Property Model Methodology: A First Application to an Operational Project in the Space Domain

Erwann Poupart, Jean-Marie Wallut and Patrice Micouin

Abstract The purpose of this paper is to provide a feedback on a Model Based Systems Engineering application to a space domain project. In the core of the paper, and after a synthetic presentation of the systems engineering methodology called Property Model Methodology (PMM), the case study, coming from the space domain, is described. In this context, PMM has been used in order to validate a top-level textual specification and to define the verification scenarios and verification cases aiming at establishing the correctness and the completeness of the physical system developed according to this top-level textual specification. The paper provides first feedbacks about PMM utilization. The conclusion summarizes the benefits and also the limitations that are identified today, and includes a presentation of the future works.

1 Introduction

There is a general agreement on the idea that there is a crisis of the classical systems engineering [1], as well as there was a software engineering crisis starting from the ninety's. Whether we consider the domains of energy production, of transport vehicle industry (road, rail or air) and even in space industry, the manifestations of this crisis are still the same: delivery delays, objective cost overruns and lack of

E. Poupart (✉) · J.-M. Wallut
CNES Centre Spatial de Toulouse, 18 Avenue Edouard Belin, 31401, Toulouse Cedex 9, France
e-mail: erwann.poupart@cnes.fr

J.-M. Wallut
e-mail: jean-marie.wallut@cnes.fr

P. Micouin
Arts et Métiers ParisTech LSIS UMR CNRS, 2 Cours Des Arts et Métiers, 7296, 13617 Aix-En-Provence, France
e-mail: patrice.micouin@incose.org

G. Auvray et al. (eds.), *Complex Systems Design & Management*,
DOI 10.1007/978-3-319-26109-6_12

maturity of the systems put into service. If the causes are certainly many, one of them is the growing gap between the means used (a widely document-centric engineering) and, on the one hand, growing objectives assigned to systems under development and, on the other hand, the conditions in which these systems are developed (large teams, from various geographical, linguistic and, cultural areas). From this generally shared understanding, the proposals for resolving this crisis diverge. For pragmatists, it consists of detecting the minimum corrective actions to obtain the greatest improvements, such as the establishment of good practice guides. 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, the last one, see a solution in a strengthening of formality and rigor of the implemented processes and exchanged products. If, in our opinion, each of the above approaches has a grain of truth and deserves to be explored, we side clearly with the last one: we assume that the crisis of the classical systems engineering, of which principles were designed decades ago, can find a solution thanks to formality in the development processes, and the exchange, between stakeholders, of engineering products as little interpretable as possible and as accurate as possible, starting with validated specification models and verified design models. In what follows, we present PMM method, we provide a report of its use in the context of a space project and we make a feedback on lessons learnt.

2 PMM: Goals, Processes and Concepts

PMM is a recently developed Model-Based Systems Engineering methodology [2]. Its compound name comes from two of its main characteristics, (1) the formulation of requirements based on the concept of property (property-based requirements or PBR) and (2) the adoption of a model-based systems engineering (MBSE) approach. This is a very classical descending development approach; however it authorizes the reuse of preexisting blocks. This development approach is compatible with current industrial development standards, specifically ARP4754A, EIA632 or Space Engineering Standards. It is also built on a third pillar, namely simulation, which is the primary means for validating specification models and verifying design models, while the verification of physical products, their integration and installation are maintained (Fig. 1).

Roughly described, PMM may start just after the validation of a Concept of Operations (CONOPS) [3] and is made up of the following activities (obviously, it includes a lot of recovery points to reengineer the system, when goals are not the right ones or are not reached):

[1]Agile Manifesto: http://www.agilemanifesto.org/sign/display.cgi?ms=000000309.

