Model-Based Systems Engineering for Systems Simulation

Renan Leroux-Beaudout, Marc Pantel, Ileana Ober, Jean-Michel Bruel

Model-Based Systems Engineering for Systems Simulation

Renan Leroux[1,2,3], Marc Pantel[1,2(✉)], Ileana Ober[1,2], and Jean-Michel Bruel[1,2]

[1] Institute Technology (IRT) Antoine de Saint-Exupéry, Toulouse, France
[2] University of Toulouse/Institute for Research in Informatics of Toulouse, Toulouse, France
{renan.leroux,marc.pantel,ileana.ober,jean-michel.bruel}@irit.fr
[3] ALTRAN, Toulouse, France
Renan.Leroux@altran.com

Abstract. Model-Based Systems Engineering and early simulation based Validation & Verification are now key enablers for managing the complexity in the development of modern complex systems like Cyber-Physical Systems. Models provide a formal account of system requirements and design decisions. Model simulation enables both design exploration and design versus requirements correctness assessment. Model simulation activities rely on Simulation Systems (i.e. systems that execute the model simulation). System execution environment models play a key role during these activities. Appropriate models must be developed for each kind of analysis conducted during Validation & Verification. More and more often, complex Systems Engineering is conducted in Extended Enterprises and the simulation activities are performed using partial models that must be completed with mock-up models for missing parts of the system. The development of Simulation Systems is thus costly and error prone and would benefit from the same Systems Engineering principles that are applied to the product. We propose a methodology for a seamless integration of the Simulation Systems development in the Products Systems Engineering. This method imports the available elements from the models of the system and its environment, from the Systems Engineering for Product space to a dedicated Systems Engineering for Simulation space. The required mock-up models are then defined in the Systems Engineering for Simulation space. As a result, we target a better management and reuse of the various environment and mock-up models in the various simulation activities during the development of the same product. This proposal is independent both of the actual methods and tools used to model the system and of the simulation environment.

1 Introduction

The use of MBSE (Model-Based Systems Engineering) and early simulation-based V&V (Validation & Verification) offers effective means to handle the complexity of real-life industrial development projects. Regularly, such projects need to combine several engineering fields in the context of EE (Extended Enterprise)

https://doi.org/10.1007/978-3-030-03424-5_29

where many stakeholders are involved such as *Cyber Physical Systems* (CPS). In this context, the simulation activities are often performed in an ad-hoc manner depending on the project, the involved partners, etc. To our knowledge and understanding, there exists no common reference methodology helping the various engineers involved in the product development in making the best choices seamlessly and efficiently for the simulation activities. Our work aims at filling this gap by proposing a methodology that specifically addresses the simulation and its needs. This paper provides a first draft of this methodology illustrated through a realistic case study.

This paper first provides insights on approaches that achieve early simulation-based V&V in the context of MBSE in common industrial settings, in particular within the MOISE (MOdels and Information Sharing for System engineering in Extended enterprise) project of the IRT-SE (Institut de Recherche Technologique Saint Exupéry – Institute of Technology Saint Exupéry), where our work takes place. One key aspect is that these activities are nowadays mostly conducted in EEs where many stakeholders target an efficient cooperation while protecting their know-how (usually named wrongly IP (Intellectual Property) which is a legal term that may only cover partly the stakeholder purpose). Thus, the various parts of the systems models are built in a concurrent engineering manner and simulation activities are conducted on partial models that must be completed with mock-up models for missing parts of the system. These models must also be completed with environment models whose content depends on the kind of validation and verification activities that are conducted relying on simulation. The building of these *Simulation Systems* (SS) is thus, in itself, costly and error prone and would benefit from the same SE (Systems Engineering) principles that are applied to the product.

Our contribution advocates the use of a rigorous methodology to build SSs tailored for its specific needs. We introduce such a methodology, by adapting to SSs development, many principles specific to SE. In this context, a particular attention is given to the representation of the environment that plays a key role in the simulation. Particularly, for needs that are specific to simulation activities, its representation must be carefully handled and shall be included to a certain degree in the modelling. Our approach is generic and potentially compatible with various actual system development and simulation technologies.

The rest of this paper is organized as follows: in Sect. 2, we present existing efforts for simulation-based early V&V in the industry, in particular in the MOISE project, as well as our running example: the AIDA inspection drone that will be used throughout the rest of the paper. Section 3 overviews the principles of our generic methodology for performing simulation in the context of MBSE, which is detailed in Sect. 4 with illustrations from the AIDA case study. Section 5 presents the expected benefits for the use of a rigorous methodology for SSs development and gives some directions for future work.

2 Context Presentation

2.1 Industrial Concerns

MBSE and early V&V are now key enablers for the development of complex CPSs in many application domains like transportation [1–3]. Model simulation is an effective approach for early V&V, allowing design decisions to be assessed earlier in the product life cycle. The duration and costs of the system development can thereby be reduced [4–6].

According to [7], simulation-based design is a "*process in which simulation is the primary means of design evaluation and verification*". Given the increasing complexity of the systems and in order to manage the structural complexity of the systems simulations, [8] proposes a MBSE [9] method to integrate simulation activities in the development process of complex systems. This approach seems all the more relevant in the concurrent design of systems involving multiple engineering domains such as: mechanical, hydraulic, electrical, etc. parts. Depending on the point of view to be assessed, different simulation activities allow to estimate, and/or to refine the different interactions between components [7,8].

While aiming at time and development costs reductions, [2] focuses on issues related to the integration, verification, validation and qualification of the simulation models. To reduce the potential ambiguities between system engineers and domain experts in charge of simulation model development, [2] adds a new actor in the process called "*Model Architect*" that coordinates the various MBSE activities. This new actor should have a multidisciplinary vision of the product whose architecture is under design and some knowledge in simulation technologies used for modeling the various parts of the architecture.

In [2], the authors propose an ontological DSL (Domain Specific Language), MIC (Model Identity Card), that covers the needs of various application domains, including the specification of interfaces and the building of simulation models. This DSL targets all the actors of the models, unfortunately it does not integrate a precise account of the behavior of simulation models. To overcome this, [10,11] use the concept of MoI (Model of Intention), defined as a model based approach to request and specify model(s) or simulation(s) for a specific scenario.

The combination of MIC and of MoI proposed by [10,12] allows to fill the gap existing between the requirements for the simulation performed by the system architect and the implementation of all the required simulation models, therefore reducing the problems related to their integration.

The importance of the simulation for early V&V of requirements and design while protecting IP in EEs is highlighted by the presence of a standard dedicated to the implementation of such SSs - the FMI (Functional Mock-up Interface) standard [13]. The co-simulation part of this standard (FMI 2.0), allows to implement multi-simulations [14] or heterogeneous simulations [15] (see Fig. 1-a), while preserving the IP in simulation models [14]. This IP protection only allows the supplier to visualize the content of its own models and not the one of the others (see Fig. 1-b).

