



HAL
open science

Property Model Methodology: A First Assessment in the Avionics Domain

Patrice Micouin, Louis Fabre, Pascal Pandolfi

► **To cite this version:**

Patrice Micouin, Louis Fabre, Pascal Pandolfi. Property Model Methodology: A First Assessment in the Avionics Domain. ERTS 2016 8th European Congress on Embedded Real Time Software and Systems, Jan 2016, Toulouse, France. hal-01273973

HAL Id: hal-01273973

<https://hal.science/hal-01273973>

Submitted on 15 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Property Model Methodology: A First Assessment in the Avionics Domain

Patrice Micouin¹, Louis Fabre², Pascal Pandolfi²

Abstract

The aim of this paper is twofold. Firstly, it is intended to provide an overview of the goals, the concepts and the process of a new Model Based Systems Engineering methodology, called Property Model Methodology (PMM). The second aim is to provide a feedback on its application in the avionics domain. In this experiment, PMM has been used in order to develop a top level specification model regarding a textual specification of an avionics function, to validate the top level specification model, and according to PMM rules to develop (1) a design model of the function taking into account architectural constraints of an integrated avionics, (2) building block specification models and (3) building block design models. Building block specification models were validated regarding their encompassing system specification model and the selected system design model while the design models were integrated and verified, level by level up to the top level design model, regarding their specification model. This paper summarizes the lessons learnt during this process and some additional results related to safety issues. This paper, with others [1,2], proves the fundamental concepts of PMM and provides a starting point for further research on Model Based Systems Engineering of a wide range of engineered systems (discrete, hybrid, continuous and multi-physics systems), but also support additional systems engineering activities (e.g. safety-reliability activities).

Keywords

Property-Based Requirement, Property-Model Methodology, Model-Based Systems Engineering, CONOPS, VHDL(-AMS), Modelica, Simulink, Validation, Verification.

1 Introduction

There is a general agreement on the idea that there is a crisis of the classical systems engineering [3], as well as there was a software engineering crisis starting from the nineties's. Whether we consider energy, automotive and transportation or space engineering industries, the symptoms of this crisis remain the same: delivery delays, cost overruns and a lack of maturity during system's infancy. According to Nam P. Suh's vision of complexity [4], system development processes are generally too complex i.e the probability, they achieve their goals on time and for an objective cost, is too low. Among the numerous causes of this crisis we can list a cumbersome document centric approach, the technical challenges assigned to the systems under development, as well as the large, multicultural geographically dispersed teams through whom systems are developed. Although there is a shared understanding of the classical systems engineering crisis, the proposals for resolving it diverge. First, there are the pragmatists who will put forward minimum corrective actions to obtain the presumed greatest improvements. For example, the creation of best practice guides and templates is such a solution advocated by the pragmatists. Others will look into more agile methods in order to reduce misunderstandings that abound in development teams. We could designate them as inter-subjectivists ("*people rather than processes*"¹). Finally, there are those

¹ <http://www.agilemanifesto.org/sign/display.cgi?ms=000000309>.

¹ Patrice Micouin
Arts et Métiers ParisTech, LSIS, UMR CNRS 7296,
2, cours des Arts et Métiers, 13617 Aix-en-Provence, France

² Louis Fabre, Pascal Pandolfi
Airbus Helicopters
Aéroport Marseille Provence
13 700 Marignane France
{louis.fabre, pascal.pandolfi}@airbus.com

who see a solution in rigorous formal processes. Although we consider that each approach contains a grain of truth and deserves to be explored, we undeniably side with the last one. Indeed, we claim that a formal development process including validated specification models and verified design models could solve the classical systems engineering crisis, whose principle was designed decades ago [5], by limiting the number of possible misinterpretations [1]. In this paper, (1) we shall present the Property Model Methodology (PMM), then (2) we will offer some insights into its utilisation through the case of an avionics function. Lastly, (3) we will summarize the main lessons learnt in this experiment and we will identify the way to go before PMM can be deployed operationally.