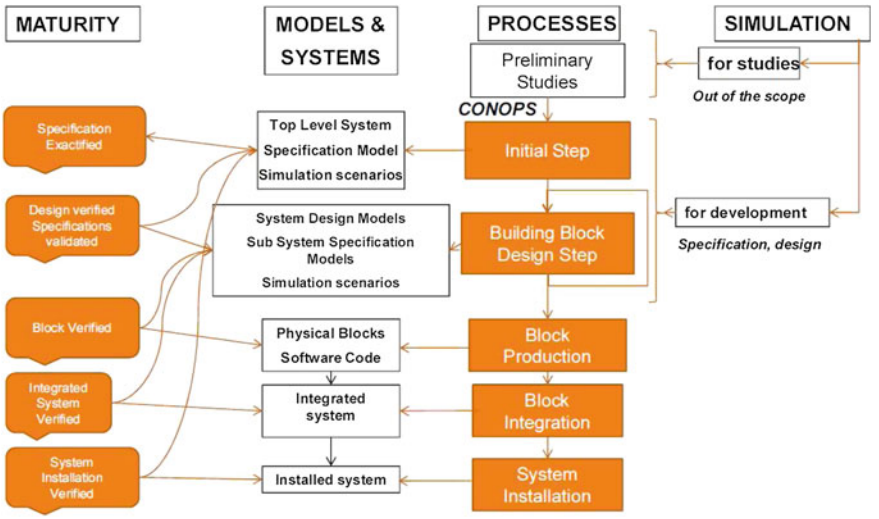

Fig. 1 PMM process model

(1) The initial step includes the establishment of the top level system specification model and the elaboration of simulation scenarios built in order to validate the top level system specification model. The system model is ready for validation. When this goal is reached, the top level system specification model may be considered as exact (as exact as possible).

(2) Subsequent steps are a finite repetition of a building block design step including the establishment of a system design model providing us with an architecture of the building block (alternate architectures may be considered and a preferred one may be selected). When this architecture introduces subsystems, equipment, items or parts, a specification model for each of these components is established, while the connections among these components are stated. Simulation scenarios are built of each of these component specification models. When this goal is reached, the low level subsystem specification models are validated step by step against the system specification model. The design model of a building block is not further decomposed when its behavior may be directly formalized as equations or when it may be picked up from a building blocks catalogue.

(3) The design process ends when all the elementary building blocks are designed or acquired. Then, step by step, elementary building block design models, and integrated building block design models are verified against their own specification models.

(4) Production and verification processes of elementary physical building blocks, their physical integration into intermediary building blocks and the associated verifications are described in [2, Chap. 11].

(5) When the top level system which is fully verified against its specification, it is installed in its environment and its operation is verified in accordance the various operation scenarios resulting from the CONOPS and partially or totally included in top level simulation scenarios.

2.1 PMM Specification Process and Specification Models

The first system development activity consists of establishing a top level system specification. According to PMM, a system specification process starts with the definition of the system goals (i.e., its intended effects or its functions) identified and modeled as outputs of the top level system specification model. PMM requirement determination approach is goal oriented, similar to those supported by Goal Oriented Requirement Engineering approaches such as KAOS [4].

Based on this goal identification, the occurrence conditions of these goals are elicited. During this process, expected inputs are identified while other outputs may be also identified and modeled, such as observable states, undesired effects and system failures. Undesired inputs may be also considered.

Then, we formalize the result of this elicitation in the form of PBRs [2, chap 6]. These PBRs are predicates linking together goals, secondary outputs, observable states and inputs. They specify system properties and their conditions of actualization. In particular, assumptions are specific PBRs related to input properties since they are always out of the system's developer control and only presumed (Fig. 2).

The basic form of a PBR is as follows:

$$PBR\text{: } [when\ Condition\ =\ >\ ]\ val(System.Property)\ \in\ Domain,$$

A conjunction operator "∧" of PBRs is defined allowing the combination of several PBRs while a partial order relationship "≤" allows to compare two PBRs. PBR1 ∧ PBR2 is the conjunction of two PBRs and is also a PBR while $PBR_1 \leq PBR_2$ means PBR_1 is less constraining than PBR_2.

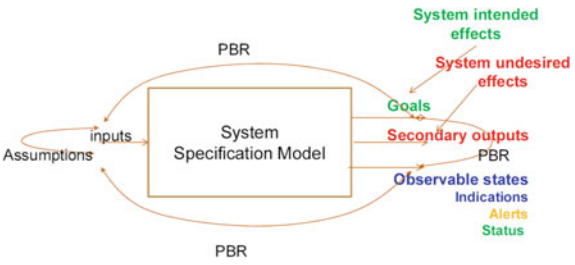

Fig. 2 Specification model

A specification model is a formal model dealing with (1) system requirements, (2) system interface requirements and (3) system assumptions. Translated in simulation models based on languages such as VHDL-AMS [5] or Modelica [6], a system model is syntactically coherent and complete. Simulation provides assistance for establishing the exactness of a system model for a defined set of validation scenarios.

