ENSTA IP PARIS	SIM Simulation et Ingénierie Multidisciplinaire	**Livrable L1.1.0**

Livrable		
Tâche 1		**Date : 04/04/2025**
L1.1.0 Niveau de confiance dans la décision de l'architecte en liaison avec la qualité des simulations		
Responsable	ENSTA	
Participants	ENSTA	

Contributeurs :	Nom	Organisation	Rôle
Responsable du livrable	Thomas Rigaut	ENSTA	Ingénieur de recherche
Responsable scientifique	Omar Hammami	ENSTA	Professeur
Co-auteurs	Thomas Rigaut	ENSTA	Ingénieur de recherche
	Thomas Vita	ENSTA	Ingénieur stagiaire
Resp. revue interne	Omar Hammami	ENSTA	

Historique du document:

Version	Date	Modifications	Etat[1]	Distribution
V0.1	04/04/2025		Ebauche	
V1	11/04/2025			
V2	17/04/2025			

[1] Etat = "Ebauche"; "En revue"; "Validé".

Abstract

The Model Architect is the actor of the design process of complex systems who assesses the system architectures defined by the System Architect, using simulation. His aim is to build the appropriate model regarding the constraints of the simulation demand.

In this deliverable, the Model Architect Problem (MAP) is defined as scheduling Model Architect tasks in order to answer the simulation demand. It uses a multi-criteria scheduling description using a three fields notation (resources, constraints, objectives). Several variants of the MAP are proposed.

However, the definition of this scheduling problem introduces new concepts (like simulation quality, the notions of risk, uncertainty, utility), model quality and model composability are defined and links with the MAP are specified.

Several mechanisms used to partially automate the Modelling and Simulation process will be introduced in this deliverable.

Finally, the study carried out on a Renault case study concerning the optimization of the on-board electric network of a mild hybrid vehicle is described.

Table des matières

Liste des figures

Liste des tables

Aucune entrée de table d'illustration n'a été trouvée.

Liste des acronymes et abréviations utilisés

Abbreviation	Definition
CBSE	Component based systems engineering
CCO	Cost-Constrained Optimization
CSE	Computational Science and Engineering
DM	Decision Maker
MA	Model Architect
MAP	Model Architect Problem
MAUT	Multi-Attribute Utility Theory
MCDA	Multi-Criteria Decision Aid
MDCM	Multi-Criteria Decision Making
M&S	Modelling and Simulation
RCO	Reliability-Constrained Optimization
RD	Robust Design
RID	Risk Informed Design
SA	Simulation Architect
SIM	Simulation et Ingénierie Multidisciplinaire
V&V	Validation & Verification

1 Introduction

The Model Architect is a new role that has been defined within the SIM project. The Model Architect is a stakeholder of the conception cycle of complex systems who configures the architecture of models, builds the workflow of generation and assembly of models, specifies models for the model providers, integrates them in the simulation and then simulates the system of interest.

In the first section of this document, the Model Architect Problem (MAP) is introduced and defined in terms of multi-criteria scheduling. This problem formalizes the workflow of tasks performed by this stakeholder when a multi-physics simulation demand is performed. A trade-off between the quality of the simulation, the resources used and the time required to get the simulation results is proposed because those criteria can be conflictual in some cases. That induces the creation of a Pareto Front.

Then, the next section concerns the notion of model quality, which must be defined and formalized in order to solve the Model Architect Problem. A state of the art is initially presented, and the links with the notions of risk and uncertainty are described. The notion of utility is also introduced because the aim of the simulations performed is to make decisions. That is why the decision-making purpose and the notions induced like utility are dealt with. The concept of composability of simulation models is also introduced because the simulation of complex models composed of several sub-models is considered. Then, those notions are used to reinterpret the MAP with model quality explained.

The next section will introduce and develop four mechanisms to partially automate the Modelling and Simulation process.

The last section exposes the work done on the Renault case study (on-board electrical network of a mild hybrid vehicle) whose description can be found in deliverable L4.2. Several simulation models for the same components are developed to treat the composability problem, and an alternative formulation of the optimization problem is proposed.

2 Multi-criteria scheduling and the Model Architect Problem (MAP)

The Model Architect is the actor who has to assess the system architecture defined by the System Architect. And for that purpose, he uses simulation. In fact, his aim is to build the appropriate model regarding the constraints of the simulation demand.

The Model Architect Problem consists in scheduling the whole workflow of the simulation process of a complex system, from the simulation demand by the System Architect (given an architecture of the system) to the return of the simulation result. This includes collecting the models of the sub-systems, assembling them and then simulating them. However, several conflicting criteria must be optimized: the time of the process, the resources used and the quality of the simulation.

The first subsection will introduce multi-criteria scheduling theory, its aims and a notation that is commonly used in the literature to characterize classes of scheduling problems. The second subsection deals with several multi-criteria scheduling formulations of the Model Architect Problem.

2.1 Introduction to multi-criteria scheduling

According to (Pinedo, 1995), "scheduling concerns the allocation of limited resources to tasks over time. It is a decision-making process that has as a goal the optimization of one or more objectives". Hence, because scheduling concerns generic terms such as tasks and allocation, this domain is effectively used in various fields of application, from computer operating systems to project management.

In order to define a scheduling problem, we are given:
- n jobs (J_i is the job number i);
- J_i is composed of p_i operations ($O_{i,j}$ is the j^{th} operation of J_i);
- m machines (M_k is machine k). The machines are also called the resources.

When solving a scheduling problem, the aim is to assign the jobs J_i to the machines M_k while respecting some constraints and optimizing given criteria. In order to describe the configuration, constraints and objectives of a scheduling problem, a notation consisting of three fields α, β, γ, has been developed in the literature. Each field is a list of words separated with commas. Hence, a specific $\alpha|\beta|\gamma$ characterizes a class of problems. The meaning of each field is the following:
- α defines the environment in terms of resources (machines);
- β lists the constraints of the problem;
- γ specifies the optimization criteria.

