A Model Identity Card to Support Engineering Analysis Model (EAM) Development Process in a Collaborative Multidisciplinary Design Environment

Sirin Göknur, Paredis J. J. Christian, Bernard Yannou, Eric Coatanéa, Eric Landel

Sirin G., Paredis C.J.J., Yannou B., Coatanea E., Landel E., 2015. A Model Identity Card to Support Engineering Analysis Model Development Process in a Collaborative Multidisciplinary Design Environment. IEEE Systems Journal, doi 10.1109/JSYST.2014.2371541

A Model Identity Card to Support Simulation Model Development Process in a Collaborative Multidisciplinary Design Environment

Göknur Sirin, Christiaan J.J. Paredis, Bernard Yannou, Eric Coatanéa, Eric Landel

***Abstract*— Today, one of the major challenges in full-vehicle model creation is to get domain models from different experts while detecting any potential inconsistency problem before the IVVQ (Integration, Verification, Validation, and Qualification) phase. To overcome such challenges, Conceptual Design phase has been adapted to the current Model Development Process (MDP). For that, the system engineer starts to define the most relevant model architecture by respecting quality and time constraints. Next, the model architect designs the delivered model architecture in a more formal way to support the integration of domain models in a consistent fashion. Finally, the model architect negotiates with different model providers with the aim of specifying vehicle and domain level models and their interfaces connections. To improve knowledge sharing between mentioned actors, we propose a Model Identity Card (MIC) for classifying analysis modeling knowledge including input/output parameters and quality expectation. The fundamental concepts that form the basis of all simulation models are identified and typed for implementation into a computational environment. An industrial case study of engine-after treatment model is used to show how MICs and integrated model design phase might use in a given scenario. A validation protocol is conducted through a heuristic observation to estimate the rate of model rework and ambiguity reduction.**

***Index Terms*— Common Vocabulary, Ontology, Collaboration, Model Consistency, Multidisciplinary Modeling, Systems Engineering**

Göknur Sirin, Laboratoire Génie Industriel, Ecole Centrale Paris, Chatenay-Malabry, 92290, France and Techno centre Renault Sas, 78288, France (e-mail: goknur.sirin@ecp.fr), (+33) 612978653

Christiaan J.J. Paredis, Georgia Institute of Technology, Woodruff School of Mechanical Engineering, Atlanta, GA 30332, USA (e-mail: chris.paredis@me.gatech.edu)

Bernard Yannou, Laboratoire Génie Industriel, Ecole Centrale Paris, Chatenay-Malabry, 92290, France (e-mail: bernard.yannou@ecp.fr)

Eric Coatanéa, Product Development research group Department of Engineering Design and Production School of Engineering, Aalto University, FIN-00076 Aalto, Finland (email: eric.coatanea@aalto.fi)

Eric Landel, Techno centre Renault SAS, 78288, France (email: eric.landel@renault.com)

I. Introduction

A. Background

Simulation-based design has gained importance in the past few years in many sectors, especially in aerospace and in automotive manufacturers where modern engineering products are becoming increasingly complex [1]. One of the benefits of employing simulation models is that it makes design verification faster and less expensive; it provides also the designer with immediate feedback on design decision [1]. However, over the last 15 years, in addition to product variety, the environmental (i.e., fuel-efficient, low-emission) and economic concerns increase the complexity of modern products. To keep pace with the rapid improvements being made to these modern products and maintain competitiveness with other manufacturers, design cycles have become shorter and more efficient. To support these shorter design cycles during the early design exploration phase, companies use more and more sophisticated, multidisciplinary simulation model (or numerical or behavioral models) [1]. A complex product development process requires that one decomposes the system to be able to understand the whole's details. In the context of automotive design, a vehicle may be partitioned by objects into body, powertrain, and suspension subsystems. Aspect partitioning divides the system by discipline. The same automotive design could be partitioned by aspects into structural, aerodynamic, and dynamics disciplines. Focusing on the companies' organizational level, the vehicle partitions affect also the tasks, roles and simulation models delegation between related engineering disciplines. The complex, multidisciplinary (or multi-domain) simulation model creation process involves a number of parallel or/and sequential activities in which experts in different domains, possibly in different companies (i.e., suppliers and sub-tier suppliers) create, reuse and exchange domain (or component or atomic) level simulation models to build up a full-vehicle system model [2]. Each engineering discipline tends to use its own domain-specific languages, tools and methods to model different aspects of a system concurrently. Imperfect interoperability