2.2 PMM Design Process, Design Models and System Verification

When the exactness of a system model is established, the second system development activity is to conceive a system design model. This activity is based on the designer's knowledge of business rules, experience on the system domain and innovations. Several candidate architectures may be considered. Since the focus of this paper in not on design activities, they are not described here (they are detailed in [2, Chap. 7]).

For candidate architectures, the third activity is to derive the system requirements (PBR) into subsystem requirements $\{PBR_1, \ldots, PBR_n\}$. To be valid, such derivations shall be such that, when the architecture A is selected, then the conjunction of $\{PBR_1, \ldots, PBR_n\}$ shall be more constraining than the system requirement PBR:

$$Derivation: when\, A\, = > PBR \leq PBR_1 \wedge \ldots \wedge PBR_n,$$

This validity condition entails the following theorem (called "*the prime contractor theorem*"): "*A sufficient condition for a system to comply with its PBRs is that its subsystems 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*".

Simulation provides, firstly, a means for establishing the validity of system PBRs derivation into a set of subsystem derived PBRs for a given level of rigor (related to the richness of validation scenarios). Secondly, simulation provides a means for verifying building block design models regarding their specification models.

During simulation sessions, the specification models monitor the corresponding design models, checking that for all submitted simulation scenario whether requirements are violated or not. When no requirement violation is detected for the complete verification of a building block design model, the building block design model may be considered as free of error regarding its specification model and for the considered effort of verification. And so on, up to the system level.

3 The Application: PEPS System Modeling

3.1 Description of the PEPS System

PEPS stands for Environmental Politics & Space Politics. It aims at developing usage of space images. This new CNES project, that has started its operational phase the 18th of June this year, is also part of Copernicus project (2014–2020) which is the most ambitious Earth observation program to date.

Copernicus is the new name for the Global Monitoring for Environment and Security program (GMES). This initiative is headed by the European Commission (EC) in partnership with the European Space Agency (ESA). ESA is developing a new family of satellites called sentinels that will provide a unique set of observations, starting with the all-weather, day and night radar images from Sentinel-1A, launched in April 2014. Sentinel-2 will deliver high-resolution optical images for land services and Sentinel-3 will provide data for services relevant to the ocean and land. Sentinel-4 and -5 will provide data for atmospheric composition monitoring from geostationary and polar orbits, respectively. Sentinel-6 will carry a radar altimeter to measure global sea-surface height, primarily for operational oceanography and for climate studies.

PEPS is the French ground system that will provide a public access to sentinels image products.

For the first time, multi-sensor, multi-scale and multi-temporal data over the Earth will be available for free which should help to create and develop environment services.

Users will be able to analyze data over a long period up to the beginning of the mission (climatic change: glacier footprint, ice field surface, lakes surface, desertification, vegetation indication), deforestation, urban expansion, road network, hydrology, volcanology, etc.

In 2017, PEPS system shall be able to store and provide access to 6.1 petabytes of data and 8 millions of products (image products, metadata and quick look). It shall also be extensible and able to store and provide access to 17 petabytes of data and 20 million of products in 2020.

After the end of 2017, PEPS system shall also provide, close to image products, high performance computing capability so that scientists or other external partners interested in developing geographic added-value services can run efficiently dedicated algorithms.

3.2 PMM Application Context and Goals

It is well known that requirements are the corner stone between specification/design, effective system (or implementation), and tests (verification that the system meets its requirements).

Previous R&T study at CNES (Limbes R&T project in 2008) has shown benefits to build a property centered design to ease system verification.

More than that, much more benefits can be obtained with higher quality of requirements in terms of semantic due to the many human actors involved in engineering requirement processes.

When system specification has been produced, reviewers will have to share a common understanding of requirements and produce RID's (Review Item Discrepancy) and this for each phase of the system engineering process:

- Phase A (Mission/operational analysis and feasibility):
  - Customer requirement review
- Phase B (Ground Segment Preliminary Design):
  - System requirements review
  - Preliminary design review
- Phase C (Ground Segment detailed design generally done by a sub-contractor):
  - Critical design review
- Phase D (Ground Segment production and verification):
  - Technical qualification review
  - Operational qualification review

Just after phase B and just before phase C, subcontractors involved in tender responses will also have to share a common understanding of requirements to design the most competitive solution.

In the same time, operational teams involved in the system as a part of it, will also have to design their tasks using the system to achieve mission goals.