In order to ease the comprehension, we present in the following a simple scheduling problem:

$$1|prec|C_{max}$$

The first field $\alpha = 1$ means that there is only one machine in the environment. The second field $\beta = prec$ means that there exists a precedence relation between the jobs that will be performed. And finally, the objective is described by $\gamma = C_{max}$. In a schedule, the completion time of job i is C_i. And C_{max} is defined by $C_{max} = \max_i C_i$. It is the maximal completion time over all jobs for a schedule. The objective in γ is to minimize C_{max}.

The three fields notation allows to differentiate between different classes of problems. It groups problems from different domains of application, using their commonalities from a scheduling viewpoint. Thereby, this notation allows to search in the literature for a solution or a property (for example, the computational complexity) of a problem using its class.

Then, γ can represent a single criterion as well as multiple criteria. The latter case corresponds to multi-criteria scheduling which is a branch of scheduling theory that deals with multi-objective problems. Hence, the aim is to optimize several conflicting criteria, and to provide Pareto optimal schedules. To be more understandable, concepts concerning multi-objective optimization must be defined. First, a multi-objective optimization problem is an optimization problem with multiple objective functions. Let be X the feasible set of decision vectors, and $Y = f(X)$ the criteria space of finite dimension. Therefore, $\exists K \in \mathbb{N} \mid Y \subset \mathbb{R}^K$. The multi-objective optimization problem is defined by:

$$\min_{x \in X} f(x)$$
$$s.t. [g_1(x), g_2(x), \cdots, g_l(x)] \leq 0$$
Tapez une équation ici.
$$with\ f: X \to Y$$
$$x \mapsto [f_1(x), f_2(x), \cdots, f_K(x)]^\intercal$$
Tapez une équation ici.
$$and\ \forall\ i \in [1, l], g_i: X \to \mathbb{R}$$
$$x \mapsto g_i(x)$$

With this formulation, f is the objective function to be minimized, and the l g_i functions define the constraints of the problem. The term $[g_1(x), g_2(x), \cdots, g_l(x)] \leq 0$ means that $\forall\, i \in [1, l], g_i(x) \leq 0$. Then, optimality in the multi-dimensional criteria space must be defined. A partial preorder can be defined in the criteria space. $\forall y, z \in \mathbb{R}^K$, we have:

$$y \leq z \Leftrightarrow \forall\, i \in [1, K], y_i \leq z_i$$
$$y = z \Leftrightarrow \forall\, i \in [1, K], y_i = z_i$$

The definition of optimality that is used is the one of Pareto optimality. A list of definitions concerning this notion is therefore provided:
Definition of a weak Pareto optimum:

$$x \in X \textit{ is a weak Pareto optimum} \Leftrightarrow \nexists\, y \in X \mid \forall\, i \in [1, K], Z_i(y) < Z_i(x)$$

The set of weak Pareto optima is noted WE.

Definition of a strict Pareto optimum:

$$x \in X \textit{ is a strict Pareto optimum} \Leftrightarrow \nexists\, y \in X \mid \forall\, i \in [1, K], Z_i(y) \leq Z_i(x)$$
$$\textit{with at least one strict inequality}$$

The set of strict Pareto optima is noted E. We have $E \subseteq WE$.

Optimality in our optimization problem is defined in the sense of Pareto optimality. A Pareto optimal solution is a set of elements whose any objective cannot be improved without degrading another objective. The goal is to approximate the set of Pareto optimal solutions which is also called the Pareto front.

The final notion to be introduced in this section concerns robustness considerations in scheduling theory. In fact, real-world problems are characterized by uncertainties on the parameters (for example, processing times). Therefore, a schedule that is valid and optimized in theory can become useless and infeasible in practice. In order to overcome this problem, a trade-off between the optimality and the robustness of the schedule regarding perturbations must be established. Two approaches to solve that are described in (T'kindt, 2006):
- the proactive approach. Some knowledge on the uncertainties of the problem is taken into account in order to search more robust solutions.
- the reactive approach. The schedule is modified in real time as unexpected perturbations arise.

In practice, a mean to add robustness to a schedule is to use the notion of flexibility. It eases the modification of a schedule. Three types of flexibilities exist according to (T'kindt, 2006): the sequencing flexibility, the temporal flexibility and the assignment flexibility.

Those concepts relating to multi criteria scheduling and robustness will be used to define the Model Architect Problem in the next section.

2.2 Formulation of the Model Architect Problem (MAP)

In this section, two formulations of Model Architect Problem using multi criteria scheduling are proposed. They differ in the way they consider the availability of sub models.

2.2.1 Model Architect Problem $P1$

The first definition of the MAP assumes that the whole set of models can be found in a "Model Factory". Then, the Model Architect Problem $P1$ can be expressed with the following form:

$$R\,m \mid sp-graph, unavail_j \mid \#(C_{max}, Q, Resources)$$

We have the resources $\alpha = R\,m$. This means that we are given m unrelated parallel machines. The machines are computers, engineers and models. Thus, they are heterogeneous.

The constraints of the problem are $\beta = sp-graph, unavail_j$. The first one, $sp-graph$, means that there are precedence constraints (between the tasks) forming a series-parallel graph. The second one, $unavail_j$, denotes that machines (resources) can be unavailable during periods that are known in advance.

Finally, the optimization criteria are$\gamma = \#(C_{max}, Q, Resources)$. This means that the goal is to reach the Pareto front of the following optimization criteria:
- C_{max} which is the total time of the whole simulation process;
- Q which is the quality of the simulation;
- $Resources$ which represents the resources effectively used during the schedule.

2.2.2 Model Architect Problem $P2$

For this second definition, let's assume that some of the models are unavailable. Thus, the design and the collection of those models belong to the set of tasks to be scheduled. Compared to the previous definition of the problem, the models are removed from the resources α. The difference between those two definitions is how the models are considered (resources or not). Moreover, the design and collection of those unavailable models belongs to the set of tasks to be scheduled. Hence, using the $\alpha|\beta|\gamma$ notation, the problem is still the same ($R\,m \mid sp-graph, unavail_j \mid \#(C_{max}, Q, Resources)$). The difference lies in the fact that the models are not considered as being part of the resources α.

2.3 Conclusion

Classes of scheduling problems can be identified using the three fields notation:

$$\alpha|\beta|\gamma$$

The field α defines the resources, β the constraints and γ the optimization criteria of the given problem. These fields can be used to search in the literature for the complexity of the problem and a possible algorithm to solve it.

