Simulation architecture definition for complex systems design: A tooled methodology

Jean-Patrick Brunet, Henri Sohier, Mouadh Yagoubi, Mathieu Bisquay, Pascal Lamothe, Pascal Menegazzi

Simulation architecture definition for complex systems design: A tooled methodology

Jean-Patrick Brunet, Henri Sohier, Mouadh Yagoubi, Mathieu Bisquay, Pascal Lamothe, Pascal Menegazzi

Abstract
For the design of complex systems like in the automotive industry, the use of Model Based Systems Engineering (MBSE) is being considered as a promising solution in order to formalize and communicate information. Numerical simulation is also routinely used as a tool to answer potential design questions that arise. However the link between MBSE and simulation still needs further improvement. In this work, a tooled methodology is proposed in order to enhance the link between system architecture and numerical simulation. In a first step, a solicitation package is formalized and implemented in a SysML-based tool to define the simulation needs. In a second step, a tool that allows to define the simulation architecture and to pilot the execution of the simulation is developed. We show that thanks to the proposed process and exchange format between the system and simulation architects, model reuse and agility is improved in a complex systems design.

1 Introduction

1.1 Context

The research presented in this paper was conducted within the AMC project (Agility and Design Margin) which is taking place in the research institute IRT SystemX. One of the main objectives of this project is to propose a tooled methodology for simulation based complex system design. We focus on design questions that can be answered by numerical simulation as this later has been considered as an efficient tool for decision-making in the design of complex

---

Jean-Patrick Brunet; Henri Sohier ; Mouadh Yagoubi ; Mathieu Bisquay
IRT SystemX
8 avenue de la Vauve, 91120 Palaiseau, France
{firstname.lastname}@irt-systemx.fr

Pascal Lamothe
PSA Groupe,
pascal.lamothe@mpsa.com

Pascal Menegazzi
Valeo,
pascal.menegazzi@valeo.com

systems. The simulation is considered as a complex system that may deviate from the original system it represents. This deviation can be explained by the need to account for different specifications that are specific to the simulation. Such specifications might be the implementation of physical phenomena or balance between accuracy and speed for example. The complex task of linking system and simulation architecture led to the emergence of a fairly new role, the simulation architect (1). The simulation architect has a key role in the proposed methodology as he interprets the needs from the system architect in order to specify a suitable simulation that would provide the required results for decision. In a standard V-Model (Fig. 1) multiple needs can be expressed during the design phase and the simulation architect will need to answer each of those within the context of the system.

The global process begins with the definition of high level stakeholders needs. Then, the system architect defines the system architecture by developing SysML diagrams at the operational, functional and physical levels. Following the need in terms of simulation, the system architect formulates his solicitation to the simulation architect based on a formalism that we have proposed in a previous work (2). This solicitation package would contain the system information as well as required initial QCD (Quality Cost Delay) requirements for the simulation. Once received, the simulation architect constructs the elements of the simulation architecture progressively, using the new methodology we propose in this paper. If needed, he will interact with the system architect to alter the initial requirements to better match the simulation possibility and system needs. The resulting simulation is then integrated in a simulation framework connected to a decision support tool. This serves a double purpose, analyzing and post-treating the results of the simulation with a close collaboration between the system and simulation architects and if needed, generating new results with minor modifications of the specification (eg: minor change of actuation distance etc…). Once the collected data are sufficient, the system architect can take the final decision and the process end.

In the AMC project different industrial partners (PSA group, Renault, Valeo, Airbus) and software providers (Sherpa engineering, Digital Product Simulation, Siemens) are involved in defining and improving the proposed process. For this, a practical design problem is used: The early design of an autonomous vehicle passing traffic light.