2 PMM: goals, processes and concepts.

PMM is an innovative model-based systems engineering (MBSE) method [6] focusing on operational goals. It is a top-down approach that authorises the re-use of pre-existing blocks at any hierarchical level of a system model's architecture. Another feature of PMM is that it complies with current industrial development standards, specifically ARP4754A [7] and EIA632 [8]. PMM is a method, derived from the scientific method [9, p 10-11] and applied for engineering systems, i.e. a sequence of operative rules, defined to build specification models and design models of engineered systems and expressed in a language. Finally, the third pillar of PMM is simulation, which is the primary means for validating specification models and verifying design models. The languages that can support directly PMM are simulation languages such as VHDL [10], VHDL-AMS [11], Modelica [12], Simulink [13] and the PMM concepts are directly mapped onto simulation language features to produce simulation models. Today, PMM has no connection with languages such as SysML [14] or Domain Specific Languages, while the OMG² is currently analysing the need to model requirements more clearly in SysML^{3,4}.

Roughly described, for an innovative system, PMM usually starts just after the validation of a Concept of Operations [15] and is made up of 4 activities⁵:

- (1) The first activity, which is carried out at the system level, is the development of a goal oriented system specification model associated with simulation scenarios defined in order to validate the system specification model. First innovation of PMM: it supports an early validation process based on simulation, before starting a solution definition process.
- (2) Then, we continue with a recursive process that consists in breaking down the system into building blocks. Each building block is made up of a specification model and one or more design model alternatives. Thus, a collection of competing system's architectures is considered, but only the preferred one is selected. The building blocks and their connections constitute the system's architecture. At each hierarchical level, the building block specification models are validated together against higher level building block specification model from the system specification model to the lowest level. A building-block is qualified as elementary when it is not decomposed any further and when its behaviour, is completely defined by a Behavioural Design Model (BDM) [6, p 133]. Libraries store the building blocks that might be re-used in future developments. From the point of view of its acquirer, any building block that is picked from a pre-existing library is also considered as an elementary building block. The design process ends when all the elementary building blocks have been identified, and modelled or acquired. The PMM recursive design process

² OMG = Object Management Group : <http://www.omg.org/>

³ <http://www.omg.org/issues/issue17016.txt>

⁴ http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-roadmap:requirements_modeling_working_group

⁵ It also includes recovery points to reengineer the system when a goal is not met.

- is, by several aspects, a very classical process. It is acquainted with different design methods such as the Suh's Axiomatic Design method [16], the Abrial's B method [17].
- (3) The third activity, after this top-down design process, is a bottom-up process that consists in recursively integrating the elementary building blocks and then the intermediate building blocks so as to finally end up with the complete system model. For each integration level, design models are integrated and verified against their own specification models. Second innovation of PMM: it supports a design verification process based on simulation before starting the physical implementation.
 - (4) The final steps that deal with the production of the physical building blocks (Hardware and Software), their integration, and the final installation of the system in its environment are out of scope of this paper, but we encourage readers to read [6, chap. 11] for further details.

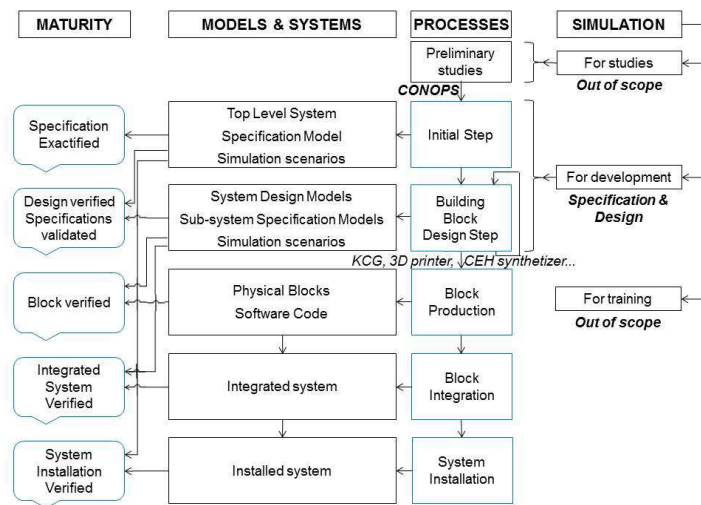


Figure 1 PMM processes.