Finally, requirements semantic will be checked once more during critical design review of phase C, and again to verify that the system including operations meets requirements during phase D reviews.

There is clearly a huge potential of improvement of engineering processes efficiency if requirements quality is improved (less ambiguity in semantic interpretation).

PMM provides a methodology and models to help system engineers to focus and make more explicit system goals. The main difficulty will be to integrate it successfully with the different human actors involved in the engineering process.

CNES had already made an experiment of PMM in the context of an R&T study in 2014 and developed a modelling front-end mock-up, named PMM Designer that was first experimented with a satellite imagery mission control system specification.

Even if the scope of the case study was limited in size, the result was that PMM concepts are robust to express requirements semantics, simple to use (due to its goal orientation, its concepts parsimony), and, even if the front-end mock-up could be enhanced, many potential benefits have been identified during this experiment, including:

- Improved quality and completeness of requirements,
- Efficiency of production and validation processes of requirements
- Efficiency of verification processes
- Efficiency of system maintenance in operational condition and its design

That were the reasons why we applied partially PMM to express PEPS system requirements more formally when PEPS system verification task started in March this year. At this time, PEPS specifications were already written in textual form in one document for all the releases from 1.1 until 1.4. They represent in total around 180 textual requirements.

The main goal was to assess benefits on our engineering process and limitations of the methodology and associated tool.

3.3 PBRs Determination and PBRs Validation

The first step was to create all the PBRs starting from PEPS system textual requirements release 1.1 (about 45 textual requirements for this 1.1 release). This formalization task was relatively efficient. Only two man weeks were necessary with the front-end mock-up to create a first release of 11 PBRs covering all the 45 requirements.

The Fig. 3 here after represents a snapshot of PMM Designer describing the PMM Top-Level Specification Model of PEPS acquisition System, with its expected outputs (goals), observables states and assumed inputs linked together thanks to PBRs (predicates) defining the outputs and their actualization conditions, assumptions on inputs, and so on. We started from 12 requirements and only 4 PBRs were sufficient to express PEPS acquisition system goals.

We observed that many textual requirements are very precise for the constraints about inputs but not correlated with the expected outputs and many missing constraints about outputs. We had to look at the implementation to characterize the intended effects.

Formalizing requirements into PBRs does gather execution context constraints with the expected outputs, so it gathers many textual requirements in one place.

Concretely, we have observed that 45 textual assertions, usually referred as textual requirements, were consolidated and restructured into 11 PBRs. Moreover, we significantly improved the quality of these requirements (providing rich, coherent and synthetic information contents).

Described in PBRs form, requirements are then verifiable, system can be tested, and moreover, other people responsible for designing and running the tests don't have to interpret again textual requirements. This requirement clarification task is done once and only once. We have already save some time to design the tests and we expect to save much more time in discussions and meetings usually spent by many people involved in the project trying to interpret textual requirements written by someone else.

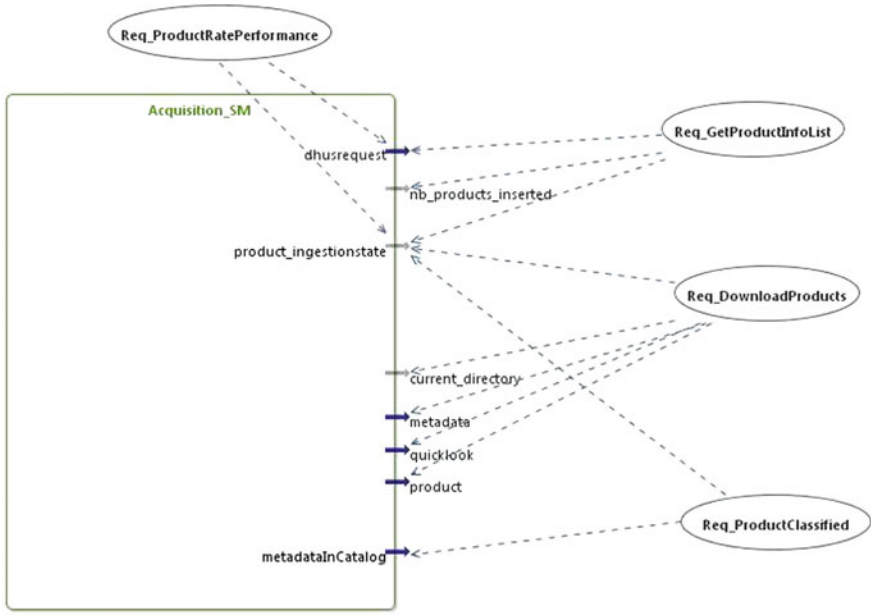

Fig. 3 PEPS acquisition system specification model with PMM designer

Another lesson learnt is related to security requirements; we had to take into account about 300 textual requirements. However, as there was already scripting tests that are able to check precisely those requirements, it was not useful to apply PMM to build the tests. We noticed that requirements were already unambiguous, for each requirement assertions about operating system properties are checked. They had spontaneously the structure of PBRs.