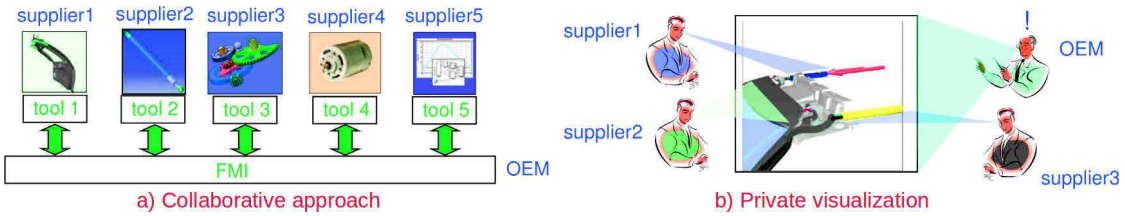

Fig. 1. Collaborative and private aspects of the FMI 2.0 standard (from [13]).

Furthermore, in order to build distributed simulations, [15] creates a bridge between FMI and the HLA (High-level architecture) standards [16,17]. In the same purpose, [18] extends the FMI 2.0 standard and supplies a tool (independent of HLA) to implement the simulations on distributed and multi-core architectures.

However, as far as we know, there is no common reference process offering a global and structured vision that allows to implement co-simulation platforms for systems models.

To address this, we advocate that the use of MBSE is meaningful not only for the development of the products, but also for the development of the SSs used during the development of the products. The current contribution illustrates how MBSE can be used for the development of the simulation tools and what are the expected associated models specific to the simulation.

The proposed method for the development of executable simulation models is generic with respect to product and SS development Methods and Tools. This is achieved by clearly separating concerns related to the (i) Systems Engineering for the Product under development, (ii) Systems Engineering for the Product Model Simulations, and (iii) simulation execution and results analysis. This separation can be expressed, for example, in this way: SEPS (Systems Engineering for Product Space) could rely on the MBSE method CESAM for developing the product, while SESS (Systems Engineering for Simulation Space) could rely on the ARCADIA method for developing the SS required for the simulation activities during the product development. Modelica [19], C, C++, Java languages, for example, could be used for implemented the various mock-ups, in the simulation execution space.

The core ideas, of this proposal, are that (i) each model simulation can be a project in itself, with its own constraints and costs; (ii) MBSE can also be applied to these simulation projects; and (iii) commonalities and building blocks can be reused for the different models involved in the development of the same product. Indeed, the development of a product using MBSE involves many different models that may be assessed using simulation. This results in a family of related, yet distinct, models that are conjointly developed and assessed as they are all involved in the development of the same product. Their assessment is done through various simulations which subsequently may require specific simulation models and tools. In this particular context, building blocks developed for the various simulation projects could be reused in the same manner as reuse occurs in SE.

2.2 The MOISE Project

The approach presented in this paper takes part in the MOISE project within the IRT-SE in Toulouse (France), with industrial partners, consulting companies, tools vendors and public research institutes (e.g., ISAE, IRIT, LAAS-CNRS, S/C ONERA).

MOISE develops a collaborative MBSE in EEs with the aim to both improve the development activities and reduce their costs. For this, in MOISE we consider requirement validation and design verification, for embedded systems, to enable seamless co-engineering between industrial partners and to manage requirements waterfall with agility and continuity. Furthermore, in MBSE, designers must ensure that the models that they have built are a correct expression of the design they had in mind. This is a specific kind of model validation that occurs each time a formal language is used to express human ideas. This is similar to requirement validation as the ideas a designer has in mind when he is building a model are similar to the informal requirements given by the user at the beginning of a project.

One of the key goals of the project is to reduce V&V costs by using early model simulation activities. The purpose of our proposal is to ease the development of SSs using MBSE and reduce the associated costs by improving reuse both in the transfer of models from product to simulation space [20] and in between the SSs built for the various models involved in the development of the same system.

2.3 The AIDA Inspection Drone Case Study

To illustrate our method, we apply it on a use case targeting an inspection drone that moves around a plane on the runway before take off (see Fig. 2). Its purpose is to support the pilot in the mandatory inspection of the aircraft before each flight. This drone should (i) quicken the pilot inspection task and (ii) improve its precision, by scrutinizing not-visually-accessible parts of the plane (e.g., the top of the wings, fuselage, ...), in order to detect irregularities, such as forgotten caps on sensors, ill closed trap doors, or mechanical defects such as thunderclaps or impacts of hail.

This drone should be manually controlled following predefined paths (drone flight plans), with enhanced automated safety capacities to avoid hurting ground staff. For this purpose, the drone is aware of the cartography of the plane and of the location of the points of interest to be scrutinized. The drone is equipped with various sensors: vision system, GPS locator, and a radar, for a greater precision, to ensure a sufficient safe distance with respect to the plane and the ground staff.

To enable the diagnostic in case of malfunction, the flight data are saved locally and transferred in real-time to the ground. Moreover, the operator can watch live images taken by the drone, to make sure that control points do not present any irregularities, and adapt the drone flight plan if needed.

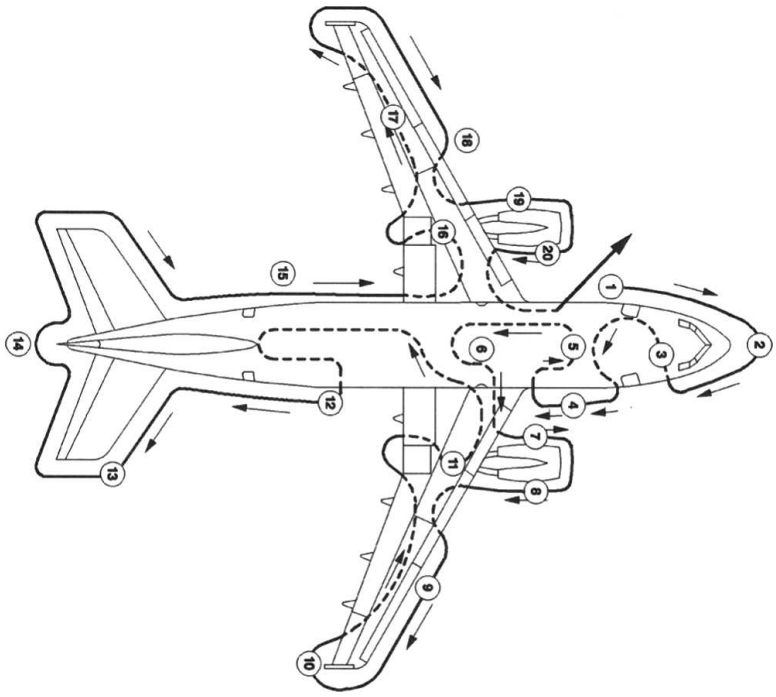

Fig. 2. External Walk around a plane.