2.1 PMM Specification Process and Specification Models.

Engineered systems are goal oriented. If they are correctly developed, they embody the intent of their designers. This is the reason why system specification is the first activity in system development. PMM proposes a Goal Oriented Requirement Engineering approach such as KAOS [18]. The specification modelling sequence is the third innovation of PMM. This is a very unusual backward process from the effects to the causes: (1) first of all, the specification process starts with the identification and definition of system goals. Goals are modelled as outputs of the specification model. (2) Then the actualisation conditions of these goals are identified. These conditions are modelled either as observable states (a special kind of PMM outputs) or as expected inputs. (3) Finally, the result of the analysis is formalized as one or several PBRs [19]. Comes the fourth innovation of PMM: PBRs are predicates that link goals, observable states and inputs in order to specify the actualization conditions of system's properties. The PBRs structure prevents introducing design biases at specification level. The same process applies for secondary⁶ (undesired) outputs such as system failures and inputs (expected or undesired). The PBR theory is based on the theory of properties [20] due to the Canadian epistemologist Mario A. Bunge.

⁶ According to a terminology due to Vladimir Hubka and Ernst Eder in [21]

The basic form of a PBR is as follows:

$$\text{PBR \#n : when } C \rightarrow \text{val(O.P)} \in D$$

This formal statement means: “when the condition C is true, the property P of the object O is actual and its value *shall* belong to the domain D ” where C is a relevant condition for the system or its environment: a functioning mode, a system state, an (undesirable) event, a time delay or a combination of such features and where the domain D is a finite or infinite set such as {on, off} or \mathbb{R}^n (possibly linked to a frame and a physical unit).

The concept of a PBR can be implemented as a conditional assertion (Boolean function) with various simulation languages such as those mentioned above.

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

Based on Bunge’s property algebra, several PBRs can be combined thanks to the conjunction operator “ \wedge ” in order to build composite PBRs. In a dual way, the partial order relationships “ \leq ” and “ \geq ” enable analysts to compare two PBRs. Thus, the expression “ $\text{PBR}_1 \wedge \text{PBR}_2$ ” is the conjunction of PBR_1 and PBR_2 and is itself a PBR. Moreover, the statement “ $\text{PBR}_1 \leq \text{PBR}_2$ ” means PBR_1 is less constraining than PBR_2 .

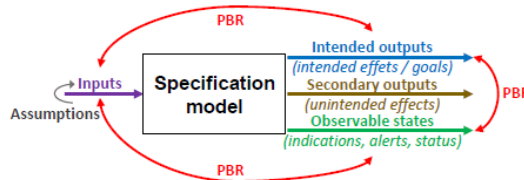


Figure 2 PMM Specification model

A specification model is a formal model that includes (1) system requirements, (2) system interface requirements and (3) system assumptions. The expression of specification models into simulation models (VHDL-AMS, Modelica or Simulink) is a solution to make sure that specification models and interfaces are coherent. In addition, the simulation of a specification model linked with the corresponding equation design model (EDM) [6, p 139] provides analysts with various advantages. First, it is an assistance to guarantee the completeness and correctness of the top level system specification model for a given set of validation scenarios. Second, it provides also a visual means to assist stakeholders such as pilots, flight engineers, authorities in the validation of the top level system specification model before starting a design on the basis of a specification model presumed free of errors. Third, as we will explain, simulation provides also capabilities for validating building block specification models against higher level building block specification models up to the system specification model. This is also an innovation of the PMM process which leads us to the “Prime contractor theorem” described below.

2.2 PMM Design Process, Design Models and PBR Derivation

Once the system specification model is complete and correct, the second system development activity consists in developing a system design model. This is a design process very similar with the Suh’s Axiomatic Design “zigzagging process” between specification and design models. For a very simple system, the system design model is usually an EDM. However, for more complicated systems, Structural Design Models (SDMs) [6, p 145] are introduced. A SDM is a formal definition of a system’s 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 $\{PBR_1, \dots, PBR_n\}$. To be valid, for a given system structural design model \mathcal{A} and for a set of assumptions on properties own by objects that belong to the environment \mathcal{EA} , the conjunction of the derived building-blocks PBR_1, \dots, PBR_n shall be more constraining than the system PBRs. This statement can be expressed formally as:

$$\text{Derivation (PBR): when } \mathcal{A} \wedge \mathcal{EA} \rightarrow PBR \leq PBR_1 \wedge \dots \wedge PBR_n$$

The 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 set of validation scenarios (Fig. 3); The extend of the set depends on the level of validation rigor.