As a first example the PBR called "**Req_DownloadProducts**" in Fig. 4 specifies that, 12 h, at most, product items shall be complete on PEPS infrastructure and that we shall have an error to be handled by the human operator only if automatism cannot handle it.

Human operators are also goal driven and will compensate automatism when the goal is clearly defined and if he can have adequate observability of the system.

Note: Some logical operators are in comment because there are not available currently in PMM designer mock-up.

Writing PBR asks good questions to the specification author, it helps him or her to characterize a realistic and verifiable goal. For example, reading requirements ACQ-1810, ACQ-1015, ACQ-2030 we had to answer to the following questions: What is product item integrity? All product items shall be present or not? How can it be verified?

About the question: "*Where product shall be stored*"? We found 4 requirements (ACQ-2010, ACQ-2020, ACQ-2030 and AUT-0120) scattered in the document but only one PBR was sufficient using infrastructure architecture knowledge.

```
predicate Req_DownloadProducts(product_ingestionstate, current_directory, metadata,
quicklook, product):
  when (product_ingestionstate.downloadTaskStatus == "DOWNLOADING") delay 12 hours =>
      // ACQ-1015 => ACQ-2030 (name of product items)
      product.name == product_ingestionstate.productId//".zip" and
      metadata.name==product_ingestionstate.productId//"_Medata.xml" and
      quicklook.name==product_ingestionstate.productId//"_quicklook.gif"and
      // ACQ-1810 (product item integrity)
      //(product_ingestionstate.md5 == md5(product)) and
      // unzip -lz OK (readable zip) and
      // metadata starts with tag <product> and ends with tag </product>and
      // ACQ-2010, ACQ-2020, ACQ-2030 and AUT-0120 (product localization)
      ($current_directory == YYYY/MM/DD/SSS/) and (SSS == S1A or S1B or etc. satellite)
      // ACQ-0820 (failure handling)
      // if only uploading process running
      ((product_ingestionstate.downloadTaskStatus == "DONE") and
      (product_ingestionstate.ingestionTaskStatus == "TODO")
      or
      (product_ingestionstate.downloadTaskStatus == "ERROR") and
      (product_ingestionstate.ingestionTaskStatus == null))
```

Fig. 4 PEPS acquisition system PBR first example

We had also to identify missing observable state for the download task and to specify more precisely what to do in case of external site error (requirements ACQ-0820 that says "*After any failure, acquisition function shall resume from its current state*."). If the error is only temporary, download task automatism shall retry, otherwise download task shall go to "error state" to be handle by human operator. The existing prototype implementation helped us to identify the missing observable states and intended effects.

This was not so easy to define because acquisition system is not able to distinguish surely between permanent or temporary errors because it does catch external site error messages that are not precise enough. PBR has to take into account what the system is able to perceive from the environment otherwise we will obtain an unreachable goal.

As a second example the PBR called "**Req_ProductClassified**" in Fig. 5 specifies that, 12 h, at most, product items shall be classified and stored in PEPS catalog infrastructure and that we shall have an error to be handled by the human operator only if automatism cannot handle it.

It shall be noted that there were **no requirements specifying this acquisition system goal about product classification**. This missing requirement has been identified during this PBR determination and validation phase. Product classification is an intended effect that required to be characterized.

```
predicate Req_ProductClassified(metadataInCatalog, product_ingestionstate):
when (product_ingestionstate.ingestionTaskStatus == "INGESTING") delay 12 hours =>
  // ACQ-0820 (failure handling)
   (product_ingestionstate.ingestionTaskStatus == "DONE") and
  // Missing classification product Requirement
   (metadataInCatalog contains product_ingestionstate.productId) and
   // landCover not empty and
   // geographical names not empty and
   // localization not empty and
   // id, spacecraft, characteristics not empty
or
  // ACQ-0820 (failure handling)
 (product_ingestionstate.ingestionTaskStatus == "ERROR") and
not (metadataInCatalog contains product_ingestionstate.productId)
```

Fig. 5 PEPS acquisition system PBR second example

We obtained that product identifier shall be present in catalog database and characterized (land cover (water, forest, city, agriculture, etc. percentage); geographically localized (continent, country, region, city, etc.) instrument characteristics (instrument, processing level, product type, sensor mode, etc.), date, orbit number, satellite, resolution, snow cover, cloud cover if optical, etc.