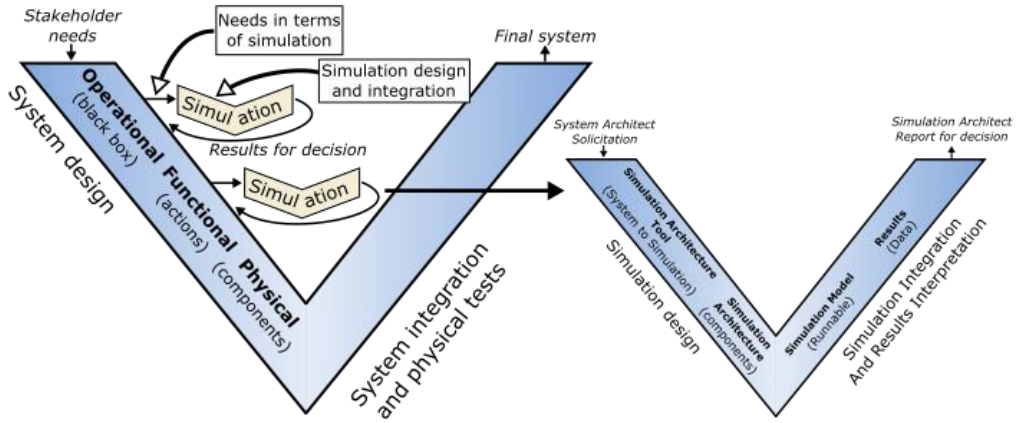

Fig. 1. Details of the simulation based decision cycle in a system V-model.

1.2 Industrial problem

One of the two industrial problems considered in the project AMC is the design of an autonomous vehicle which can pass traffic lights. Such industrial problem is important as it allows the different partners of the project to share a common language, to identify the current needs in terms of methods and tools, and to validate new solutions. The considered vehicle has a sensor which provides the color and the distance to a traffic light within a certain range subject to uncertainties. The vehicle then follows a control algorithm to adapt the speed to the traffic light and anticipate possible color changes. The speed is reduced when approaching the traffic light to avoid brutal or even unreachable decelerations. Furthermore, when the color of the traffic light changes to orange, the vehicle stops if it cannot pass the traffic light within one second. In the use case considered in this paper, the system architect defined four parameters: two for the characterization of the sensor and two for the control. Setting those parameters impacts two important decision criteria: the cost of the sensor and the total energy consumption of the vehicle. The system architect formulates the design question as a multi-objective optimization problem and transfers the solicitation package to the simulation architect, who will design the simulation architecture, perform said simulation(s) and provide the results to the system architect in order to take a decision regarding the design of the vehicle.

2 Agility in complex conception cycle

As mentioned in the previous section, the system architect will regularly request simulation to obtain sufficient data in order to take decision. In some cases the link between system engineering and simulation is very direct, with no important change between system and simulation architectures. In the STIMULUS tool (3) formal textual requirement and state diagrams can be tested using random variables, and CIL4Sys applications (4) allow testing of formal sequences and state diagrams through on the shelf or simple plant simulation models and user defined environments. However, complex simulation models are often required for the representation of physical systems with both controls laws and physical phenomena. It usually, then, becomes more difficult to mix system and simulation as they would need to involve different peoples and topologies. MBSE being considered a semi-formal or even informal language, automation and communication between the different stakeholders is complex (5).

Currently the information needed to be transferred from the system to the simulation actors are ill defined with no specific and readily available format, as SysML in the case of system engineering. It is commonplace to use a document-based process (emails or written documents) to provide those information leading to loss of traceability and lack of standardization. We presented a new tooled methodology in a previous article (2), which aims at enhancing the link between system architects and simulation experts by formalizing simulation needs by exploiting the SysML diagrams of the system architecture. Starting from this formalization of the simulation needs by the system architect, a new tooled methodology is proposed in this paper, extending our previous work, enabling the definition of the simulation. The aim of our tooled methodology is to formalize the exchanges between system and simulation architects in order to limit the loss of information, leverage past simulations, formalize requests for new models and give the system architect access to the right data for a supported decision.

This methodology is presented in figure 2. The solicitation as described in section 3 is sent to the simulation architect who can then assemble a simulation architecture based on specifications indicated for each system function. In order to do so, he can use past simulations associated to system specifications close to the one requested in the solicitation (with the help of computerized inferences) or, if no existing model match the specification he can generate a new model through the use of the MIC (Model Identity Card) standard (6–8). The generated MIC will define all necessary requirement in order for a model provider to create a matching model.

Once the simulation architect validates a simulation architecture corresponding to the solicitation, he will generate and link the corresponding simulation to a simulation framework. This framework will allow to generate results for a decision support tool within the specified scope of the model. The system architect, in collaboration with the simulation architect, will then be able to analyze the results and either take a final decision or use the knowledge gained to

alter the initial solicitation and repeat the whole process. This allows for fast agile loops within the global V-cycle of the system design process. Agility is also improved on the simulation side, where the simulation is developed iteratively in function of the changing needs from the system side.
For each step of said process, we proposed a tooled methodology: one dedicated to the System Architect (described in section 3), another for the Simulation Architect (described in section 4).
We believe that through the proposed methodology, true agile cycles can be implemented within a MBSE based design cycle by furnishing a formal framework for exchanges between the stakeholders (system architect, simulation architect, and model providers). Allowing for faster exchange of information, it improves information based decision making and more generally accelerates the cycles.