This notation was used to express the Model Architect Problem in terms of multi criteria scheduling. However, different definitions can be used, depending on the viewpoint concerning the relationships between resources and tasks.

Moreover, other definitions of the Model Architect Problem have to be defined in order to introduce the notions of robustness and flexibility in the evaluation of schedules. That is why the fields of multi-criteria fuzzy scheduling and stochastic scheduling must be investigated because they could possibly tackle those notions of robustness and flexibility.

3 Model quality

3.1 Introduction

In simulation-based engineering, simulation is used for a decision-making purpose, in order to choose between alternative architectures for example. The notions of risk and uncertainty have an importance in the decision-making field. Then, the concept of utility that is used to model the preferences of the decision maker over outcomes of interests is developed. In the first subsection, a state of the art concerning these terms is made, and the links with Modeling and Simulation (M&S) found in the literature are shown. Then, model quality assessment in M&S will be studied. Finally, a literature review of the composability of simulation models will be presented. Note that neither Renault nor Airbus Group provided internal references or definitions of the terms risk, uncertainty, utility and quality. The second subsection concerns model quality in the MAP (Model Architect Problem) because in the first section about scheduling, the quality Q was mentioned but not defined in the MAP.

3.2 State of the art

This section introduces several concepts linked with the notion of simulation model quality in M&S: risk, uncertainty, utility and composability. Their connections through model quality are then presented. Finally, the notion of composability of simulation models is defined.

3.2.1 Risk

According to the ISO 31000 standard, the risk is "the effect of uncertainty on objectives". (Chaves, 2014) lists some of the definitions of risk used in the engineering field. The risk can be defined as:

$$E[X] = \int_a^b x.p(x)\,dx$$

$$E[X] = \sum_i p_i.x_i$$

X is in fact a random variable that expresses the severity of the outcomes of an event (it could be a cost, for example), $p(X)$ is the probability density function of X, a and b are the limits for the severities of the outcomes. Hence, $E[X]$ is the expected value of the risk (the first equation for the continuous case, the second for the discrete one). This definition takes into account the classical understanding of the risk R of an event e, which is:

$$R(e) = p(e) \cdot C(e)$$

Thanks to this last equation, we can deduce that the risk of an event is the severity of the risk outcome multiplied by the probability that the event will occur. This definition of risk can be found in (Van Bossuyt, 2013).
As it can be noticed thanks to the ISO definition of risk, the notion of risk is strongly linked with the one of uncertainty. The latter will be defined in the next subsection.

3.2.2 Uncertainty

Uncertainty in the context of Modeling and Simulation (M&S) has been defined in (Walker, 2003) and (Oberkampf, 2002). According to (Walker, 2003), uncertainty is any departure from the unachievable ideal of complete determinism. Moreover, three dimensions of uncertainty in a model appear:
- The location of uncertainty: it describes where uncertainty arises in the model (context, model uncertainty, inputs, parameter uncertainty, model outcome uncertainty);
- The level of uncertainty: from determinism to indeterminacy;
- The nature of uncertainty: epistemic or variability uncertainty;

In (Oberkampf, 2002), a distinction is made between uncertainty and error, even if the notions of epistemic and aleatory (same as variability) uncertainty are used. Aleatory uncertainty is the inherent variation associated with the physical system or environment under consideration. Epistemic uncertainty is a potential inaccuracy in any phase or activity of the modeling process that is due to lack of knowledge. And error is a recognizable inaccuracy in any phase or activity of modeling and simulation that is not due to lack of knowledge.

Then, (Oberkampf, 2002) also defines the phases of modeling and simulation. They are, in chronological order:
- Conceptual modeling of the physical system;
- Mathematical modeling of the conceptual model;
- Discretization and algorithm selection for the mathematical model;
- Computer programming of the discrete model;
- Numerical solution of the computer program model;
- Representation of the numerical solution.

However, feedback can also propagate between these phases. (Oberkampf, 2002) applied the methodology to a missile flight example, listing the uncertainties and errors occurring during each M&S phase.

Another important field of literature about uncertainty concerns uncertainty propagation in simulation models. In fact, given the probability distributions of the random variables of a system (the parameters of the model, for example), the problem is to determine the probability distributions of the system output (the simulation outcomes). In (Chen S. H., 2007), five categories of uncertainty propagation methods for functions are listed:
- Simulation based methods;
- Local expansion based methods;
- Most probable point based methods;
- Functional expansion based methods;
- Numerical integration based methods.

The concept of uncertainty in M&S influences the decision making process and its associated risk. That is why it is strongly linked with the concept of utility that will be developed in the next section.

3.2.3 Utility (decision making)

In decision-making, the aim of the decision maker (DM) is to choose between different alternatives A_k. Each alternative A_k can lead to a consequence O_k. The utility U is a function that represents the preferences of the DM over the consequences in order to discriminate alternatives. Hence, using the notion of utility as a criteria, the DM transforms his decision-making problem into an optimization problem. To make a decision, the DM has to choose the alternative which maximizes the expected value of the utility.

As the utility represents the preferences of the DM over outcomes, it also expresses the DM's behavior towards risk. Therefore, utility functions can express risk averse, risk seeking and risk neutral attitudes.
The theory of utility was developed by Von Neumann and Morgenstern in (Von Neumann, 1944). The Multi attribute utility theory (MAUT) was then created to consider multiple attributes in the utility function. In MAUT, the utility function takes values in a multi-dimensional space in order to manage trade-offs between different criteria.

Decision making in the field of M&S is an issue that has already been identified in the literature. For example, (Radhakrishnan, 2005) uses the concept of utility to solve the decision problem of model selection. The method developed, which is the model sufficiency estimation technique, is used to select the most useful model from a set of models, via the estimation of model utilities. An uncertainty analysis is then performed to determine the decision's confidence. The main interest of this article is that it considers model selection as a decision problem. However, this approach implies the availability of each model. A reverse uncertainty propagation method is also proposed. It deduces sufficiency requirements about model accuracy and cost, for example, from backward propagation of decision uncertainty. This mechanism determines how truthful a model needs to be in order to make decisions.