This validity condition of PBR derivation leads to the “**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.”

This result of PMM, shared⁷ with the B method, is a key point for a prime contractor to secure his subcontracting process.

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

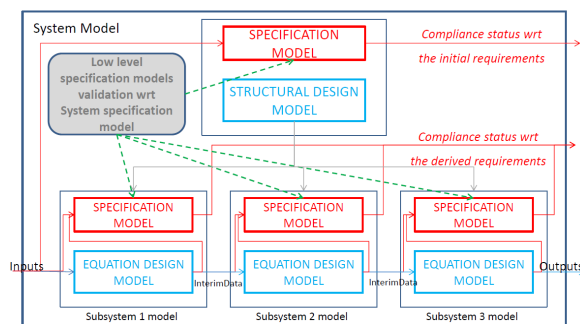


Figure 3 Validation process of subsystem specification models

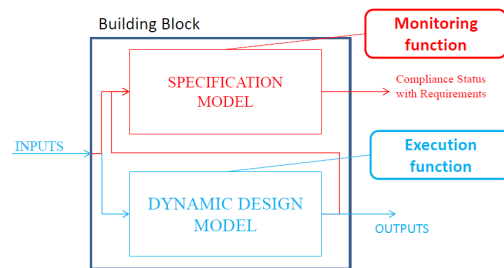


Figure 4 Specification and design model roles during a verification process

Simulation also provides capabilities for verifying design models. While the simulation is running, for all submitted simulation scenarios, Fig. 4 shows that the specification models monitor the interacting design models so as to check whether any requirement is violated. If there is no violation detected during the verification of a building block design model, then the building block design model may be presumed as free of design errors.

To complete the picture, the PBRs theory provides us with a theorem of practical importance, the “**Safety theorem**”:

“The introduction of any additional PBR at any building-block level of a system does not impair the safety of this system. It can impair its feasibility.”

⁷ However, keep in mind the differences between evidences provided by simulation and formal proofs.

Obviously, this theorem, established in the frame of the PBR theory, is wrong in the context of textual requirements.

3 “Clearance Test Function” experiment

The case study comes from the avionics domain and is based on an avionics function called “Clearance Test Function” (CTF). This is a limited experiment selected in order to prove the PMM concepts and to identify benefits, limitations and possibly improvements. The VHDL language was used as target simulation language. Specification and design models were written in VHDL and the ModelSim[®] simulator from Mentor Graphics was used. Note that a translation schema of PMM models in VHDL models is provided in [22].

3.1 Experiment context

The main goal of the “Clearance Test Function” is to provide the crew with a status of various systems and equipment installed inside the helicopter, before taking off. In conjunction with the MMEL⁸, it provides to the crew with an aid to clear or not the flight depending on the availability and the healthy status of systems and equipment such as sensors, actuators, control units, etc.. The “Clearance Test Function” is activated on ground when the helicopter is powered on. It includes several sequences of tests, triggered automatically or upon crew request. The function consists in the reporting of synthesized messages for the crew computed on the basis of PBITs, CBITs and IBIT⁹s produced by the systems and equipment involved and H/C or avionics states. The “Clearance Test Function” is a decision making aid for the crew.

At the time of the experiment, this function was under development in the frame of a new avionics suite developed by Airbus Helicopters and called “Helionix[®]”. It is now installed on several in service helicopters such as the H175, H145.

At the beginning of the experiment, (1) a specification including 200 textual requirements and (2) a preliminary Rotorcraft Flight Manual (RFM) describing pre-flight tests were available and were foreseen to be used as main inputs for the experiment. Additionally, functional and physical architectural choices of Helionix[®] avionics suite, were well known at this time.

3.2 PMM application

This paper describes the PMM application, but practical examples cannot be disclosed in this paper. Disclosable pieces of examples will be part of the conference presentation.

Specification:

After an initial learning phase of the input data, the first step of the experiment consisted in the establishment of the top level PMM specification model of CTF. With the objective to start from the operational need, the RFM was demonstrated to be a better source than textual requirements to produce the PMM specification model because it was really representative of operational need (as a CONOPS would represent this need).

In order to build the top-level specification model, the messages indicated in RFM were inventoried and grouped depending on the progress in the test sequence. These groups were modeled as intended outputs of the specification model. The possible values for each output corresponded to a specific message displayed to the crew.