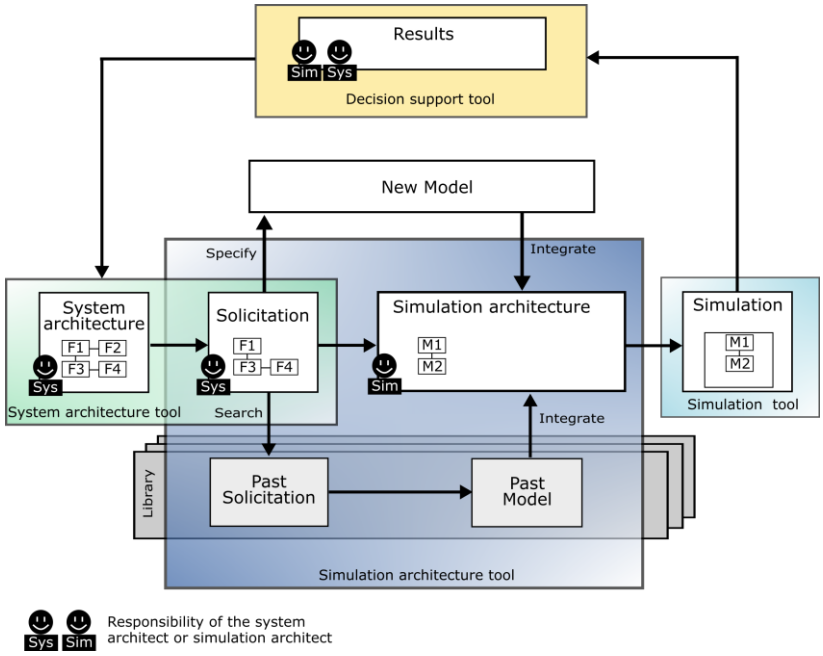

Fig. 2. Proposed methodology for the simulation definition.

3 Solicitation package from the system architect

3.1 Current practice and alternative

The development of a system, considered here to be supported by a model of the system architecture, can raise various questions requiring simulation. Such

question is sometimes manually summarized in a textual document, whether it is a Word document or an email. This textual document occasionally includes diagrams illustrating the system to simulate. It is often challenging to have a complete and clear question. Furthermore, textual documents make it much harder to identify questions which have already been answered and which were similar, preventing from efficiently re-using past simulations. Textual documents finally make it hard to validate that the system is correctly represented in the simulation, and that the simulation results correctly answer question.

Instead of this documental approach disconnected from any model of the system architecture, the process described in this paper first uses the concept of solicitation from the system architect. This concept was introduced in (2). The solicitation is a documented question of the system architect, directly based on the model of the system architecture. Indeed, the model of the system architecture precisely aims at sharing information about the design of the system with different types of people, from mechanical designers to electrical designers, and it is also perfectly suited for a collaboration with simulation exerts.

3.2 Content of the solicitation package

The system architect can use the system architecture to communicate three types of information in his solicitation: the part of the system related to the question and which needs to be simulated, the question itself, as well as the environment scenarios to be tested. The part of the system related to the question is important as the question is generally not about the full system. For example, when the question is about the ability of an autonomous vehicle to pass traffic lights, the air conditioning system may typically be ignored. When describing the part of the system related to his question, the system architect should choose the appropriate level of detail. For example, in the case of an autonomous vehicle passing traffic lights, he may choose not to take into account the internal components and the exact way the sensor works to output the color and distance of a traffic light.

The question asked by the system architect in his solicitation can be, amongst others, a request for validation. For instance, a system architect may want to check that the average energy consumption of a car is below a standard reference. The question can also be a request for optimization. A system architect may want to find the best design parameters to minimize one or more performance criteria, like the average energy consumption or the production cost. Whatever the question is, its formulation can be supported by the model of the system architecture. The model includes for example the requirements to be tested in a validation, or the unset parameters whose values are to be explored in an optimization. Furthermore, the system architect may also want to monitor specific properties of the system,

like its speed or its instantaneous consumption, which he can also select in the model of the system architecture.