3.2.4 Model quality in M&S

Model quality is quite developed and formalized in the field of software engineering. For example, (Crnkovic, 2004) classifies quality attributes for component-based systems. Attributes can be directly composable, architecture related, derived, usage-dependent or system environment context determined. This paper treats attributes such as reliability, availability, safety, confidentiality, integrity and maintainability.

On the contrary, the notion of model quality remains quite ambiguous in Modeling and Simulation. Many indicators may have an influence on a M&S application quality. Accuracy is not the only one. (Balci, 2004) develops a quality-centered approach for M&S applications (acceptability criteria, Verification and Validation (V&V) are dealt with).

However, attempts to treat model quality more formally have been made. The report from the Fidelity Implementation Group (Gross, 1999) describes the notion of fidelity in various activities of M&S. The concept of fidelity is relevant because it is what differentiates computer simulations from other computer programs. This report also provides a glossary of M&S terms and a mathematical formalism for fidelity standards.

Nevertheless, other concepts that can be partially mapped to quality have been developed in the literature. For example, the NASA introduces a credibility assessment scale for M&S results in its 7009 technical standard (Standard for models and simulations (NASA, 2008)). The credibility assessment scale is divided into 8 factors classified into 3 categories:
- the M&S development category (verification evidence factor, validation evidence factor);
- the M&S operations category (input pedigree evidence factor, results uncertainty evidence factor, results robustness evidence factor);
- the supporting evidence category (use history factor, M&S management factor, people qualifications factor).

Each factor is evaluated on a 5-levels scale. A global credibility factor is then deduced from the 8 factors. The two first categories are related to the intuitive notion of quality, mentioning verification and validation (V&V), uncertainties and robustness. Third category seems more subjective but has an influence on the confidence about simulation results.

3.2.5 Composability

According to (Petty, 2003) and (Petty, A formal basis for a theory of semantic composability, 2003), composability is the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements. And in practice, they define two types of composabilities:
- the syntactic composability;
- the semantic composability.

These two concepts will be presented in the next subsections. In the frame of composability in Petty's work, a model M is defined as a computable function:

$$M: X \rightarrow Y$$
$$where\ X \in S \times I\ and\ Y \in S \times O$$

In this definition, S is a non-empty set of states, I is a set of inputs, O is a set of outputs, and $s \in S, i \in I, o \in O$ are vectors of integers. Hence, a simulation is the sequential execution of a model and is represented by a deterministic labeled transition system $L(M) = (S, I, M_S)$ where M is a model, and M_s is the state model of M. A trajectory in $L(M, s_0)$ is a sequence of alternating states and inputs beginning with $s_0 \in S$ and defined by:

$$< s_0, i_1, s_1, i_2, \cdots, s_i > or < s_0, i_1, s_1, i_2, \cdots >$$
$$where\ s_k = M(s_{k-1}, i_{k-1})$$

Therefore, there are terminating and non-terminating trajectories.
Then, in order to define composability, several concepts related to validity need to be defined. In M&S, the model being built refers to a natural system. A perfect model M^* can be linked to that natural system. Hence, the validity of other models for the natural system is measured with respect to the perfect model. In order to assess the validity of the model, strong and weak bisimulations can be used to compare the executions of the two models. Bisimulations are binary relations between transition systems. Two systems are said to be bisimilar if they have the same behavior (if their moves can be mapped between each other). Moreover, equivalence relations and metrics can be employed to assess the validity.

Theoretical results exist in the literature that tackle computational problems in composability. For example, (Petty M. D., 2003) treats the computational complexity of the component selection problem for composition. This problem had already been identified and partially solved by (Page, 1999). In fact, the general component selection problem is NP-complete, even if the properties of a component or a selection are already known (via the use of an oracle). It allows to determine the intrinsic difficulty of the component selection without considering the difficulty of objectives decidability. Hence, other problems related to composability also need to be treated (from a computational complexity viewpoint): the determination of component and composition objectives, the validity determination and the composition complexity.

3.2.5.1 Syntactic composability

Syntactic composability refers to the property that simulation models can be connected in a proper manner so that they can exchange information during a simulation. According to Petty, the question is whether the components can be properly connected.

3.2.5.2 Semantic composability

Contrary to syntactic composability, the concept of semantic composability corresponds to whether simulation components can be meaningfully composed, that is to say to which extent a set of individually meaningful (validated) components can be composed into a meaningful (valid) global simulation model.

3.3 Model quality in MAP

The Model Architect Problem (MAP) can now be reinterpreted as the state of the art for the concept of quality was defined in the last section. In fact, enhancing the quality Q of the simulation would lead to increased completion time T to create the simulation models and then perform the simulation, but also to an increased need of resources R. That is why the

trade-off between $< Q, R, T >$ used in the definition of the Model Architect Problem seems relevant. To the best of our knowledge, there is no tool that estimates this trade-off between $< Q, R, T >$ in multi-physics simulations. There is even no solution in existing software to guarantee the quality of a simulation model.

3.3.1 Model semantics definition

The concept of Model Identity Card (MIC) was proposed and is used in the SIM project to specify simulation models and then to verify that a simulation model fulfils the specification. Quality attributes are contained in the MIC description, because they translate a part of the model properties needed in order to run the appropriate simulation.

A first category of properties described in the MIC concerns the ports and variables of the considered model. They are characterized in order to guarantee the syntactic composability between the models. Indeed, these interfaces are the means by which models will exchange information during the simulation. Then, characterizing ports in the MIC would also characterize the syntactic composability.

Then, a second category of information named model quality can be found in the MIC. This category contains indicators from the credibility assessment scale of the NASA mentioned above:
- Validation of models;
- Verification of models;
- Input pedigree;
- Uncertainties;
- Robustness;
- History.

Those indicators are evaluated on a 5 levels scale like the NASA sub factors. However, those levels only indicate the advancement in the associated processes. They do not carry information about the objectives of those processes and the properties they establish. Hence, in order to guarantee the semantic composability of simulation models, the existing view concerning model quality in the MIC is not sufficient.

