PMM models into Modelica models

4.1. PBRs in Modelica

The implementation of PMM with Modelica starts with the expression of PBRs as Modelica conditional "assert" equations. For instance, the translation of the "General Clearance" PBR (see 3.1) looks as Fig. 9.

```
if On_Ground and Ground_Speed <= AC_DP.Vr
then
assert (Prop_tip_height > Prop_tip_mini_height,
"General clearance requirement failed", level =
AssertionLevel.warning);
end if;
```

Figure 9 A Modelica textual representation of the PMM "General clearance" PBR

During a simulation run, the "assert" equation calls for an assertion evaluation each time the actualisation condition is true. If the result of the evaluation is false, the simulator signals a requirement violation.

Alternatively, Fig. 10 shows that analysts who are not familiar with programming concepts can specify PBRs by using the Modelica graphic capabilities.

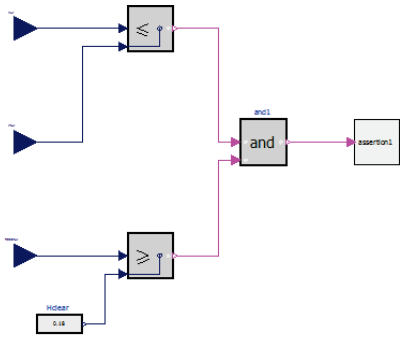

Figure 10 A Modelica graphic representation of the PMM "General clearance" PBR

4.2. PMM specification in Modelica

Fig. 11 shows that Modelica partial models implement PMM specification models. The conditional "assert" equations, which stand for the PBRs, are modelled in the equation section of a Modelica partial model.

```
partial model Aircraft_specification
        ..
    Equation
//"General clearance" assert equation
// Another Modelica assert equation
end Aircraft_specification;
```

Figure 11 A Modelica textual representation of the PMM Aircraft Specification Model

Fig. 12 is a Modelica graphic representation of the Aircraft Specification Model that is equivalent to the Modelica textual representation in Fig. 11.

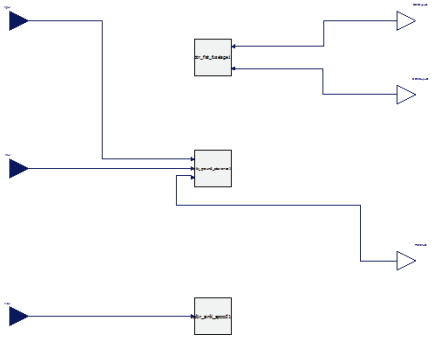

Figure 12 A Modelica graphic representation of the PMM Aircraft Specification Model

4.3. PMM design in Modelica

In PMM, a design model can be an Equation Design Model (EDM), a Behavioural Design Model (BDM), a Structural Design Model (SDM) or a Reliability Design Model (RDM) [4, p. 132].

```
model MLG_Assembly_Design
extends MLG_Assembly _Specification;
    ..
    Position MLG_Position;
    Velocity MLG_Vz = der(MLG_Position);
    Acceleration MLG_Az = der(MLG_Vz);
equation
    Mass * MLG_Az = (-Mass * Ground.g) +
    i.SA_Load - i.Wheel_Load;
    o.MLG_Position = MLG_Position;
    o.MLG_Vz = MLG_Vz;
initial equation
    MLG_Position = ..;
    MLG_Vz = ..;
end MLG_Assembly_Design;
```

Figure 13 A Modelica textual representation of the PMM MLG assembly Equation Design Model

EDMs and SDMs are translatable into Modelica. Fig. 13 and Fig. 14 show that a design model always extends a specification model whatever its type. The difference between both types of model lies in the equation section. Fig. 13 is an EDM that contains equalities "A = B", conditional or iterative equations.