Finally, the environment scenarios are also an important part of the solicitation. The environment scenarios generally start as a rough description of the environment and its evolution during the test. In the case of an autonomous vehicle, it can first be decided to have a straight road with a traffic light at 500m. In the process described here, this rough description is part of the solicitation of the system architect. The environment scenario is then progressively refined when the simulation is developed, up to the exact properties of the road for example. The environment scenario can be defined thanks to the model of the system architecture which characterizes the system's environment, including roads and traffic lights. However, while the model shows that there can be a traffic light on the side of the road, the system architect must specify in his solicitation that the test should include a traffic light at 500m.

Thus, the formulation of the solicitation is facilitated by the model of the system architecture as it includes various necessary information. Furthermore, the solicitation is not only generated as a textual document, but also as a model. While numerical continuity is generally lost with a textual document, the model of the solicitation has multiple logical links with the model of the system architecture it comes from.

3.3 Implementation of the solicitation package

Using the model of the system architecture to formulate the solicitation requires new software functions. Indeed, it relies on the processing of numerous and complex relationships between the data of the model, from state diagrams to sequence diagrams and bloc diagrams. Some of these software functions have already been prototyped as a plugin in an existing SysML editor called PhiSystem, provided by Sherpa Engineering and based on Papyrus. Fig. 3 shows the menu added to the top bar. The menu first makes it possible to copy the system architecture, which is a practical solution to associate the solicitation to a certain version and configuration of the system. The menu also offers a quick access to diagrams as well as summaries of information at the different levels of the system architecture (operational, functional, and physical) in order to select the perimeter of the question and the necessary level of detail. The menu finally allows to ask a question based on the requirements of the system, and to hide the part of the system which is not related to the question.

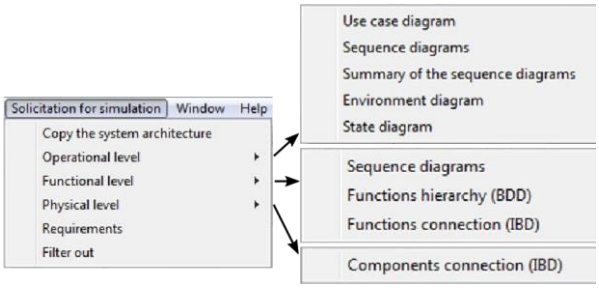

Fig. 3. Menus added to the top bar in PhiSystem.

For example, if the objective of a system architect is to optimize the control and sensor properties of an autonomous car to minimize both the cost and the energy consumption when passing traffic lights, he can first go to "Use case diagram" in the top-bar menu of Fig. 3 and select the use case "To pass traffic light". The implemented software functions automatically find the related data and highlight the perimeter of the question in the menu "Functions connection (IBD - Internal Block Diagram)", as shown in Fig. 4. The level of detail can be set in the menu "Functions hierarchy (BDD- Block Definition Diagram)".

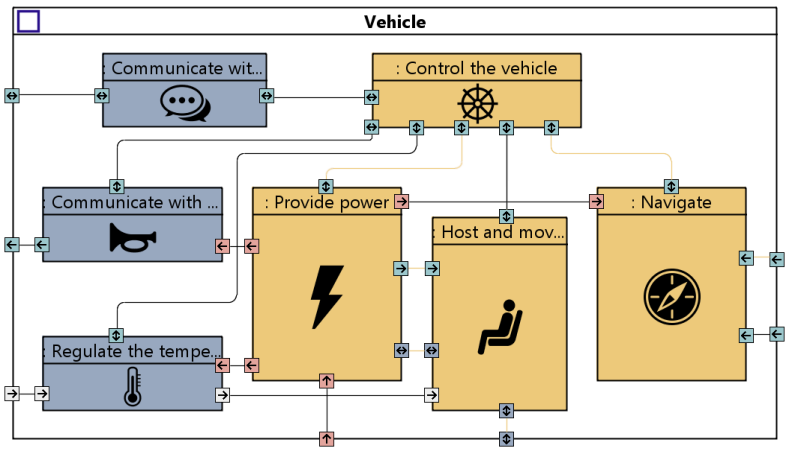

Fig. 4. Perimeter of the system related to the solicitation.