Semantic characteristics of simulation models need to be defined to characterize the behavior and meaning of those models. Different levels of semantics can be defined by abstraction and refinement processes because concrete semantics are not always suitable or available to describe the expected properties of the model. This mechanism is exploited in the field of abstract interpretation (Cousot, 1992) in computer science. This theory of sound approximation of programs semantics may describe the behavior models. The problem is to find a suitable level of abstraction to describe the behavior of the model for validation purposes. Those semantical aspects are needed to treat the model selection problem because the current definition of the MIC (which mainly relies on syntactic properties of models) does not sufficiently addresses the behavior of the model.

3.3.2 Sub-problem 1: model specification for composability

The first problem concerns the specification of the simulation models to be composed. When the MA is given an architecture of the system to be simulated and a simulation demand, and after decomposing topologically the global simulation model into a network of sub models, the problem is priori how to express and specify the quality properties that would be implemented by the global model and the sub models, in order to guarantee the syntactic and semantic composabilities when the models would be integrated and the simulation performed.

3.3.3 Sub-problem 2: integration of models for composability

The second problem concerns the posteriori check of the quality properties (specified in the first problem) during the integration of the sub models, at both sub model and model levels, before running the simulation. The aim is to check the composability, specifically the semantic one.

3.3.4 Other sub-problems

Another problem to be considered is the one of model selection. Considering a demand for a network of models via the specifications provided by the first problem, and a set of eligible models for each sub model of the network, how to select a candidate for each model of the network and create it effectively in order to maximize the degree to which the quality metrics are implemented.
One last problem is the choice of an appropriate abstraction for the semantics of a model which allows the verification of the specification of the model without being too abstract.

3.4 Conclusion

This section focused on the quality issues that can be found in the MAP. Several problems concerning quality that are necessary to solve the MAP were identified, concerning both the definition of the specification of a model and its verification.

4 Automation methods for numerical simulation

4.1 Introduction

The MIC (Model Identity Card) is a formalism which purpose is to allow the specification and the characterization of simulation models. A MIC can be viewed as a demand of a simulation model. This demand is made by an actor called the Model Architect (MA). The role of this actor is to perform the simulation of a system when given a simulation demand of the system of interest by the system architect (SA). For that purpose, the MA specifies the MICs of the simulation models to be produced. Those MICs serve as an input for the third type of actor, the Model Providers (MP). They have to produce and furnish the simulation models corresponding to the specified MICs. Finally, the MA integrates those models in a network of models according to its simulation architecture, in order to run the simulation and interpret its results.
This process means that MICs contain necessary information for the fulfillment of the M&S (Modelling and Simulation) process. Since the MIC is defined by a formalism, it can be interpreted. Hence, several tasks of the M&S process can be automated using the MIC as an input. In this document, four mechanisms dealing such tasks are introduced and developed. They are:
- MIC2MO: it generates the backbone of a simulation model from its MIC;
- MIC2V: it creates a program that automatically checks requirements on simulation traces;
- CompMICs: it compares two MICs and gives a measure of similarity between those two MICs;
- MO2MIC: it creates the MIC of a model directly from the source code of the model.

Those mechanisms were developed within the scope of Modelica simulation models. The availability of tools dealing with such models (OpenModelica and Python to treat the results of the simulations), combined with the easy access to the Modelica specification (in order to get the BNF grammar of the language) influenced that choice. However, the application

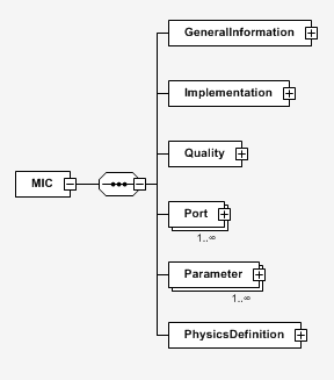
to Modelica models is a proof of concept. Those mechanisms could be extended to other simulation languages provided that their BNF grammar is available.
Finally, those four mechanisms, when combined, allow to generate a network of models, to populate it and to check requirements on its simulation.

4.2 MIC definition and simulation automation with pseudo-MICs

4.2.1 MIC definition

The MIC is defined as a common vocabulary to specify and characterize simulation models. It characterizes a model through the interfaces definitions.
A MIC is composed of different classes of information. They are: General information, implementation, quality, Ports, Parameters, Physics definition.

Figure 1: MIC classes



Each class contains fields of information.

4.2.2 Simulation automation with pseudo-MICs

The work presented in this document deals with the partial automation of the activities linked with the model associated to a MIC. A partial automation would ease the run of simulations in order to treat the quality aspects of the simulation request.
In this work, the only information that is kept from the MICs is the one that is relevant in an automation perspective. This information is the one contained in the classes quality, ports and parameters.
However, with an automation purpose, this information is not sufficient to generate backbones of networks of simulation models and to compare simulation models via their specifications. In the former case, the information concerning the architecture of the simulation lacks, because it allows to describe networks of connected models. In the latter case, the description of the internal components of the model lacks. For example, it is not possible to discriminate between the MIC of a resistor and the MIC of a Lithium battery when the comparison is only based on the definition of the interfaces of the models. However, those elements differ by their parameters and internal variables.
For those reasons, in this work, the specifications of simulation models are not MICs but variations from MICs that are in fact pseudo-MICs. With those pseudo-MICs, a simulation model is specified by the definition of its constituting elements (sub-models, ports, variables, parameters). A hierarchical representation of those elements in the form of a tree is used.
A pseudo-MIC can either characterize a simulation model, a port, a variable or a parameter. A model can contain models, ports, variables and parameters. A port can contain variables.

However, this hierarchical representation of the specification of the model is not sufficient. Adding information about each constituting element of the model would induce more detailed specifications to allow to differentiate them. The problem is to define the granularity of this information. The pseudo-MIC of a model must remain simple to define while it has to embed enough information to characterize the elements. Some of this information is already defined in MICs and will be used in the present work.
For that purpose, properties can be linked to each constituting element, according to its type.
For a model, the allowed properties are:
- The connection: At the level of model m, it specifies a connection between two ports p_1 and p_2 as $Connect(p_1, p_2)$. However, if p_1 belongs to model m_1 and p_2 belongs to model m_2, m_1 and m_2 must be sub-models of m. Moreover, p_1 and p_2 must be consistent with each other, that is to say that they must contain the same number of variables with the same names and units. This latter condition ensures the syntactic composability, that is to say that the two ports can be effectively connected.