3 The V Cycle and Simulation Activities

3.1 Overview

The main purpose of our work is to ease the simulation-based early V&V activities in MBSE. Let's recall the distinction between Validation and Verification: According to Boehm, Validation targets building the right product (i.e. the product that fits the user needs – the implicit requirements that are the source for writing the product specification) and Verification targets building the product right (i.e. the product that satisfies its explicit requirements – the product specification). Thus, if the product is correct (i.e. has been verified), Validation mainly targets detecting an erroneous translation from the implicit requirements in the user mind to the explicit requirements – the product specification. This analysis must involve the user. A first assessment may be conducted without the user in the loop: checking that the explicit requirements are complete and consistent. When the development process involves several phases that each have explicit requirements and expected results, these definitions needs to be adapted to be applied to each phase: Phase Verification assesses the compliance of the results with respect to the requirements; and Phase Validation assesses the requirements (completeness and consistency). When all phases have been conducted, Verification is complete whereas final Validation activities are still needed for the user to accept the product. The IEEE Standard Glossary of Software Engineering Terminology states that Verification is "*The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase*" and that Validation is

"*The process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements*". This user part of the validation can be extended to any human activities conducted in a development that require translating the implicit human ideas to explicit documents. In MBSE, these documents are expressed as models. Thus, Model Validation assesses that the model is a correct rendering of the ideas the developer had in mind, whereas Model Verification checks that the resulting model satisfies the explicit requirements, that are most of the time also expressed as models resulting from the previous phases. This can even be extended to documents that have no formal semantics (e.g. natural language, drawing, etc.) used as requirements. Formal Model Validation assesses that a formal model is a correct rendering of an informal document, whereas Formal Model Verification assesses that a formal model satisfies explicit requirements available as formal models.

One of the oldest and most common life-cycle model for system development processes is the *V-Model*. This is a theoretical model that was never applied as it was defined. It consists of sequential process phases, where each phase must be completed before the next one begins. The *V-Model* is rather common in the manufacturing industry such as aviation, automobile, and many others where there exists usually three main phases: product, systems and equipments. A product combines several systems that integrates various equipments. Each phase consists of sequential process steps defined in the development method used. For example, ARCADIA experimented in MOISE relies on the Requirement, Function, Logical and Physical steps.

On the theoretical side, which has almost never been applied as is, the left side of the V corresponds to the development of the product and the right side of the V corresponds to the various V&V activities.

Before the introduction of MBSE, the left hand side of the *V-Model* roughly corresponded to the product development with very little efficient V&V activities conducted. Indeed, only proofreading could be conducted as the results of each steps where semi-formal graphical or textual documents targeting human readers. The right hand side of the model covered test-based V&V of the equipments; then equipments integration and test-based V&V of the systems; systems integration and, in the end, test-based V&V of the product.

Its use could raise problems of diverse natures as most of the efficient V&V activities were only conducted quite late when the various equipments needed for a product had been implemented. It often resulted in the late discovery of Requirement, Functional, Logical or Physical issues during the integration of the various validated and verified equipments. For minor issues, minor changes in the requirements, design and implementation may be possible, while staying efficient and cost effective. However, in some situations, issues discovered during late V&V activities will require major architecture changes or local patches to circumvent and prevent the problem. In all cases, this leads to increased costs and delays in delivery, additional maintenance difficulties, and potentially the addition of new weaknesses.

Furthermore, architecture exploration suffered from the same issues. Either, it was conducted early in the development by humans based on document reading. It was usually far from optimal for complex systems. Or, it was conducted later using tests based on the implemented equipments. But, this led to very high costs as all the equipments required by all the variants of the architecture had to be developed and all the variants of the architecture also had to be implemented.

Last, design document proofreading is not well adapted to IP protection in EE. Indeed, in order to conduct a V&V activity using proofreading, the designer needs to have access to all documents related to the system he designed including the ones built in other enterprises.

To prevent this, the introduction of MBSE and early V&V was shown to be of precious help and is currently being deployed in most manufacturing industries. MBSE enforces the writing of formal models in each step of each phases instead of semi-formal documents, and the assessment of these models using simulation-based testing. These assessments allow the early validation of requirements and verification of design steps. Furthermore, it allows validating the models written by the designers to assess that they are a correct rendering of the ideas he had in mind. The early V&V can take various forms, such as model exploration and structural analysis. That would allow to check for instance that there is no isolated communication port and that the direction of communication paths is unambiguous, thus detecting issues in the architecture that would only be detected in the integration V&V activities.

The actual interpretation of the V cycle depends on the abstraction level at which we consider the system. Figure 3 provides the product, system and equipment views that correspond to the main engineering phases.

At the highest level of abstraction – the one covered by the upper left part of the *V-Model* – the customer needs are expressed and coarse models of the environment of the future product are required to validate the expression of these needs. An exploratory phase is usually conducted to assess the appropriate use of new technologies (see Fig. 3, phase 1) with respect to previous similar products.

The exploratory phase must respond to questions like: (i) what kind of material should be used for its physical parts: steel, aluminum, carbon, composite fiber? (ii) what is the worse case of winds the drone will be submitted to? (iii) can the AIDA drone be protected from radio or EMC interference? To answer these questions, material models or environment models (atmospheric, radio, EMC (Electromagnetic compatibility)) should be simulated with more or less precise description of known and already identified interactions with the system to be studied.

This phase handles the expression of customer needs. For example, in AIDA (see Sect. 2.3), the drone shall conduct an inspection around the plane to detect irregularities.

In this exploratory phase, simulation can be used to illustrate high-level behaviour, under the form of textual requirements using customer vocabulary, or sequence, activity, or state diagrams: operational scenarios that will be executed in front of the user that can accept or not the simulated behaviour. For instance, the procedure of drone intervention around the plane needs to interact with

the pilot, the meteorological data provider, and eventually the control tower. These diagrams allow to precisely define the order of interaction with different stakeholders, and simulation can be used to validate it.

In this exploratory phase, the environment models must have the appropriate accuracy to assess that models, within this phase, reflects really its intended semantics and behavior. The environment models, in the simulation, could be later refined to better capture reality.

The second phase of the *V-Model* (see Fig. 3, phase 2) is the *system phase* that takes into account higher-level requirements stemming from the previous phase in order to express operational requirements associated with the various systems to be designed in order to build the final product. This is an essential part, that represents the core of our work, although our proposal is generic enough to be applied to the other phases. Models and activities involved in this phase will be provided in the following sections of this contribution.