between the OEMs, suppliers and their tooling causes costs overruns and delays. Distributed design teams typically handle the model at different levels of abstraction, ranging from very high-level system decompositions to very low-level detailed specification of components [3]. This is particularly challenging for the design of multidisciplinary systems in which components in different disciplines (e.g., mechanics, structural dynamics, hydrodynamics, heat conduction, fluid flow, transport, chemistry, or acoustics) are tightly coupled to achieve optimal system performance [2]. In this multidisciplinary collaborative design environment, most of the engineers modify the existing simulation models to fulfill a specific purpose for which they were not originally made; it can be a source of inaccuracy, uncertainty, duplication and time delay. To this end, Model Validation and Verification (V&V) plays a key role in mitigation such risks. In this paper a simulation model is considered valid under a set of experimental conditions, if the model's response accuracy is within acceptable range for intended purpose [4, 5]. To be able to build the "right" or "valid" model, in engineering practice, designers need to use the component level simulation model as a black box fashion. Although, most of the technology or service provider's simulation model is in Black Box (BB) format for preserving the intellectual properties. BB models can be interacted with only through the inputs and outputs of a well-defined interface. The challenges to use BB model is to take into consideration the number of distinct interface issues, parameters, or messages that have to be passed among the components. In addition to the model interface consistency problem, there are many other factors that may cause inconsistencies, such as human error, miscommunication between teams and misunderstood assumptions. These inconsistencies are all sources to uncertainty and its propagation in multidisciplinary modeling environment is more complicated than in a single disciplinary domain [5, 6]. The effect of the uncertainties in one domain model may propagate to another through interrelated variables, and the system output finally suffers from the accumulated effect of the individual uncertainties. Thus, the information flow in modeling practice is one of the key aspects of its uncertainty, which may imply risk that the product attributes does not ultimately meet user needs [2].

B. Problem Statement and Our Contributions

Today, many companies use the V-cycle Systems Engineering process for product development as proposed by Frosberg and adapted by the National Council on Systems Engineering [7]. In this process, OEMs take the responsibility of requirement specification, system design, and integration and Verification and Validation (V&V) steps. This is followed by the supplier, which develops the domain models. Although the simulation model is tested at the supplier level, the OEMs are responsible for the final integration, system and acceptance testing to ensure that the given implementation of a system level model meets its intended goals and demands. In this process, most of the defects are discovered late, during the IVVQ (Integration, Verification, Validation, and Qualification) phase. This may create multiple wastes, including rework and, may-be the most harmful namely, incorrect simulation models, which are subsequently used as basis for design decisions. According to de Weck, early design validation and verification may reduce rework in the more expensive implementation and physical prototype validation phase, which is the main driver for product development cost [8]. It has been estimated that the cost of imperfect simulation model interoperability is at least $1 billion per year for members of the US automotive OEMs [9] and $400M in an aircraft development program with 1-2 months of delays [10].