- The timestep: a fixed timestep and its numerical value can be specified, or a variable timestep also. Only one timestep can be defined per model.

- The timestep tolerance: a range $[lb, ub]$ specifies the allowed timesteps.

- The propositional variable: a propositional variable of model m is the comparison of a variable v that is hierarchically a descendant of m with a floating value. The comparison operators that can be used are $(<, \leq, =, \geq, >, \neq)$. An example of propositional variable is: $propVar := SOC < 1.01$.

- The requirement: a requirement of model m is defined by a MITL (Metric Interval Temporal Logic) formula on the propositional variables of m. The grammar used for MITL formulas is the following one:

$$\varphi := p \mid \neg\, \varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1\, \mathcal{U}_{[a,b]}\, \varphi_2 \mid \varphi_1\, \mathcal{U}\, \varphi_2$$

Where p is a propositional variable, φ, φ_1 and φ_2 are formulas, $0 \leq a < b$ and $a, b \in \mathbb{Q}$. $\mathcal{U}$ is the until operator. Other operators of logics can be defined:

$$\varphi_1 \vee \varphi_2 \equiv \neg\big((\neg\, \varphi_1) \wedge (\neg\, \varphi_2)\big)$$
$$\varphi_1 \Rightarrow \varphi_2 \equiv (\neg\, \varphi_1) \vee \varphi_2$$
$$\varphi_1 \Leftrightarrow \varphi_2 \equiv (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$$

Other temporal operators can be derived:

$$F_{[a,\,b]}\varphi \equiv \top\, \mathcal{U}_{[a,\,b]}\, \varphi$$
$$F\, \varphi \equiv \top\, \mathcal{U}\, \varphi$$
$$G_{[a,\,b]}\, \varphi \equiv \neg\, F_{[a,\,b]}\, \neg\, \varphi$$
$$G\, \varphi \equiv \neg\, F\, \neg\, \varphi$$

Where $\top$ means True, and $\bot$ False. F means Future and G Globally.

For a variable, the allowed properties are:
- Min: it specifies a minimum value for the variable;

- Max: it specifies a maximum value for the variable;

- Flow: if this property is defined, then the variable is a flow variable;

- Init: this specifies the initial value of the variable;

- Unit: it characterizes the unit of the variable.

For a parameter, the allowed properties are:
- Min: it specifies a minimum value for the parameter;

- Max: it specifies a maximum value for the parameter;

- Value: this specifies the value of the parameter;

- Unit: it characterizes the unit of the parameter.

The recursive representation of a model from the description of its constituting elements associated to the definition of the properties of these elements is the pseudo-MIC representation that will be used in the work described in this document.

4.3 Automation mechanisms

The pseudo-MIC described in the previous section is a formalism that can be interpreted. This allows, given the information embedded in MICs, the automation of several mechanisms of the M&S process. Four such mechanisms are presented in the following subsections:
- MIC2MO: its transforms a pseudo-MIC into the backbone of a simulation model;

- MIC2V: it generates a program that will check the requirements defined in the pseudo-MIC of a model along the simulation traces;

- CompMICs: it compares two pseudo-MICs structurally and gives a measure of similarity between those two pseudo-MICs;

- MO2MIC: Given the code of a simulation model, it generates the pseudo-MIC of the model.

4.3.1 MIC2MO

A pseudo-MIC embeds the structure of a simulation model in terms of sub-models, variables, parameters and ports. This pseudo-MIC can be interpreted recursively to generate the backbone of the simulation model. The backbone contains the declaration of the model, its sub-models, the instantiation of variables (with their possible starting values), parameters (with their value) and ports.
However, the generated backbone cannot be simulated because it has an empty equation section. No behavior is embedded in this backbone since the pseudo-MIC does not contain any information about the behavior of the simulation model. Hence, in order to be able to simulate the model, the generated backbone must be enriched by model providers with equations describing the behavior of the model.

Figure 2: Example of a MIC2MO transformation of the pseudo-MIC of a lithium battery

In Figure 2, an example of a MIC2MO transformation is given. From the pseudo-MIC of a Lithium battery model on the left, the backbone of the simulation model in Modelica code is generated on the right. As it can be seen in the code, all the variables, parameters and ports are declared. The ports are instantiated. Some variables or parameters can also be instantiated with their starting value (for variables) or value (for parameters), depending whether those properties are declared or not at the pseudo-MIC level.

4.3.2 MIC2V

The mechanism MIC2V generates, given a pseudo-MIC, a program that will check requirements on simulation traces. It uses the propositional variables and requirements as defined in the pseudo-MIC specification using MITL formulas on comparisons of simulation variables to values.

Two inputs are necessary in order to run MIC2V: the pseudo-MIC of the simulation model and the file containing the results of the simulation. The pseudo-MIC embeds the definition of the requirements to be checked. The result file of the simulation is also necessary to have access to the values of the variables of interest for the requirements along the simulation run.

Modelica simulation models are considered in this work. The simulation of such models using openModelica software generates a result file in .mat format (the Matlab format). Therefore, given the definition of requirements in the pseudo-MIC and the name of the result file, a Python program is generated. It checks the MITL formulas of the requirements along the simulation run. All the operators of the MITL grammar presented above are defined. A propositional variable is defined in pseudo-MICs by comparing a simulation variable to a floating value. The propositional variable is then calculated along the simulation run by comparing the array of values of the simulation variable (from the result file) to the value. Then, the composition of logical operators combined with the use of propositional variables allows to calculate the truth of the requirements along the simulation run.

A future evolution of MIC2V could be the generation of a formatted report providing the violations of requirements. Such a kind of report could be interpreted automatically.

4.3.3 CompMICs

CompMICs is a mechanism that compares pseudo-MICs structurally and gives a measure of similarity between those two pseudo-MICs.