The *equipment* phase (see Fig. 3, phase 3) focuses on the underlying hardware platform and the associated deployed software. It is developed on the basis of the requirements produced at the system phase (phase 2). In the context of simulation, at this stage we target particularly accurate simulations. That could cover the simulation of a processor whose behaviour is described at the clock cycle level of accuracy, the simulation of a communication protocol taking into account the physical layers of the OSI standard, etc.

This last kind of simulation is not addressed in this paper, but our proposal could be easily adapted to handle such constraints, usually involving HIL (Hardware In the Loop).

3.2 MBSE-RFLP Method

The MBSE approach, used for the AIDA use case, relies on the RFLP (Requirement, Functional, Logical, Physical) general methodology that drives many industrial methods and tools, like the ARCADIA methodology [21] and the associated CAPELLA toolset. With this toolset, during the development of the

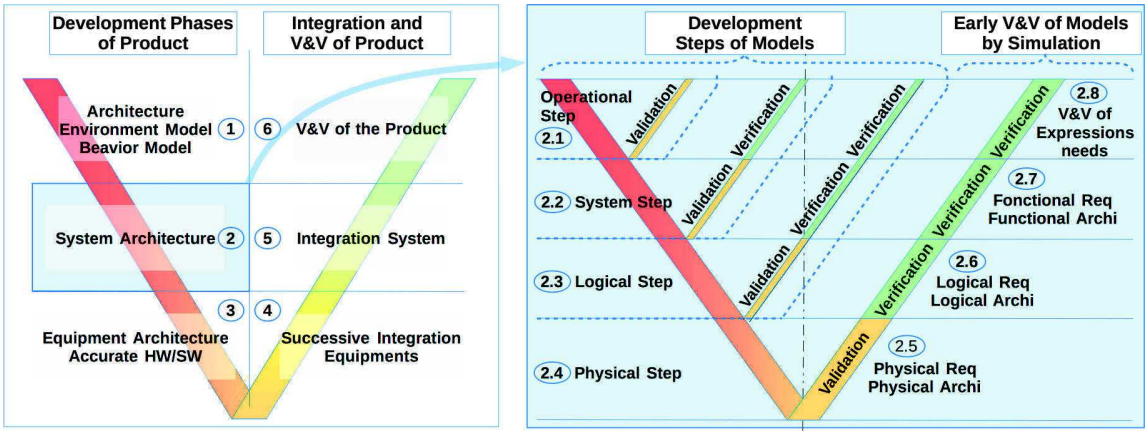

Fig. 3. Global V cycle and focus on system development layer.

system models, it is possible to build several kind of architecture corresponding to each layer (see Fig. 3, phases 2.1 to 2.4). These models, in each layer, can be assessed through simulation, both to check that models reflect the system designer intention (validation) and that models satisfy the requirements expressed in the previous layer (verification).

The first step of the System phase, Operational Analysis, analyzes the operational requirements issued from the Product phase and builds corresponding models that will drive the following steps of the System phase. These models can be validated using simulation.

To meet the operational requirements expressed in these first models, the next step is the "System Architecture design" that mostly consists in refining models from the previous phase. The obtained architecture shall meet the requirements of the operational layer, which can be assessed through V&V. Newly defined functions and their communications interfaces (see Fig. 3, phase 2.2) adds new requirements to be met by the next steps.

When this functional architecture is complete and mature enough, it may be significant to group similar elements into common and specific functions, providing logical components (as it is the case with the *Allocation functions* from Figs. 8, 9 and 10). This results in an intermediate architecture layer (the *Logical layer* in Fig. 3, phase 2.3) situated between the functional layer expressed above and the physical layer to which these logical functions will be allocated. This logical layer can ease the deployment and the assignment of components (more precisely, their inner functions) to the equipments in the physical layer.

Finally, the *physical layer* specifies the physical architecture of the equipment, as well as deployment and inter-dependency links. Additional requirements are added to this layer to specify/constrain component deployment, communication means and interfaces between them (see Fig. 3, phase 2.4).

The principles of the MBSE method used previously are quite similar to other approaches like CESAM [22]. Therefore, the MOISE approach can be adapted to other methods.

4 Proposed Approach

Building the required tools for a specific model simulation activity can be expensive. It is submitted to temporal constraints related to the development process and associated steps. In a common industrial frame, this kind of simulation project involves numerous specialists and may require the building of a specific simulation platform with a significant computing power.

This contribution advocates that it is possible and meaningful: (i) to apply system engineering principles to Simulation Systems (see Fig. 4), (ii) to handle the models for simulation as autonomous objects, and (iii) to take into account separately its support of execution, including the computing power and its spacial distribution, within the various stakeholders in the EE, that participate in the development of the product whose models must be simulated (see Fig. 6).

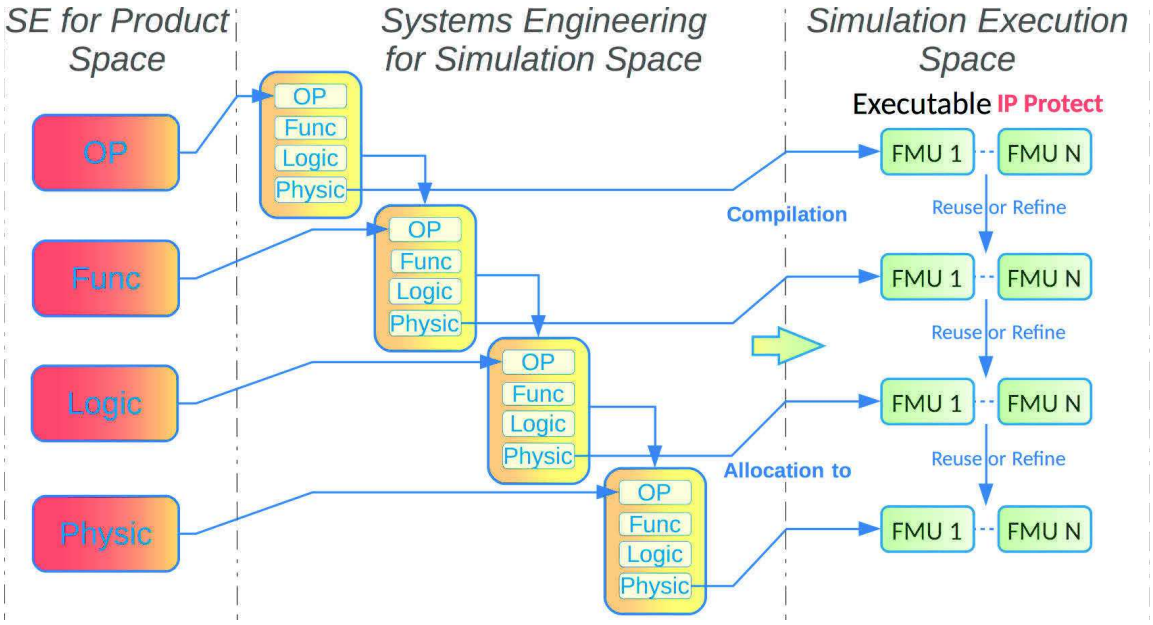

Fig. 4. Global View of the methodology, without processing platform allocation.