There are many potential sources of inconsistency such as error in the simulation model code or mismatch of interfaces connections (i.e., differences in time step, units, solver, hardware and software versioning…). However, a significant proportion of defects are associated with interfaces between modules or between requirements and implementation rather than a design or coding error within a single module. Leveraging the interface inconsistency problem requires a **Conceptual Model Design** phase and an interface consistency checking at early development stage. Model Design phase contains schedule and transparency agreement between OEMs and domain model providers. Today, in model development environment, there are two main actors; System Architects and Domain Model Providers. **System Architects** are the sponsor of model development activity. He or she defines the projects' expected time, cost and decision parameters issues. **Domain Model Providers** are the domain experts who build models with theirs specific domain knowledge. As illustrated in Fig. 1, the System Architect has a functional (system world) view. He defines an operational scenario, a trade-off analysis and provides a draft version of model architecture for Domain Model Providers who has a physical view. One of the gaps that we noticed during our two years of research investigation in the automotive OEM Company is that there is no clear and formal scheduling and transparency agreement between System Architect and Domain Model Provider at early model development stage (or conceptual design phase). In addition, most of the time, interaction of these two actors may create a bottleneck for communication because they do not have the same level of understanding. One of the interests of this work is to create a more formal and clear model request to the domain model providers, in order to obtain the right model at the right moment. Another gap in model development activity is the lack of detailed Model design phase at early model development stage (downstream of V-cycle) (see Fig. 2). As illustrated in Fig. 1, in the middle, Model design activity contains formal model architecture design with domain models' interfaces definitions. Vehicle level and domain level model specifications include an early interfaces consistency control between specified interfaces. Model design phase gives a structural and semi-behavioral view about the system to be modeled. Thus, this transversal view from Functional to Physical View should be managed by a new actor of the collaboration named "**Model Architect**". Each Model Architect has a multidisciplinary vision of a product, and simulation knowledge. They have also a deep understanding of both the system-level requirements for the vehicle model, as

well as how their models must interface with other domain models (see Figs. 1 and 2).

Therefore, knowledge sharing is one of the key points between these three actors but today's internal communication is ensured by informal knowledge sharing and it becomes an error prone activity because; different disciplines often use different vocabularies, so that semantic differences can cause misunderstandings between these actors [3].

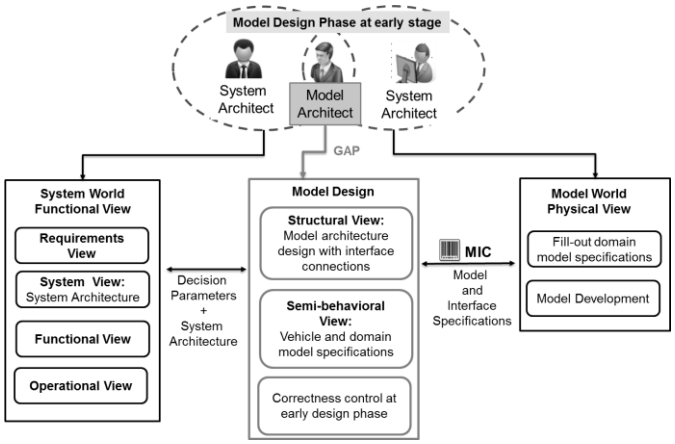


Fig. 1. Research Gap in Collaborative Model Development Process

To obtain an effective knowledge transmission, we need to establish a formal knowledge sharing via a common vocabulary. Providing a common vocabulary for the M&S users can help communicate fact-based decision in a maker's assessment of the credibility of M&S results. The ability for users to select from a list of options is an immensely important capability. Because creating full-vehicle simulation models is a multidisciplinary process, it is important that the same strategies are used across different teams of domain experts. By limiting large groups of users to the same vocabulary and set of options whenever possible, inconsistencies arising from miscommunication or misinformation can be reduced significantly. Consequently, the potential contributions of this work are

- to create a common vocabulary named "**Model Identity Card (MIC)**" for simplifying simulation models specification, sharing and reducing ambiguity,
- to reduce the amount of rework caused by interface inconsistency between domain models, and
- to propose the tools used to establish and to demonstrate the applicability of the proposed method.

The remainder of this paper is organized as follows. In Section 2, we give a general literature review and explain three principal steps of proposed detailed model design phase. In Section 3, we introduce our methodology about common vocabulary creation called "Model Identity Card (MIC)" for classifying and simplifying simulation models specification. Section 4 introduces an after treatment model's case study and MIC's validation protocol. The conclusion and future research are given in Section 5.