Fig. 14 is an SDM that provides the composition of a building block in its declarative part, whereas its equation part describes the building block structure thanks to its *connect* clauses, e.g. "connect (A, B)".

```
model Main_Landing_Gear_Design
   extends Main_Landing_Gear_Specification;
   import Shock_Absorber_Design;
   import Wheel_Design;
   import MLG_Assembly_Design ;
   Shock_Absorber_Design SA;
   Wheel_Design Wheel;
   MLG_Assembly_Design MLG_Assembly;
equation
   connect(i.AC_Position, SA.i.Body_Position);
   connect(i.AC_Vz, SA.i.Body_Vz);
   connect(SA.o.SA_Load, o.MLG_Load);
   connect(SA.o.Msa, MLG_Assembly.i.Msa);
   connect(Wheel.o.Mwh,
      MLG_Assembly.i.Wheel_Mass);
   ..
   connect(MLG_Assembly.o.MLG_Vz,
      SA.i.MLG_Vz);
   connect(MLG_Assembly.o.MLG_Position,
      Wheel.i.MLG_Position);
   connect(SA.o.SA_Load,
      MLG_Assembly.i.SA_Load);
end Main_Landing_Gear_Design;
```

Figure 14 A Modelica textual representation of the PMM main landing gear Structural Design Model

Fig. 15 shows a graphic representation of the MLG Structural Design Model that is equivalent to the Modelica textual representation in Fig. 14

BDMs and RDMs are outside the scope of this study, but interested readers may refer to [3, p 133 and p 207].

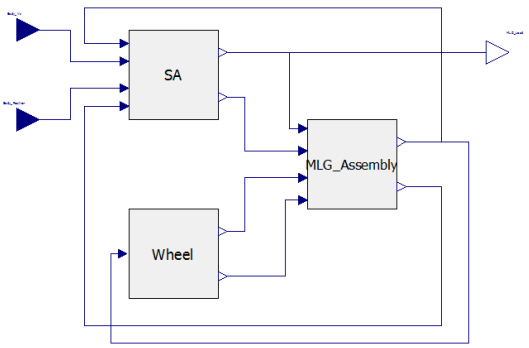

Figure 15 A Modelica graphic representation of the PMM MLG Structural Design Model

4.4. Bench & simulation

In PMM, executable benches aim at ensuring the exactness of the specification models and the truth of the design models.

The first goal of executable benches is to establish the exactness of the specification models for each operational scenario. The exactness of a top level system specification model is not only a technical issue, but also a matter of agreement among stakeholders. Indeed, stakeholders have to first confirm (or infirm) that the simulated behaviours of the representing models are consistent with the expected behaviours of the represented type of systems, and afterwards that the system specification model is an exact specification. Similarly, executable benches enable analysts to validate the derivation of requirements from one architectural level to another and provide evidences that the building block specification models are coherent with the system specification model from which they result.

The second goal of executable benches focuses on the truth of the design models. Liebniz [27] distinguished the formal truth ("*vérité de raisonnement*") of a proposition from its factual truth ("*vérité de fait*"). Formal truth and factual truth are not established in the same way. The formal truth of a proposition relies on its coherence with its premises, whereas the factual truth of a proposition depends on the consistency with the real facts to which it refers.

The design models, which are semiotic systems [3, chap 4] expressed in a simulation language such as Modelica, are formally true when they are coherent with their encompassing environment. If the design models are formally true, then they do not result in any failure (requirement violation or exception) for a sufficient set of operational scenarios in a bench.

The formal truth of the design models is not sufficient because simulation does not protect us from making chimeras. Therefore, design models shall also be factually true, that is, they shall be consistent with the behaviour of the physical represented building blocks and system. To be consistent, the virtual behaviour resulting from the simulation of the design models must be in agreement with the experimental results obtained from the reference systems of simulated design models. Simulation data must be compared with drop test results to confirm the factual truth of a MLG design model, for instance. One should note that the factual truth of a model is just an approximate truth depending on the required accura-

cy. We call *representativeness of models* [3, p. 71] their factual truth.