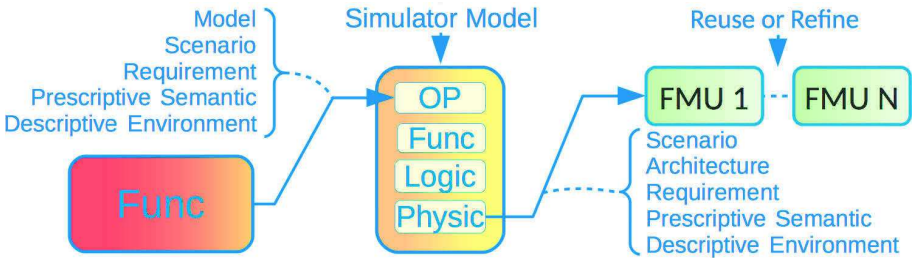

Fig. 5. Data transfer from ES product space to ES simulation space. (Color figure online)

4.1 Model-Based Systems Engineering for Simulation

MBSE helps the system engineer in assessing the relevance of the system architecture and the compliance with the desired system properties. The same results can be expected from its application to Simulation Systems. Our proposal has the particularity that both methodologies (e.g., CESAM [22], ARCADIA [21], ...) are independent: the one used for the development of the simulation system can be different from the one used for the design of the product.

The clear separation between the Systems Engineering for the Product and the Systems Engineering for the Simulation offers numerous advantages. The main benefit of our approach is that it offers flexibility and adaptability. Diligent to the (cultural) context of the companies involved, it allows to comply to the methods and the working habits of the involved people and facilitates the collaboration of the various actors involved in the development and simulation.

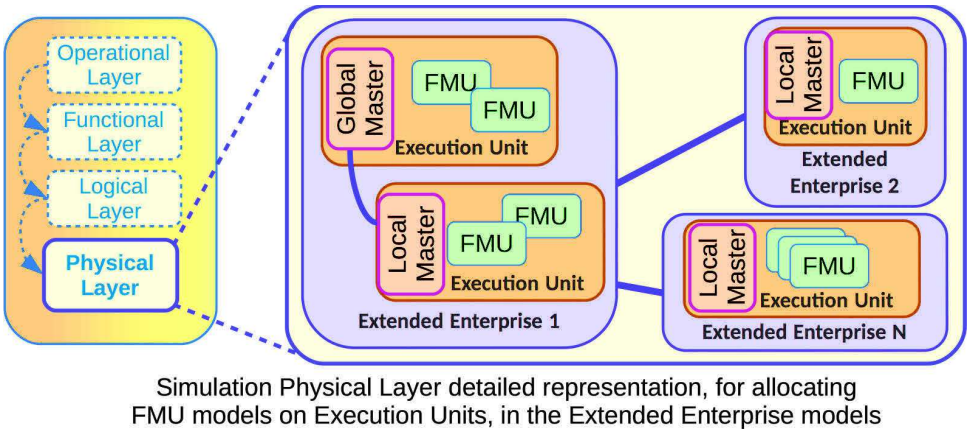

Fig. 6. Content of the simulation physical layer.

4.2 Systems Engineering Product Space

The SEP (Systems Engineering for Product) space is the entry point of our method. It relies on models to represent the various aspects involved in the system design phase of the product. To illustrate our point, we use the MBSE-RFLP design method, as explained in Subsect. 3.2. The system engineer in charge of the product plays the role of the *System Architect*. By acting on the four layers, he will be able to use simulation, throughout the design cycle to assess that the obtained models correctly capture his purpose, and are conforming to the models from the previous phases: operational architecture for the requirements, functional architecture of the provided services, components in the logical architecture and allocation of these logical components in the physical architecture.

4.3 Simulator Systems Engineering Space

Each simulation project for each simulation V&V activity in the SEP space will be developed in the SES (Systems Engineering for Simulation) space. The system engineer in charge of the simulation project is called the SA (Simulation Architect). SA has the choice of the most appropriate method of conception. To illustrate more easily our proposal, we have chosen the same design method as the one from the SEP: the MBSE-RFLP method. It is also in this space that the SA specifies the inputs/outputs of simulation models and shares them for execution on the simulation platform of each stakeholder (see Fig. 6).

4.4 Models Simulation Execution Space

This space is dedicated to the management, integration and execution of executable simulation models. These activities can be facilitated by the use of co-simulation standards like FMI. Due to space limitations, we do not address the specifics of this space in this contribution.

4.5 Simulation Architect

The SA should have some particular skills, such as: (a) have a wide knowledge of the various domains being simulated, and (b) be open minded and endowed with communication skills [2]. Indeed, typically the simulation involves different actors that are each expert in their field of activity. The SA is the interface between the System Architect and these experts involved in the implementation of the simulation models. The SA performs in the SES space and the models simulation execution space and thus will also communicate with people in charge of the infrastructure of the simulation, in the various companies. Let us mention, for example, the people in charge of the implementation of computers and OS, of the security of the internal and external networks, of the management of physical access rights to the hardware and software.

4.6 Simulation: From Its Request to Its Execution

The description of the process proposal is generic, regardless of the system design method used, and of the selected layer in the SEP space: Operational (OP), Functional (Fun), Logical (Logic) or Physical (Physic) as illustrated in Fig. 4: Product Space.

The starting point is the System Architect needs to assess properties of the product's architecture. Therefore, he transmits all the information needed to the SES space. Figure 5 illustrates the information flow between the functional layer of the product and the operational layer of the SES space.

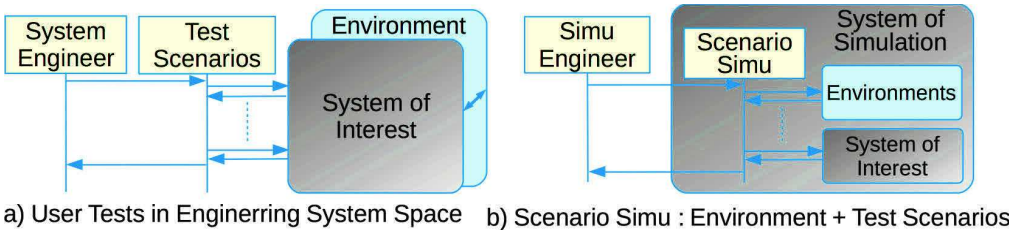

Fig. 7. Evolution between Product Space & Simulation Space.