This acquisition system goal is very important to characterize because it helps to check completeness and consistency of the peps images product catalog.

Note: For this first experimentation, only the presence in catalog was checked during the verification tests, assertions relative to product classification were human checked using Graphical User Interfaces -GUIs-. We can imagine checking it automatically in the future.

Then, the second step was to validate requirements formalization with its author to check that semantic interpretation was correct. The result is conclusive, only some minor points were reported. The author of the specification, who is a co-author of the present paper, agreed that this may help to improve our engineering processes even if it's easier for him to read the textual form of the PBR and not the predicates that are more adequate for computers

Then, those formalized requirements were presented to the contractor that will be responsible for the next releases of PEPS system from 1.2 until 1.4. Usually test design and specification task is done by the contractor that shall try to catch precisely requirements semantics and shall have some discussions with the author of the specification to clarify them. Once formalized, PBRs were presented to the contractor using PMM Designer. The result of this presentation was that the contractor agreed with the benefits of a more efficient mutual understanding of requirement semantics. They requested to use the PMM Designer as a possible support for this common understanding.

3.4 PEPS System Verification

For the verification tasks, it still remains to implement the test generally in a scripting language, even if we can imagine later some code generation starting from PMM models. PBR determination and validation facilitate test design task because what is to be checked is clearly defined.

We have translated PBRs into the scripting tests language (SQL requests and shell scripts). For the PBRs to be verified by humans (through GUIs), we translated them in human-readable statements. We can imagine automate some of those tests using other scripts in the future.

Technical qualification for PEPS system V1.1 was done successfully.

For the operational qualification, we had to show to human operators PEPS system goals using the application in release 1.1. We focused on expected outputs, observable states and system failures to be compensated by the operator. We observed that PBR's fits well with this operational process and we only had to show the real system's behavior when it is driven by all the PBR-based test scenarios and not the predicates and any requirements.

4 Conclusion: Lessons Learnt and Future Works

Benefits: the benefits observed in the experiment in 2014 are confirmed by this first application to the PEPS operational project:

- The requirements determination is goal-oriented, with as consequence, an improvement and a facilitation of requirements production,
- The quality and completeness of requirements are also significantly improved with a clear connection between expected outputs and the conditions of their actualization (observable states, failures, inputs).
- The efficiency of the validation process is also improved. Even inexact, a PBR remains unambiguous, measurable and testable. Its interpretation is done once and only once, preventing an undefined number of misinterpretations by the various stakeholders.
- Although it is still too early to claim it, the efficiency of the verification process should be significantly improved. When validated, PBRs are unambiguous (no possible misinterpretation) and testable, connected to their conditions of actualization as expected for test cases.

Limitations: Currently, the main limitations observed are related to the tool. PMM Designer is a preliminary mock-up of a modeling and simulation PMM tool. It provides the main editors requested by PMM. However, it does not provide a complete PBR editor, many operators are missing. It needs improvements to ease the modelling process. It does not interface either simulation back-end, thereby making

it impossible to PBRs validation and design verification by simulation, while this validation / verification by simulation is one of the strengths of the method.

Future works: It is planned to iterate the process described in this paper for the next PEPS releases from 1.2 until 1.4. We will have of course many other lessons learnt to report by the end of this year with those coming releases.

References

1. Newport, J.R.: Avionic Systems Design. CRC Press (1994)
2. Micouin, P.: Model Based Systems Engineering: Fundamentals and Methods. Wiley & ISTE (2014)
3. IEEE Standard 1362, System definition—concept of operations document (1998)
4. von Lamsweerde, A.: Requirements Engineering. Wiley (2009)
5. IEEE Standard VHDL Analog and Mixed-Signal Extensions, IEEE 1076-1, IEEE Computer Society (2007)
6. Modelica Association, Modelica®—A unified object-oriented language for systems modeling language specification version 3.3 (2012)