4 Proposed methodology for the simulation architecture definition

The solicitation defined in the previous section allows to carry the simulation needs (quality, cost and delivery) from the system to the simulation. The simulation architect will then need to define the simulation architecture. This would entail finding existing models, pre/post-processing functions, specify

missing models or functions and ensure that the final architecture correctly represents the system and responds to the needs expressed in the solicitation. The simulation architect needs to convert the assembly of system functions into a coherent assembly of models and software functions.
In order to keep a consistent approach to those simulation blocks, it is chosen to describe them through the properties of the "Model Identity Card" (MIC) which was initially presented in (9). The MIC aims at fully specifying a simulation model independently of its execution, as such it can be used to identify an existing simulation model (useful in a reuse case) or to specify requirement for a provider. This approach allows to create consistent architectures mixing existing, reworked, or specifically created models each connected through the ports properties defined in their MIC. The entire workflow of simulation can be validated through the ports properties.
Once assembled the final simulation generated from this simulation architecture will itself be described as a MIC model. The MIC format allows to recursively describe all the internal models of the simulation architecture keeping a traceability of its construction. It is important to note that, at the moment, while a MIC model contains an indication of its validity scope through the indication of range for each variable, it does not contain indication on its accuracy. This might be added in future evolution of the format.
The following sections aim at describing the tools put into place to assist the simulation architect in creating, validating, executing and reporting the simulation architecture that correspond to a specific solicitation.

4.1 Developed components

a. Solicitation package import module

This tool aims at eliminating the uncertainty in information and traceability incurred in document or email request. The solicitation generated by the system architect is sent, unaltered to the simulation architect tool which will then extract information on the simulation needs, scope and granularity as well as start the traceability of the associated simulation architecture.

b. Models library

Existing simulation models are stored in a library that link both the MIC and the system function (or functions) matching the given model. This aims at facilitating the reuse of previous simulation work by offering visibility of existing models as well as proposing inference type associations to link system function in the new architecture to semantically close ones from previous work. Fig. 5 represents the process of creating a new simulation architecture using the proposed tool. A representation of the filtered system architecture is present in the top with the

simulation architecture situated below it. On the right, the models library is displayed. In this figure, the simulation architect already started the creation of the simulation architecture with two system functions (represented in green in the architecture) already associated to some simulation models. Based on the needs, some simulation models have also been added with no direct association to the system functions. At the step shown in the figure, the simulation architect has selected a system function (in red) that he wants to associate to a model. The models library tool (panel on the right of the figure), based on inferences at the system level, proposes a previously used simulation model called "ElectricVehicle" that could be used to simulate this function. The simulation architect selected it (reason for which the simulation model has a green outlay) and the system presented the other system blocs that would also be fulfilled by this model (in orange).
In the case a system function would not have a suitable existing model, the simulation architect can choose to create a specification MIC that would be added to the simulation architecture. Upon having an associated model with that MIC it would be added to the models library for future reuse.

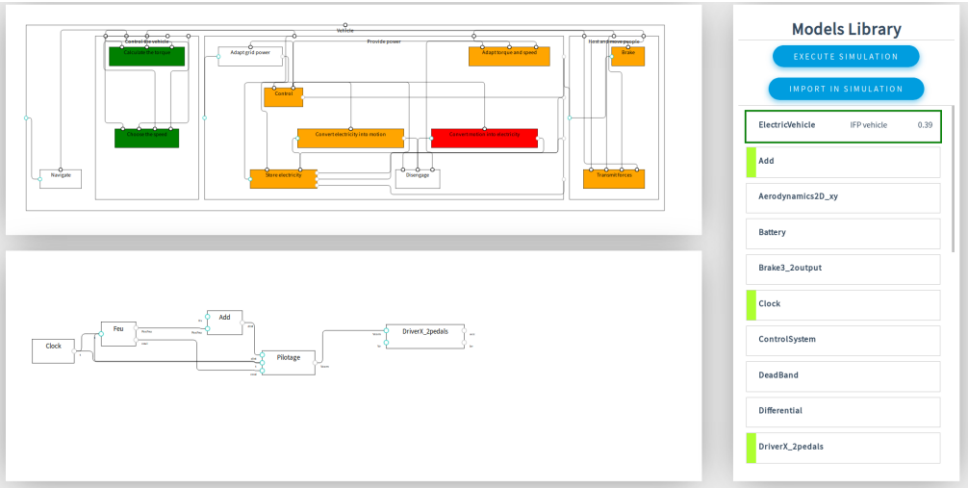

Fig. 5. Screenshot of the tool during the simulation architecture definition process.