Our approach is illustrated using the AIDA drone use case introduced in Sect. 2.3. With respect to Fig. 7-a, the drone is the *System of Interest* whose architecture is currently being designed and must be assessed using simulation. The System Architect is currently assessing the behavior of the "Move" function (see Fig. 8, Product Model) present in the System of Interest at the "Func" layer of the SEP space (see "Func" layer in Figs. 4 and 5). The"Move" function shall control correctly the position and the speed of the drone.

To perform a simulation, the System Architect provides some information to the SA:

- *the system architecture model*. With respect to our case study, it includes the "Move" function and all the directly related functions ("Compute Actual Position and Speed", "Inertial Central", "Compute Next Position and Speed", "Receiver GPS")
- *test scenarios models* describing the interactions between the "System Engineer" actor and the System of Interest during the simulation
- *requirements for traceability*
- a *prescriptive semantic model* of the function to be simulated. In our case, this corresponds to a model of the execution semantics of the "Move" function, as an UML activity or state diagram, or as a Modelica or Simulink model, or even as software code, etc.
- *environment models requirements*. In our case (see Fig. 8-Product Model), the System Engineer asks to include in the simulation, the "Provide Signal from GPS Environment" and "Create Lift force from Atmosphere Environment" models.

The System Architect transfers all these information to the SES space, as one can see in Figs. 4 and 5).

The SA handles these data as requirements, and places them in the dedicated operational layer for simulation (see Fig. 5).

From these elements, he begins to build the simulation system functional architecture whose purpose is the simulation of the "Move" function (see Fig. 5: OP → Func). This layer imports functions from the "Product Model", provides functions to describe environments and the scenario for simulation.

The SA models the internal environment (turquoise blue) as a family of functions, currently under development, directly or indirectly connected to the function of interest. In this "Internal Environment", the output and input of functions, directly connected to function of interest, describes the expected behaviour of the function of interest and are thus considered as correct by construction. For the AIDA use case, functions placed in "Internal Environment" are "Inertial Central", Compute Actual Position & Speed".

For instance, the "Compute Next Position & Speed" function is not part of the "Internal environment" because, this function has already been designed and the associated validated and verified models are already available.

The external environment model describes the environment of the System of Interest at an appropriate level of detail to ensure the expected quality of the analysis. These models can be reused, with eventual refinements, from previous simulation (see Fig. 5).

In the SEP space, the GPS satellite sends signals to the GPS receiver. However, for simulation purposes, we do not need to provide details of the relations between GPS signal and the GPS receiver. Thus, in our example, the path "Provide signal" and "Receiver GPS" from SEP are modeled in SES space by the simpler "Position & Speed" function of the "GPS Environment".

The System of Interest of the simulation is the "Move" function which is identified in dotted red line in Fig. 8. All the other simulated functions are drawn with full red or green line.

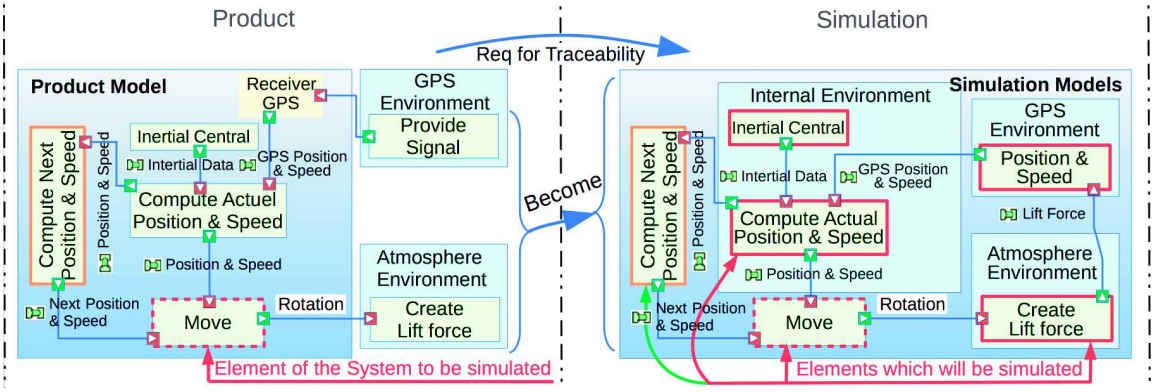

Fig. 8. From product SE Space to Simulator SE. (Color figure online)

When the successive refinements of the functions reach a precise enough description in functional layer for the assessment under way, these functions (see Fig. 9) are assigned to the components of the logical layer (see Fig. 10).

However, before actually assigning them to a logical layer, the refinement of Fig. 9 can be interpreted in two ways: (i) It may be a refinement provided by the system architect. In this case, the SA must rely on this refinement instead of the container "Move" function. If needed, the System Architect must provide the Prescriptive Semantic Model for the refined function (the intended behaviour), (ii) It may also be a Model of Intention represented by the functions: "Regulation Pos&Speed", "Compute Motor Speed", "Regulation i^{th} Motor Speed" and "Create Rotation i^{th} Motor". The SA keeps the "Move" function unchanged, retrieves the Prescriptive Semantic Model from the System Architect, and forwards it to the experts in the Simulation Execution space.

In our case study, we consider that it is a simple refinement. The logical layer groups related functions as "logical components". The relation used for the grouping depends on the purpose of the model designer. For example, in

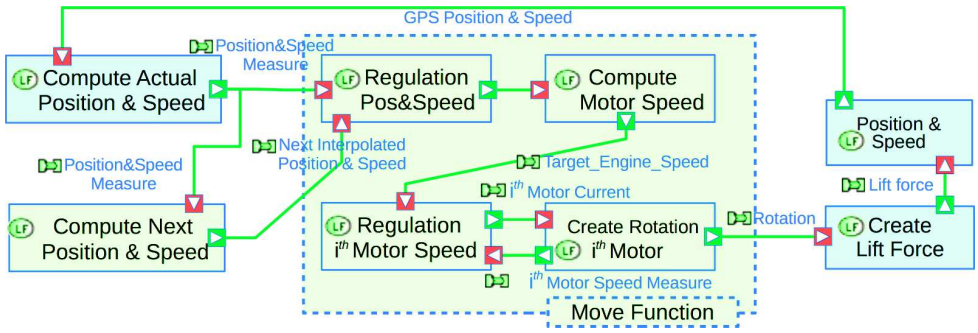

Fig. 9. Detail of "Move" function, in simulation SE Space. Env Functions in Turquoise Blue. (Color figure online)