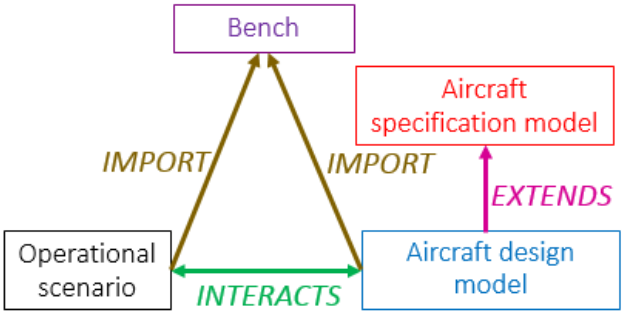

Figure 16 A Modelica graphic representation of the PMM Aircraft Bench

Fig. 16 shows a bench that has to be built to validate specification models or to verify design models. Benches include operational profiles, specification models and design models. Operational profiles cover the relevant operational scenarios – e.g. take-offs, landings –, with various settings – e.g. weight, speed, COG. The landing scenarios must include all conditions: soft and hard landings, level landing and one gear landing. The bench can be used to compare simulation results to real experiments data, such as drop test data, take-off test data, etc. so as to evaluate the representativeness, i.e. its approximate factual truth.

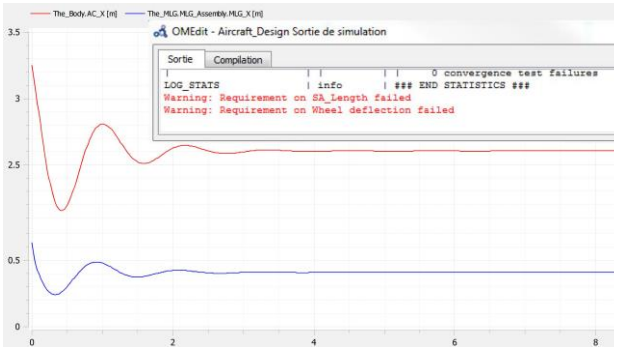

Figure 17 The monitoring performed by a specification model signals the violation of two PBRs

Depending on the context and the intent of the simulation, any violation of a PBR signals an error regarding a PBR derivation or a design model. For instance, Fig. 17 depicts the waveforms of two properties: the CoG position and the wheel center position of the AC object during the operational scenario of landing at $3ms^{-1}$. Both curves on the graph show the evolution of these positions. At the beginning, before touch-down, the MLG is extended. At touch-down, it is fully compressed before reaching its statically deflected position. One may note a slight rebound depicted by the local maximum. In this example, two messages pop up to highlight two validated requirement violations signaling two design errors: "*Requirement on SA_Length failed*" and "*Requirement on Wheel deflection failed*".

4.5. PMM and FMI

The Functional Mock-up Interface (FMI) is a standard that supports: (1) the exchange of models between simulation tools, and (2) the coupling of simulation tools in a co-simulation environment [28]. A model, a co-simulation slave or the coupling part of a tool is an executable called Functional Mock-up Unit (FMU) [28]. Consequently, all design models of PMM can be exchanged as FMUs as long as the preferred simulation tools comply with FMI. As PMM and FMI are in harmony, suppliers and acquirers can safely cooperate by exchanging models as FMUs in binary format that protect their intellectual property. Moreover, since many modelling and simulation tools comply with FMI, each domain expert is free to work with the tool that suits him best. In a nutshell, FMI is the bedrock of interoperability within and across the functional areas of the extended enterprise. Finally, FMI is of particular interest as PMM enables designers to reuse design models.

5. CONCLUSION

We may conclude that this experiment corroborates hypotheses. First, PMM is a Model Based Systems Engineering method that is relevant for continuous multi-physics systems. Second, the implementation of PMM models with Modelica enables engineers to carry out early requirements validation and designs verification processes. We can also confirm that this approach fills the gap between the operative rules [3, p 53] of systems engineering and the substantive rules [3, p 55] of engineering disciplines since PMM integrates engineering sciences in a coherent manner. As a future work, we plan to extend the current results to hybrid systems and to safety discipline.

REFERENCES

[1] Jackson, S., (2010), "Memo to industry: the crisis in systems engineering", INSIGHT, International Council on Systems Engineering, Vol 13-1, pp. 43.

[2] Hall, A.D., (1962), "A methodology for Systems Engineering", Van Nostrand, Princeton, New York.