In this work, pseudo-MICs are interpreted as graphs. Hence, comparing pseudo-MICs amounts to comparing graphs. The graph edit distance is a metric that is used to compare graphs. Comparing graph G_1 with graph G_2 using the graph edit distance gives a measure of similarity

between G_1 and G_2. The aim of the graph edit distance method is to find the simplest manner to transform G_1 into G_2, by transforming nodes of G_1 into nodes of G_2, by deleting nodes of G_1 or by adding nodes of G_2 according to some predefined rules that associate a cost to each unitary transformation. The cost of a global transformation from G_1 to G_2 is the sum of the costs of the unitary transformations that constitute the global transformation. The graph edit distance between G_1 and G_2 is the minimal cost of a global transformation between G_1 and G_2. This value is influenced by the choice of the cost matrices.
Given two pseudo-MICs M_1 and M_2, CompMICs transforms M_1 into a graph G_1 and M_2 into a graph G_2 and then return the graph edit distance between G_1 and G_2.

4.3.4 MO2MIC

The mechanism MO2MIC generates the pseudo-MIC from the simulation model. MO2MIC is in fact a compiler from Modelica code to pseudo-MICs. The Modelica grammar in EBNF (Extended Backus-Naur Form) is available in the specification of the language. It is used to create a lexer and a parser to interprete the Modelica code into an AST (Abstract Syntax Tree). This AST is then transformed into a pseudo-MIC resolving some dependencies created by the object oriented paradigm of Modelica.
The pseudo-MICs generated by the MO2MIC mechanism only contain information that is directly found in Modelica models. However, additional information that is not directly embedded in the code of simulation models can be declared at the pseudo-MIC level. This feature of the MO2MIC mechanism has to be kept in mind. For example, the definition of requirements or the properties linked to timesteps will not be found in pseudo-MICs generated by MO2MIC.

4.4 Implementation

The mechanisms were implemented in Python. The list of files to be used is the following:
- ModelElement.py: it defines the hierarchical structure of pseudo-MICs and the different types of properties associated to each element. It is used to instantiate pseudo-MICs;

- MOLex.py: it defines the lexer for Modelica code;

- MOYacc.py: it defines the parser for the Modelica grammar;

- MICGenerationFromMO: it defines functions to generate pseudo-MICs from Modelica files and folders of Modelica files;

- MOASTToMIC: it defines functions to transform the AST produced by the parser into pseudo-MICs;

- graphRepresentation.py: it defines the structure of a graph, and functions to transform pseudo-MICs into graphs and to calculate graph edit distance;

- UnitParsing.py: it defines a parser and a lexer to transform units in Modelica into a format that is interpretable for pseudo-MICs;

- MICXMLTransformations.py: it defines functions to allow the import and export of pseudo-MICs in XML format;

- MICSetGeneration.py: it defines a function to generate a set of XML files from a set a pseudo-MICs, respecting their hierarchical structure;

- compareMICToMICs.py: it defines a function that is used to compare a pseudo-MIC to the pseudo-MICs that are in one folder;

4.5 Conclusion

Four mechanisms were introduced in this section in order to partially automate the Modelling and Simulation process. MIC2MO generates the backbone of a simulation architecture. MIC2V creates a program to automatically check requirements on simulation traces. CompMICs compares MICs. This allows to search simulation models via their pseudo-MICs in databases of models and specifications. This mechanism can be used to help populate a network of models. MO2MIC creates the MIC of a model from its source code in Modelica. It can be used to capitalize on simulation models.
Those four mechanisms, when combined, allow to generate a network of models, to populate it and to check requirements on its simulation. They were used on the Renault case study of the SIM project (on-board electrical network).

5 Case study: Renault case study (on-board electric network)

5.1 Introduction

The Renault case study concerning the optimization of the on-board electric network of a mild hybrid vehicle was developed. This case study is described in details in the deliverable L4.2.

In the context of legislative measures taken to face environmental challenges, Hybrid Electric Vehicles (HEV) and Electric Vehicles (EV) are strongly emerging solutions. Therefore, the optimization of the costly electrical energy storage system is a key factor in reducing both CO2 emissions and costs of HEVs. The considered problem is the one of sizing and positioning of three components of the energy storage system: a lithium battery, a lead-acid battery and a super capacitor. This optimization problem is mono-objective under constraints. The objective, to be minimized, is the cost of the on-board electrical network. The constraints relate to geometrical, energy and electrical properties of the batteries of the system. The optimization parameters relate to the characteristics of the three components mentioned above (mainly their open circuit voltage and capacity/capacitance). In this optimization, the simulation of the system appears in the verification of the electrical properties of the energy storages of the system. The on-board electrical network is simulated using given rolling profiles during which the states of charges (SOC) and voltages of the two batteries must remain within a specified range.

5.2 Work done

The Renault case study was implemented using the deliverable L4.2 and additional information provided by Renault. The simulation models were developed using the Modelica language in OpenModelica. Packages of models with different complexities were created for the lithium battery, the lead-acid battery, the super capacitor and the DC/DC converter in order to ease the reconfigurability of the global simulation model. This characteristic could be further used to study the notion of composability with this case study.

The optimization problem based on openModelica simulations has been implemented with two ways: the first implementation uses the modeFrontier software and the second one is based on an open-source python code. modeFrontier is a workflow software for multi-objective and multi-disciplinary optimization.

Finally, two formulations of the optimization problem have been considered. The original one, described in deliverable L4.2, is mono objective with simulation constraints. The objective to be minimized is the cost of the system, and the simulation of the system appears in the constraints. In fact, the voltages and states of charges (SOC) of the two batteries must lie within bounds during the simulation (they do represent part of the constraints of the problem). This, in practice, leads to an over-constrained problem. That is why a second formulation of the problem in which the simulation constraints were transformed into a second objective (to be minimized) was proposed. This new formulation is a problem without simulation constraints.

The four automation mechanisms presented in the previous section were used on the Renault case study. MIC2MO was used to generate the backbone of the whole network of simulation models. MIC2V was used to generate a program that checks the requirements on simulation traces. Those requirements corresponded to constraints of the first formulation of the problem. CompMICs was used to test the search of models via their pseudo-MICs in the Modelica standard library (the pseudo-MICs of that library were generated by MO2MIC).

5.3 Conclusion

The original optimization of the Renault case study was performed except that the geometrical constraint was solved manually prior to the optimization. This reduced the difficulty of the problem that was effectively solved. The next step would be to reincorporate the geometrical constraint in the previously mentioned resolution.
The work on this case study is also described in (Fontaine G. a., 2016).
Then, the aim would be to treat the composability problems described in the previous section thanks to the multiple available models for each component of the system. It would also be useful to evaluate the utility of the use of the MIC for the composability problems via this case study.