II. Methodology and State of the Art

To create a multidisciplinary vehicle model, it is necessary to plan and develop detailed domain level models according to vehicle-level goals and requirements. Once these domain models have been planned, developed, verified, and validated, they can be integrated together to simulate a complete vehicle. To be able to detect any potential integration problem in the early design phase, we add a detailed model design phase with a correctness check in the current model development process which improves the traditional V-cycle. It helps to synchronize actual needs of the downstream process with what the upstream process delivers. As mentioned earlier, in a design process there are different stakeholders such as Model and System Architects and Model Providers, who have different needs, views and viewpoints. In Model Based System Engineering, views and viewpoints can be used to model the perspectives of different stakeholders and their interests. A viewpoint describes a particular perspective of interest to a set of stakeholders, while a view is a stereotyped package that is said to conform to a particular viewpoint [7]. Dassault Systèmes has already used the MBSE view/viewpoint approach called RFLP (Requirements/ Functional/ Logical/ Physical) in their industrial tool (i.e., Catia V6). RFLP model describes the left-hand descending branch of the "V-Model". Based on the well-known V-cycle design process, RFLP allows concurrent engineering to coordinate the separate activities and views of distributed design teams. R defines Requirements view. The Functional view (F) defines what the system does operational (Intended use). The Logical View (L) or logical/organic architecture defines how the system is implemented, the breakdown structure, the block diagrams, logical interfaces, logical connections, the behavior (discrete behaviors, physics behaviors, and hybrid behavior). The Physical View (P) defines a virtual definition of the real world product. In this paper, we customize these different views based on our needs. We regroup requirement and functional view as a Functional View and we use the term Structural View instead of Logical View (see Fig. 2).

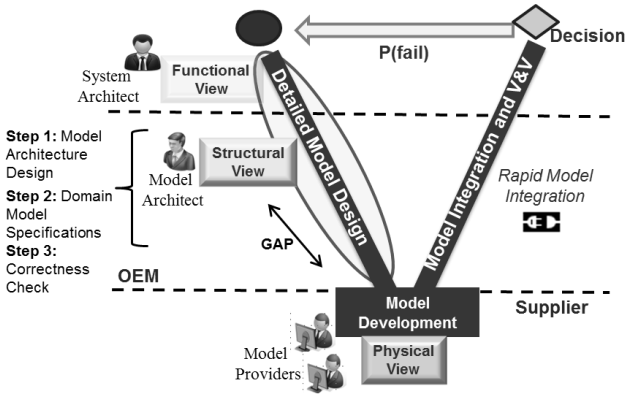


Fig. 2. Integrated model design phase to V-cycle systems engineering process

As shown in Fig. 2, the communication between stakeholders starts with model specifications, and ends with model delivery with well-defined documentation. To ensure the communication between these stakeholders (i.e.; System Architects, Model Architects and Domain Providers) is a challenging task, they either presume a common understanding of a domain of discourse or state their assumptions explicitly. Communication is ambiguous when the assumption of common understanding is incorrect.

In this section, Detailed Model Design phase are explained through Model Identity Card (MIC). MIC is created more specifically to characterize the model (object itself and its interfaces), and to ensure the transparency agreement between Model Architects and Model Providers. The proposed structural view consists of three main steps which are Formal Architecture Design; Vehicle and Domain Models specifications with MIC and Correctness Control at the early design phase (see Fig. 1 and 2).