⁸ MMEL=Master Minimum Equipment List

⁹ (P=power-up, C=continuous, I=initiated)BIT=Built-in Test

PMM: A First Assessment in the Avionics Domain

After this phase of identification of expected outputs, their actualization conditions were analyzed. For instance, the analysis provided us with the following conditions: the message “Pump xx failed” (intended output) shall be displayed to the crew 2 sec at most after a “Pump test” (modeled as observable state) request sent by the crew when the following conditions are fulfilled: the helicopter is on ground (modeled as observable state), one engine at least is started (modeled as observable state), and a Fail IBIT status from Pump xx” (modeled as input) is received by the avionics.

The same analysis was performed for each specific message and for each group. Although somewhat tedious, it was easy to represent these conditions as PBRs or, what amounts to the same, as boolean functions. Translation errors were possible but would be detected during a sufficiently rigorous validation (obviously this event occurred).

After each intended output associated with its PBRs, the same procedure was applied to the observable states in order to specify the actualization conditions of observable states.

Finally, assumptions about the inputs (values, frequency, sequencing, ..) were also modeled as PBRs, completing a first version of the specification model of the CT function.

This PMM specification model was made up of 14 property based requirements (PBR) prescribing completely the external reactions of the CTF function to handle about 200 external events coming from its environment and also few dozen internal events. It was not necessarily consistent with the real need, it could contain specification errors but it was formally coherent and formally complete, which is rarely the case for textual specifications.

Validation:

It is the responsibility of the validation process of showing that the specification is as exact as possible, that is to say, it characterizes the right system i.e, the expected system. Although tedious, in PMM, it is quite easy to build validation scenarios from the expression of PBRs and this should be largely automated. This ability to automatically generate validation cases is another feature of PMM validation process. We have built a simple scenario to validate the specification model.

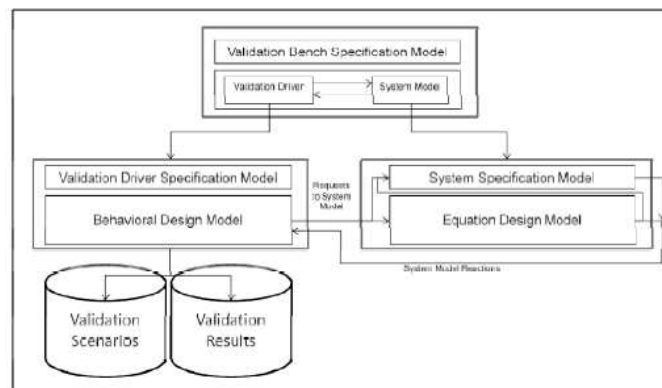


Figure. 5. PMM validation process of a specification model.

The next step was to transform the CTF Specification Model into an Equation Design Model according to the PMM process. As expected, the experiment showed that this transformation was direct, but it was handmade for lack of available transformation tool. The CTF system model, comprising its top level Specification Model and its equation design process was embedded in a validation bench with a validation driver in order to perform the validation process of the CTF top level Specification Model by the means

PMM: A First Assessment in the Avionics Domain

of simulation (Fig. 5.). This validation process, performed with a moderate effort, showed that the CTF system model outcomes were in line (after some corrections) with the preliminary RFM.

Quickly, the PMM specification model of CTF became a precise representation of the observable behavior of CTF, reflecting expected effects and their conditions of actualization as they were described in the RFM, while staying absolutely free from any design choices to implement the function. On the basis of this validation process, the CTF Specification Model was considered as validated (as exact as possible), i.e. in line with the right function.

Fourteen property based requirements were defined to describe the complete expected behavior of CTF. However, more than the significant reduction of requirements, which is not, in itself, a good indicator, what should be noted is (1) the accuracy of PBRs, (2) their formal coherence and formal completeness, (3) their ability to be derived in cases and scenarios of validation and verification and (4) their lack of design bias. They allow very firmly separating what concerns the specification of which is within the design.