6 Conclusion

The Model Architect Problem was treated within this report. Multi criteria scheduling was presented and the system of notation of scheduling problems (three fields notation) was introduced. This notation was used to express the Model Architect Problem in terms of multi criteria scheduling. However, different definitions of the MAP can be used, depending on the viewpoint concerning the relationships between resources and tasks.

Other definitions of the Model Architect Problem will have to be established in order to introduce the notions of robustness and flexibility in the evaluation of schedules. That is why the fields of multi criteria fuzzy scheduling and stochastic scheduling must be investigated.

Solutions of those different Model Architect problems should be provided in next versions of this report.

Then, the notion of model quality was dealt with because it intervenes in the definition of the Model Architect Problem, even if it lacks formalization. Its links with the terms uncertainties and errors was developed. The decision-making purpose aspect of the Model Architect activities were also introduced with the notions of risk and utility. The concept of composability of simulation models was introduced, both at semantic and syntactic levels. The links of those notions with the Model Architect Problem (MAP) were then exposed.

Four mechanisms aiming at partially automating the Modelling and Simulation process were presented. They would aid to treat the quality problems during the development of the simulation.

Finally, the last section was devoted to the work performed on the Renault case study of the on-board electrical network: the development of different models for each component of the system, the variations on the formalization and implementation of the optimization. The next step corresponds to the study of the composability of this case, partly via the use of MICs. The goal would be to identify the strengths and weaknesses of the MIC concerning composability.

Références / Bibliographie

Balci, O. (2004). Quality assessment, verification, and validation of modeling and simulation applications. *Simulation Conference, 2004. Proceedings of the 2004 Winter*.

Beiglböck, M. a.-K. (2012). Utility maximization, risk aversion, and stochastic dominance. *Mathematics and Financial Economics*, 1--13.

Ben-Haim, Y. a. (2012). Robustness, fidelity and prediction-looseness of models. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 227--244.

Chaves, A. a. (2014). Comparison of risk analysis approaches and a case study of the risk of incorporating solar photovoltaic systems into a commercial electric power grid. *Systems Engineering*.

Chen, S. H. (2007). A Comparative Study of Uncertainty Propagation Methods for Black-Box Type Functions. *ASME 2007 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*.

Chen, W. a. (2004). Analytical uncertainty propagation via metamodels in simulation-based design under uncertainty.

Cousot, P. a. (1992). Abstract interpretation frameworks. *Journal of logic and computation*, 511--547.

Crnkovic, I. a. (2004). Classification of quality attributes for predictability in component-based systems. *Proc. of Workshop on Architecting Dependable Systems*.

Fontaine, G. a. (2016). Multi-objective Optimization of Automotive Electrical/Energy Storage System. *IEEE International Conference on Industrial Technology (ICIT2016)*.

Fontaine, G. (s.d.). *Theoretical modeling and associated processes for Model Architects in a multidisciplinary environment. Palaiseau, Ecole Polytechnique. Ongoing*.

Georgescu, I. (2012). Expected utility operators and possibilistic risk aversion. *Soft Computing*, 1671--1680.

Gross, D. C. (1999). *Report from the fidelity implementation study group*.

Küttler, S. (2013). *Dimensionnement optimal de machines synchrones pour des applications de véhicules hybrides. PhD Thesis. Compiègne: Université de Technologie de Compiègne*.

Malak, R. a. (2009). Modeling design concepts under Risk and Uncertainty using Parameterized Efficient Sets. *SAE International Journal of Materials and Manufacturing*, 339-352.

Milanese, M. a. (2003). Model quality in nonlinear SM identification. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, 6021--6026.

NASA. (2008). *NASA Standard for Models and Simulations*.

Oberkampf, W. L. (2002). Error and uncertainty in modeling and simulation. *Reliability Engineering & System Safety*, 333--357.

Page, E. H. (1999). Observations on the complexity of composable simulation. *Proceedings of the 31st conference on Winter simulation: Simulation---a bridge to the future-Volume 1*, 553--560.

Paredis, B. D. (2010). A Review of Methods for Design under Uncertainty from the Perspective of Utility Theory. *ASME 2010 International Design Engineering Technical Conf. \& Computers and Information in Engineering Conf*.

Petty, M. D. (2003). A composability lexicon. *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 181--187.

Petty, M. D. (2003). A formal basis for a theory of semantic composability. *Proceedings of the Spring 2003 Simulation Interoperability Workshop*, 416--423.

Petty, M. D. (2003). Computational complexity of selecting components for composition. *Proceedings of the Fall 2003 Simulation Interoperability Workshop*, 14--19.

Pinedo, M. (1995). *Scheduling: theory, algorithms, and systems. 1995*. Prentice-Hall, Englewood Cliffs, NJ.

Radhakrishnan, R. a. (2005). A methodology for model selection in engineering design. *Journal of mechanical design*, 378--387.

Ruderman, A. M.-K. (2013). Simulation-based conjoint ranking for optimal decision support process under aleatory uncertainty. *Journal of Intelligent Manufacturing*, 641--652.

Stevenson, D. (2005). A Critical Look at Design, Verification, and Validation of Large Scale Simulations.

T'kindt, V. a.-C. (2006). *Multicriteria scheduling: theory, models and algorithms*. Springer.

Van Bossuyt, D. L. (2013). A case for trading risk in complex conceptual design trade studies. *Research in Engineering Design*, 259--275.

Van Buren, K. L. (2014). Model selection through robustness and fidelity criteria: Modeling the dynamics of the CX-100 wind turbine blade. *Mechanical Systems and Signal Processing*, 246--259.

Von Neumann, J. a. (1944). *Theory of Games and Economic Behavior*.

Walker, W. E. (2003). Defining uncertainty: a conceptual basis for uncertainty management in model-based decision support. *Integrated assessment*, 5--17.

Zadeh, L. A. (2006). Generalized theory of uncertainty (GTU)—principal concepts and ideas. *Computational Statistics & Data Analysis*, 15--46.

Zhang, S. a. (2013). Concurrent treatment of parametric uncertainty and metamodeling uncertainty in robust design. *Structural and Multidisciplinary Optimization*, 63--76.

Annexes