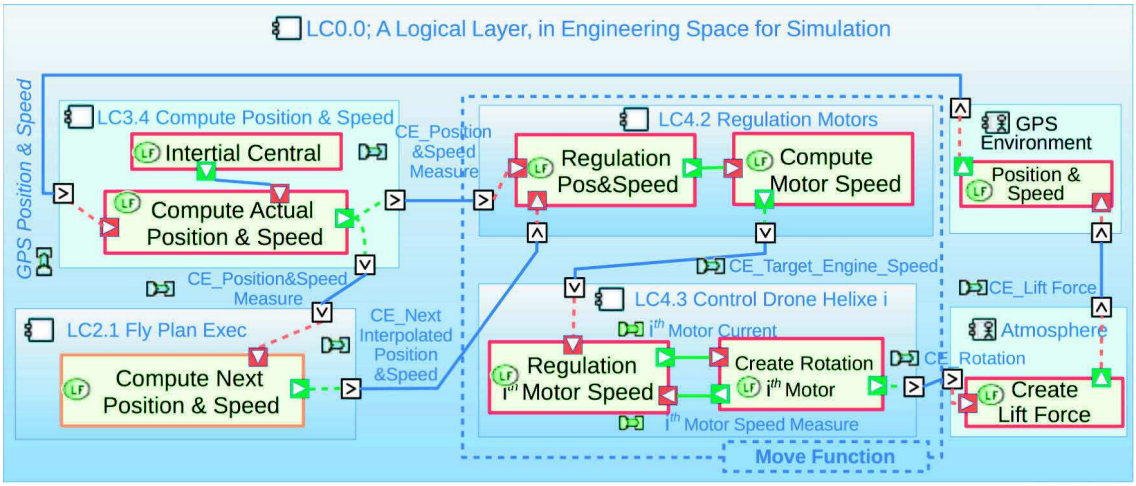

Fig. 10. Allocation of functions to logical components. Internal&External Env in Turquoise Blue. (Color figure online)

the SES, the notion of component is preserved, to which is added the notion of business domain (mechanical, electrical, etc.). The component becomes a logical and specialized container for a particular business domain.

In the Fig. 10, we can see functions grouped in logical components from our example: Functions "Inertial Central" and "Compute Actual Position & Speed" are grouped in the logical component "LC3.4 Compute Position & Speed", "Compute Next Position & Speed" in "LC2.1 Fly Plan Exec(ution)" for example.

The physical layer (see Fig. 6) allows expressing the EEs elements: the entities (companies), the physical execution supports (Execution Unit, down to possibly, Processors and associated Threads), the communication means (Networks), and the simulation models. In this layer, the SA will deploy the inner functions of Logical Components using, for example, FMU (Functional Mock-up Unit) on the execution platform. These FMU are placed in companies, on their simulation platforms. As evoked in the state of the art, FMUs have a standardized communication interface, and the model comes in the form of an executable binary.

At this point, the SA engineer has taken into account the model from the space of SE, set up the necessary environments of simulation, refined the functions, created Logical Components, defined the hardware structure of simulation and assigned inner functions of Logical Components to Physical Components FMU-type on Execution Units, located in companies (see Fig. 11 for the AIDA use case simulation of the "Move" function.). Red lines correspond to Communication Links between Execution Units of different Extended Enterprises. The Global Master algorithm manages exchange data between Local Master, placed on Execution Units of each Enterprise. The Local Master drives the execution of its attached FMU.

It remains for the SA to transform scenarios of the requested tests, as simulation scenarios that will drive the execution. The purpose is to integrate the environments of simulation, to specify the order of execution, in order to provide

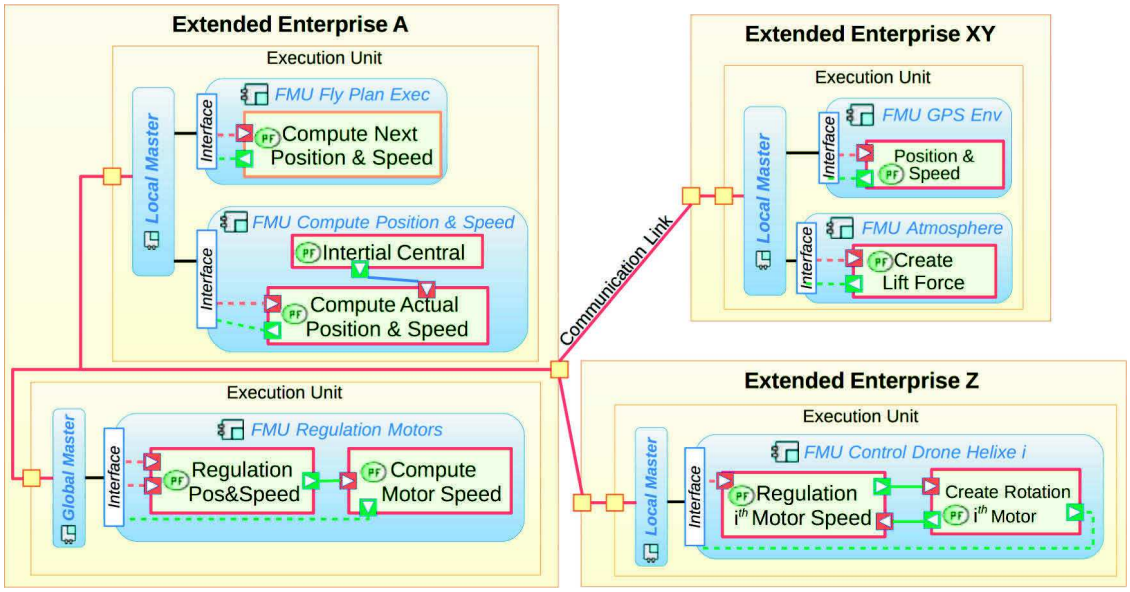

Fig. 11. Allocation of inner functions of LComponents to FMU, in Physical Layer.

the necessary data for the "Move" functions of the product model (see Fig. 4). For instance, a requirement for the environment could be: "a wind of Northwest sector and speed of 5 knots", or "a fog with visibility of 20 m".

This diversity of approaches leaves to the experts the choice of the most appropriate model, still respecting the inputs/outputs of the FMI interface.

Besides the aforementioned files, he will forward to each company involved in the co-simulation, the data regarding the configuration of the appropriate algorithm (global/local master) that are provided in the physical layer and the simulation scenario. These data provide useful information to enable companies to schedule their own simulations. These local algorithms executed in each company must allow to send and receive all the intermediate data of the simulation toward the algorithm controlling the simulation from a global point of view (global master). At the end, the SA distributes the appropriate resulting simulation data towards the concerned companies.

5 Conclusion and Future Work

This contribution proposes an approach to improve the integration of early simulation in the model based systems engineering development life-cycle in the context of extended enterprises. The proposed approach is generic, thus independent of any particular system development methods or tools. We illustrate and discuss our approach using the AIDA drone case-study for plane inspection.

The major benefit of this approach is to put the simulation at the core of the development process, by carefully defining its own development steps and positioning it against the overall product development. Moreover, our approach contributes to a better organization of the development, by making explicit links between simulation models and other models built during development.