Design and PBR derivation:

The next step was to model a PMM Structural Design Model (SDM) of CTF consistent with the Helionix[®] avionics system architecture in terms of behavioral and physical breakdown. The CTF SDM had to fit the Helionix[®] integrated architecture, its hardware and software components and the space and time partitioning of its resources. This implementation was time consuming, it was not held back by any theoretical or practical issues, but by the hand-made and boring writing of the CTF models. This activity showed that it was possible to submerge CTF function design model in the avionics system design model, and to comply with the constraints of its architecture, respecting its breakdown and the allocation of resources according to the defined temporal scheduling.

In the case, the derivation of PBR was not an issue; every component has been assigned a contribution to satisfy the mother PBR. Obviously these derived PBRs (in accordance with ARP4754A's derived requirement definition) were not directly traceable from the mother PBR and depended on Helionix[®]'s architecture choices and would be very different in the context of a federated architecture, for example.

As a first consequence of the use of a simulation language such as VHDL, the various configurations of models built during the development process were formally coherent and formally complete. In particular, the interfaces of all the building blocks composing each CTF system model configuration were coherent and complete.

Verification:

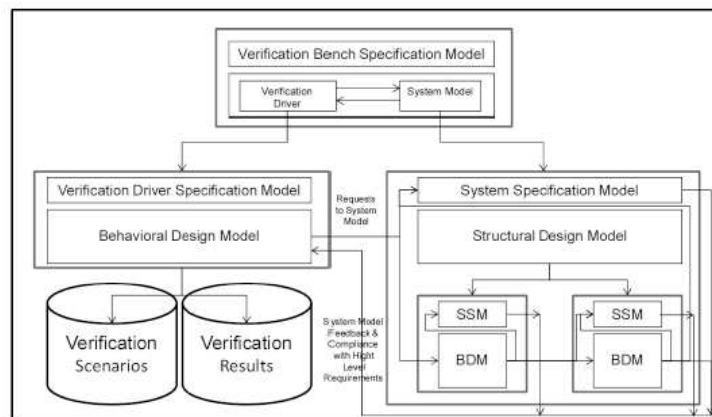


Figure. 6. PMM verification process of a design model.

As fourth step, the CTF system model, comprising its top level Specification Model and its SDM was embedded in a verification bench with a verification driver in order to perform the verification process of the CTF System Model by the means of simulation (Fig. 6.). This verification process, performed with a moderate effort, showed that the CTF design model was in line with its specification models (after several corrections). For the performed effort of verification, the SDM was deemed error free. Deadline requirements were satisfied by the temporal scheduling (with assumptions on worst case execution time). During this experiment, we found that it was possible to automatically generate simulation cases from the structure of PBRs depending on the level of criticality of the system considered. These simulation cases put together in simulation scenarios were used as validation scenarios for specification models and verification scenarios for design models. These simulations scenarios could also be re-used for the testing of physical products [6, p 162-167]. Thus, PMM allows, on the specification models basis, to size early testing effort which must be performed and almost automatically generate scenarios corresponding to the required effort.

Safety considerations:

The final step of the experiment consisted in specific investigations about safety requirement modeling and the verification of the efficiency of design mechanisms to prevent failures. The experiment allowed to verify whether a given architecture could tolerate faults or not, thanks to a special device of PMM, the RDM [6, p 207]. The Reliability Design Model (RDM) is a design model derived from a structural design model (SDM) that allows us to observe the behavior of a system model in the presence of single or combined faults. It has been shown that, for a moderate level of verification rigor, a system model considered in the experiment properly met the requirement “*no single failure will result in a catastrophic condition*” according to AC29-2C, F-17 [23].

4. Lessons learnt and conclusions

The experiment described above was conceived as a proof of concept and it is not excessive to say that the Property Model Methodology concept has been validated beyond the initial expectations. Not only, all the expected confirmations concerning the relevance of PMM concepts and PMM methodological process have been obtained but further results were achieved.

First of all, the experimentation established the suitability of the concept of PBR and showed that the process of specification associated therewith allows the production of Specification Models without design bias. Then, the experiment confirmed that the Equation Design Models allowed the validation of specification models and showed that the former could be produced automatically from the latter. It confirmed also that the validation and verification cases could be automatically generated from the structure of PBRs with a gradation depending on the level of rigor requested.

The experiment showed that it was possible and quite easy to design a CTF solution that was consistent with architectural choices made for reasons (Helionix[®] architectural principles) beyond the specific needs of the developed function.