c. Simulation Architecture Verification and creation tool

As presented in Fig. 5 the simulation architecture tool allows the simulation architect to progressively define the simulation architecture while maintaining consistency with the system architecture. While new models are added to the simulation architecture, the tool uses the MIC attributes to check the consistency of the model. This is insured through proposing automated links between similar ports (matching dimensions, units and directions), thus verifying that linked models possess compatible ports. Once the whole architecture is verified by the

tool and correspond to the specification made by the simulation architect, it can be sent to the model transformation module. Currently, the validation of the simulation performance (QCD) is insured by the simulation architect. In future iteration, it could be imagined that said validation would be at least partially automated through this module (precision within range, validity of the output …)

d. Model Transformation Module

Once the simulation architecture is finalized, the MICs it includes correspond to known models in the models library. This allows this tool to use the formal description of the simulation architecture to generate a valid simulation by assembling the models.
As the MIC also contains the type of environment on which the model needs to be executed, this module also allows to prepare the environment for the execution of the simulation.

e. Computer aided Decision Making

Upon creating the simulation, this module enables the simulation architect to specify the input parameters and variables and run it accordingly. The results are then displayed according to the needs of visualization extracted from the solicitation and the users' specifications. The results are automatically extracted and displayed. The tool also enables running the simulation using different inputs, post processing the results and highlight the advantages of each architecture through direct visual comparison.
The system architect (with the eventual expertise of the simulation architect) can compare the results to the needs defined in the solicitation. He then can either take a final decision upon the given solicitation or use the knowledge gained to alter the initial solicitation and restart an iteration of the process.

5 Conclusion

This work further refines the role and tools of the simulation architect. It presents a methodology where the system and simulation architects have distinct, complementary roles within the conception cycle. The presented methodology simplifies and formalizes exchanges between the system architecture and the simulation in complex systems design. Through the use of the solicitation package, the system architect is able to communicate a clear, formal and exhaustive request to the simulation architect. This remove doubts or inconsistencies that can arise from improperly defined demand. From this solicitation package and through the developed tool, the simulation architect is capable to define, validate, run and analyze a simulation model that is tailored to answer the requirement set by the system architect.

This methodology also simplify the reuse of prior simulation model by using formal description of those (MIC) and association with prior system function. By presenting those prior model in a standardized and centralized library associated with computer generated suggestion, the workload associated with reuse and validation of prior model can be greatly reduced.
Lastly, this proposed methodology also facilitate agile project management during the conception cycle. By allowing centralized, traceable, standardized communication between the shareholders of the project as well as facilitating reuse and simplifying validation of new simulations against the solicitation package, we ensure faster simulation iteration during the conception cycle.
As a future work, we aim to explore the impact of this novel methodology on the whole conception cycle, by considering state-of-the-art agile methods.

References

1. Retho F. Collaborative methodology for virtual product building to support aerial vehicles with electrical propulsion design. 2015.
2. Sohier H, Guermazi S, Yagoubi M, Lamothe P, Maddaloni A, Menegazzi P, et al. A Tooled Methodology for the System Architect's Needs in Simulation with Autonomous Driving Application. In: Syscon 2019. 2019. p. 735–42.
3. argosim. Requirements Simulation with STIMULUS - ARGOSIM [Internet]. [cited 2019 May 24]. Available from: https://www.argosim.com/home/stimulus-for-requirements/
4. CIL4Sys Engineering – Home [Internet]. [cited 2019 May 24]. Available from: http://cil4sys.com/
5. Rauzy AB, Haskins C. Foundations for model-based systems engineering and model-based safety assessment. Syst Eng [Internet]. 2019 Mar 1 [cited 2019 May 24];22(2):146–55. Available from: http://doi.wiley.com/10.1002/sys.21469
6. Sirin G, Paredis CJJ, Yannou B, Coatanea E, Landel E. A Model Identity Card to Support Simulation Model Development Process in a Collaborative Multidisciplinary Design Environment. IEEE Syst J [Internet]. 2015 Dec [cited 2019 May 24];9(4):1151–62. Available from: http://ieeexplore.ieee.org/document/7004782/
7. Sirin G, Retho F, Yannou B, Callot M, Dessante P, Landel E. Multidisciplinary Simulation Model Development: Early inconsistency detection during the design stage. Advances in Engineering Software. 2017.
8. Fontaine G. Modélisation théorique et processus associés pour Architectes Modèle dans un environnement multidisciplinaire. 2017.
9. Sirin G. Supporting multidisciplinary vehicle modeling towards an ontology-based knowledge sharing in collaborative model based systems engineering environment [Internet]. 2015. Available from: http://www.theses.fr/2015ECAP0024/document