Also, there is a significant reuse potential within the models dedicated to the simulation architecture. As a result the V&V activities can become more efficient and explore a large spectrum of situations for the same cost.

The benefits and potential improvements in the current development process introduced by our proposal come in different forms: with respect to the organization of the work, to architecture exploration and simulation of the product, as well as in terms of re-use of the simulation architecture models.

The major interest of this approach is to ensure the independence between various categories of models: models issued from the system engineering development, models issued from the SE of the simulation, and models created for the simulation execution itself. Additionally, information on the traceability between these models is explicitly stored. Our proposal does not enforce any particular methods or tools for system development. By defining adequate model transformations between the various modeling tools, the stakeholders can use the SE tools and methods that best fit their needs. The same analysis stands for the company responsible for the design and the integration of the simulation. This approach also allows to clearly separate the responsibilities between the design of the product and the design of the simulations in a "*don't be both judge and jury*" spirit (independence between specification, implementation and verification usually required by certification bodies).

Another advantage of our proposal is to be able to successively reuse (partial) models and results of the previous simulations for similar systems. That would be the case for instance for models of functions, components, hardware equipment, or simulation architecture. Similarly, it is possible to reuse or refine executable mock-ups or FMU. This approach facilitates the set up of exploratory simulations for architectures, and allows to conduct partial simulations of the product that may be later refined.

The clear separation between the product and simulation spaces will provide a high flexibility in the construction of the simulation platform: knowing partners interested in the design, it is possible to model and then create gradually, the simulation platform, inside and between partners, in accordance with the available budget.

Environment models are key elements for simulation. We intend to explore and to specify attributes needed to give a precise account of their nature.

Acknowledgements. The authors would like to thank the MOISE project members for their contributions as well as the IRT-SE and the French *Commissariat Général à l'Investissements* and the *Agence Nationale de la Recherche* for their financial support in the frame of the *Programme d'Investissement d'Avenir*.

References

1. Leroux, R., Ober, I., Pantel, M., Bruel, J.-M.: Modeling co-simulation: a first experiment. In: Proceedings of MODELS-2017, Satellite Event: Workshops, September 2017. http://ceur-ws.org/Vol-2019/gemoc 3.pdf
2. Sirin, G., Paredis, C., Yannou, B., Coatanéa, E., Landel, E.: A model identity card to support simulation model development process in a collaborative multidisciplinary design environment. IEEE Syst. J. **9**(4), 1–12 (2015)
3. Capasso, C., Hammadi, M., Patalano, S., Renaud, R., Veneri, O.: A multi-domain modelling and verification procedure within MBSE approach to design propulsion systems for road electric vehicles. In: Mechanics & Industry, vol. 18, p. 107. AFM, EDP Sciences 2016, January 2017. www.mechanics-industry.org
4. Fiorese, S.: Découvrir et comprendre l'Ingénierie Système. AFIS - Cepaduès, March 2012
5. BKCASE, SEBoK: guide to the engineering body of knowledge. http://sebokwiki.org/wiki/Guide_to_the_Systems_Engineering_Body_of_Knowledge_(SEBoK)
6. Topper, J.S., Horner, N.C.: Model-based systems engineering in support of complex systems development. Johns Hopkins APL Tech. Digest **32**(1), 419–432 (2013)
7. Shephard, M.S., Beall, M.W., O'Bara, R.M., Webster, B.E.: Toward simulation-based design. Finite Elem. Anal. Des. **40**(12), 1575–1598 (2004). https://doi.org/10.1016/j.finel.2003.11.004
8. Graignic, P., Vosgien, T., Jankovic, M., Tuloup, V., Berquet, J., Troussier, N.: Complex system simulation: proposition of a MBSE framework for design-analysis integration. Procedia Comput. Sci. **16**, 59–68 (2013)
9. INCOSE, Systems Engineering Handbook, A Guide for System Life Cycle Processes and Activities, 4th edn. Wiley, New York (2015). https://sepnobrasil.yolasite.com/resources/INCOSESystemsEngineeringHandbook4e2015.pdf
10. Retho, F., Smaoui, H., Vannier, J.-C., Dessante, P.: Model of intention: a concept to support models building in a complex system design project. In: ERTS, pp. 115–126 (2014)
11. Retho, F., Smaoui, H., Vannier, J.-C., Dessante, P.: A model-based method to support complex system design via systems interactions analysis. In: Proceedings of the Posters Workshop at CSD&M (2013)
12. Sirin, G., Retho, F., Yannou, B., Callot, M., Dessante, P., Landel, E.: Multidisciplinary simulation model development: early inconsistency detection during the design stage. In: Advances in Engineering Software (2017). https://hal.archives-ouvertes.fr/hal-01673538
13. FMI: Functional Mockup Interface. http://fmi-standard.org/
14. Galtier, V., Vialle, S., Dad, C., Tavella, J.-P., Lam-Yee-Mui, J.-P., Plessis, G.: FMI-based distributed multi-simulation with DACCOSIM. In: Proceedings of the Symposium on Theory of Modeling and Simulation: DEVS Integrative Symposium, ser. DEVS 2015, San Diego, CA, USA: Society for Computer Simulation International 2015, pp. 39–46 (2015). http://dl.acm.org/citation.cfm?id=2872965.2872971
15. Neema, H., et al.: Model-based integration platform for FMI co-simulation and heterogeneous simulations of cyber-physical systems. In: Proceedings of the 10th International Modelica Conference, 10–12 March 2014, Lund, Sweden, vol. 96. Linköping University Electronic Press; Linköpings universitet, pp. 235–245 (2014)
16. Dahmann, J.S., Fujimoto, R.M., Weatherly, R.M.: The department of defense high level architecture. In: Winter Simulation Conference (1997)

17. Dahmann, J.S., Fujimoto, R.M., Weatherly, R.M.: The DoDHigh Level Architecture: an update. In: Winter Simulation Conference (1998)
18. DACCOSIM: Distributed architecture for controlled co-simulation. https://sourcesup.renater.fr/daccosim/index.html
19. The Modelica Association: Modelica. https://www.modelica.org/
20. Bossa, B., Boulbene, B., Dubé, S., Pantel, M.: Towards a co-simulation based model assessment process for system architecture. In: Proceedings of the 2nd Workshop on the Formal CoSimulation of Cyber Physical Systems, Satellite Event of the Software Engineering and Formal Methods Conference (2018)
21. Voirin, J.-L.: Model-Based System and Architecture Engineering with the Arcadia Method. ISTE Press - Elsevier, London (2017)
22. CESAM-Community, CESAM: Cesames systems architecting method, January 2017. http://cesam.community/wp-content/uploads/2017/09/CESAM-guide_-_V12092017.pdf