The last but not the least, the experimentation has highlighted several possible non-recurring cost (NRC) savings in the development of a system: (1) a system model formally coherent from the top to the end building blocks and free of interface errors (2) specifications interpretable as little as possible while misinterpretations are source of errors and waste of time, (3) early requirement validation through simulation before any physical (hardware or software) realization of the products composing the system, (4) a design verification through simulation and, in particular, of failure prevention mechanisms, (5)

PMM: A First Assessment in the Avionics Domain

automatic generation of unit test cases, integration test cases for the physical system and its building blocks and installation test cases of the system into its environment.

However, despite these results, the deployment of PMM is not immediately feasible. Indeed, during the experiment, the models were produced by hand by an engineer familiar with VHDL. This prerequisite is not acceptable if PMM is applied by engineers from different engineering disciplines. More, existing simulation tools are design oriented, bottom-up oriented and ignore specification aspects and, a fortiori, the theory of PBRs. So, a PMM front-end, hiding the underlying language for modeling PMM artifacts in a way acceptable by discipline engineers, and especially for modeling PBRs and for producing specification models, remains to be developed. This objective is at hand regarding simulation languages such as Modelica [2] and Simulink. It remains to be determined regarding VHDL and VHDL-AMS. An alternative way to this approach targeting a particular simulation language is of a PMM Modeler based on a pivot language and capable of generating models expressed in various simulation languages. This is the way that the Ellidiss company chooses to explore thanks to its LMP technology [24].

References:

1. Poupart E, Wallut J.M, Micouin P, Property Model Methodology: A First Application to an Operational Project in the Space Domain, CSD&M 2015, November 2015.
2. Pinqu   R, Micouin P, V  ron Ph, Segonds F: Property Model Methodology: a case study with Modelica, to be published.
3. Newport, J.R.: Avionic Systems Design, CRC Press (1994)
4. Suh N.P, Complexity, Theory and Applications, Oxford University Press, 2005.
5. Hall, A.D., (1962), A methodology for Systems Engineering, Van Nostrand, Princeton, New York.
6. Micouin, P.: Model Based Systems Engineering: Fundamentals and Methods, Wiley & ISTE (2014).
7. ARP4754A, Guidelines for Development of Civil Aircraft and Systems, SAE, Warrendale (PA), 2010
8. ANSI/EIA632, Processes for engineering a system, GEIA, Arlington, VA, 2003.
9. 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.
10. IEEE Standard VHDL, (2008), Language Reference Manual, IEEE 1076, IEEE Computer Society.
11. IEEE Standard VHDL Analog and Mixed-Signal Extensions, IEEE 1076-1, IEEE Computer Society, 2007.
12. Modelica Association, Modelica^{  } - A Unified Object-Oriented Language for Systems Modeling Language Specification Version 3.3 May 9, 2012.
13. Karris, S.T: Introduction to Simulink with Engineering Applications, Orchard Publications, 2011
14. OMG Systems Modeling Language (OMG SysMLTM), Version 1.4, September 2015, <http://www.omg.org/spec/SysML/1.4/>
15. IEEE Standard 1362, IEEE Guide for Information Technology – System Definition – Concept of Operations Document, 19 March 1998.
16. Suh N.P, Axiomatic Design, Oxford University Press, 2001.
17. Abrial J-R, Modeling in event-B, System and Software Engineering, Cambridge University Press, 2010.
18. Lamsweerde von, A. Requirements Engineering: From System Goals to UML Models to Software Specification, Wiley (2009).
19. Micouin, P., (2008), “Toward a property-based requirement theory: system requirements structured as a semilattice”, in: Systems Engineering, Vol. 11-3, pp. 235-245.
20. Bunge. M, Treatise on Basic Philosophy, vol 3, Ontology I: The furniture of the World, D. Reidel Publishing Compagny (1977).
21. Hubka V, Eder W.E: Theory of Technical Systems: A Total Concept Theory for Engineering Design, Springer-Verlag, 1988.
22. Micouin, P., (2014), “Property-Model Methodology: A Model-Based Systems Engineering Approach Using VHDL-AMS”, in: Systems Engineering, Vol. 17-3, pp. 249–263.
23. Federal Aviation Administration, AC29-2C Certification of Transport Category Rotorcraft, latest version.
24. Dissaux P., Hall B. Merging and Processing Heterogeneous Model, to be published, ERTSS 2016