[3] Micouin, P., (2014), "Model-based systems engineering: fundamentals and methods", Wiley & ISTE.

[4] ARP4754A., (2010), "Guidelines for development of civil aircraft and systems", SAE, Warrendale (PA).

[5] ANSI/EIA632., (2003), "Processes for engineering a system", GEIA, Arlington, VA.

[6] IEEE Computer Society., (1998), "IEEE guide for information technology – system definition – concept of operations document", IEEE Std 1362.

[7] von Lamsweerde, A., (2009), "Requirements engineering", Wiley.

[8] Hubka, V., Eder, W.E., (1988), "Theory of technical systems. A total concept theory for engineering design", Springer-Verlag.

[9] Micouin, P., (2008), "Toward a property-based requirement theory: system requirements structured as a semilattice", in: Systems Engineering, Vol. 11-3, pp. 235-245.

[10] IEEE Computer Society., (2008), "IEEE standard VHDL language reference manual", IEEE Std 1076-2000.

[11] IEEE Computer Society., (2007), "IEEE standard VHDL analog and mixed-signal extensions", IEEE Std 1076-1.

[12] Modelica Association., (2012), "A Unified object-oriented language for systems modelling language specification", Version 3.3, May 9, 2012.

[13] MathWorks., (2015), "Simulink User's Guide" R2015b, The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098.

[14] Thuy, N., (2014), "FORM-L: A Modelica Extension for properties modelling illustrated on a practical example"; proceedings of the 10th International Modelica Conference, March 10-12, 2014, Lund, Sweden.

[15] Jeannet., B., Gaucher, F., (2016), "Debugging embedded systems requirements with STIMULUS: an Automotive Case-Study", ERTSS 2016, Toulouse, France.

[16] Micouin, P., (2014), "Property model methodology: a model-based systems engineering approach using VHDL-AMS", in: Systems Engineering, Vol. 17-3, pp. 249-263

[17] Sadraey, H.M., (2013), "Aircraft design: a Systems engineering approach", Wiley.

[18] Raymer, D.P., (2006), "Aircraft design: a conceptual approach", AIAA Education Series, 4th edition.

[19] Currey, N.S., (1998), "Aircraft landing gear: principles and practices", AIAA.

[20] Suh, N.P., (2001), "Axiomatic design, advances and applications", Oxford University Press.

[21] Palm III, W., (2014), "System dynamics", 3rd edition, Mc Graw-Hill International Edition.

[22] Bunge, M., (2007), "Philosophy of science", volume 1: from problem to theory, Chapter 1, The scientific approach, Transaction Publishers, New Brunswick, New Jersey, 4th print.

[23] Tundis, A., Rogovchenko-Buffoni, L., Fritzson, P. and Garro, A., (2013), "Modelling system requirements in Modelica: definition and comparison of candidate approaches", 5th International Workshop on Equation-Based Object-Oriented Modelling Languages and Tools, 19 April, 2013.

[24] Kuhn, M.R, Otter, M, Giese, T., (2015), "Model Based Specifications in Aircraft Systems Design", Proceedings of the 11th International Modelica Conference, September 21-23, 2015, Versailles, France.

[25] Otter, M., Thuy, N., Bouskela, D., Buffoni, L., Elmqvist, H., Fritzson, P., Garro, A., Jardin, A., Olsson, H., Payelleville, M., Schamai, W., Thomas, E. and Tundis, A., (2014), "Formal requirements modeling for simulation-based verification", Proceedings of the 11th International Modelica Conference, September 21-23, 2015, Versailles, France.

[26] Lee, D.G., Suh, N.P., (2006), "Axiomatic design and fabrication of composite structures", Oxford University Press.

[27] Leibniz, G.W. (1714), " La Monadologie". http://classiques.uqac.ca/classiques/Leibniz/La_Monadologie/leibniz_monadologie.pdf (accessed on 24th January 2016).

[28] Modelica Association Project., (2014), "Functional Mock-up Interface for Model Exchange and Co-Simulation V2.0", July 25, 2014.