Step 1: *Formal Vehicle Architecture Design*

System architecture selection and characterization is extremely useful in complex, multidisciplinary vehicle system analysis. Architectures provide a holistic view of a system and allow different stakeholders to work together with a common basis in the same vehicle system definition [12]. To design a good vehicle, it is necessary to analyze each of these system architectures from a variety of perspectives including performance, fuel economy, or even thermal behavior [2, 13]. Creating all possible system architectures manually is necessary for the first time but the reuse of an existing architecture is recommended because of time and cost concerns. To avoid having to build complete vehicle model architecture from scratch each time, it is useful to develop a pre-wired vehicle model architecture. The approach for achieving this relies on a reference model for vehicle architectures [2]. AUTOSAR, for instance, is the product of an industry-wide effort to produce a standardized architecture for controls design and development that can be used among major OEMs and their suppliers [14].The term referential architecture has been already addressed and developed internally by Ford Motor Company as the Vehicle Model Architecture (VMA) [2, 15]. The foundation for the entire approach is Vehicle Reference Architecture (VRA) model, a formal SysML model that defines the logical decomposition of a vehicle into its subsystems. Although there have been several research efforts that have focused on enabling structural model design within a MBSE context, most of these efforts have focused on the integration between a Systems Modeling Language (SysML) model and a variety of simulation model tools (Simulink or Modelica) [2, 15]. However, many engineers today are still unfamiliar with SysML. Because of this, the training and licenses needed to put SysML tools into practice across large engineering teams can be cost-prohibitive. In addition to SysML there are some non-formal architecture design tools such as MS PowerPoint and Visio that design engineers use frequently in current engineering practice because of its easy usage. Thus, in this paper, we adopt a different approach. Rather than assuming that the structural model design is possible with SysML environment, we prefer to use a system engineering tool named arKItect because this tool allows us to easily specify the architecture of system in a hierarchical manner. In this tool, the functional flows describe the interactions between the system functions as well as the interactions between the system and the external environment. The flows can be either data or physical flows. Based on a powerful hierarchical type definition, arKItect allows designing very easily our own meta-model by using a given meta-model structure: objects, flows and their composition rules. To the best of our knowledge there is no similar project that has been developed to support the edition of model characterization via Model Identity Card (MIC) in early M&S design. The design of formal system architecture is one of the activity areas of Model Architects.

Step 2: *Vehicle and Domain Models Specifications with Model Identity Card (MIC)*

Complete vehicle model allow different vehicle-level attributes, such as energy, safety management and performance to be examined and optimized for various operating scenarios. These vehicle-level attributes are tightly coupled; investigating the tradeoffs between these attributes is crucial for system design. Model architects and domain model providers are supposed to specify the domain and vehicle level models via MIC. While the domain model specifications are intended to specify what kind of model to create, the vehicle model specifications specify how all of those domain models will be integrated. Model requirements include defining the operating system that models should be compatible with, expected accuracy, robustness and which simulation environments will be used to run the integrated vehicle model, and any other guidelines that the domain model should comply with. While domain engineers can view this information, they do not have the authority to directly modify any of it. This distinction is important and is made at several other steps throughout this process. By identifying who has the ultimate authority to make decisions during different phases of the modeling process, the potential software tools and MIC created to support it can be tailored to different users. Another important specification is about predefined set of interfaces so that it may correctly integrate with the other domain models. The set of interfaces modeled is derived directly from the details of the specialized analysis architecture. These specifications are the utmost importance because it could result in major inconsistencies between the models created by different domain engineers. This complete list of model attributes is then reviewed by the model architect, who negotiates with each domain team to develop a consistent set of models for the entire vehicle. These system and model architects must negotiate with both the domain engineers providing interfaces and those receiving interfaces, so that all of the analysis models are compatible. Because this is an iterative process, it may require several rounds of negotiations with all the different teams before a common vehicle-wide set of interfaces can be agreed upon. Using a formal check list and a correctness control is critical for early virtual prototype validation.

Model integration and exchange problems have already been addressed in various industrial projects such as ISO 10303 – STandard for the Exchange of Product model data (STEP) for

efficiently exchanging electronic product data between product life cycle tools. Since then, there has been a strong push to effectively use structured knowledge to improve the work in the engineering domain because; the collaboration between knowledge experts in different domains is one of the first steps towards effective knowledge management strategies [16]. Recent research efforts focused on ontologies and ontology development methods for engineering design. Ontologies are extensively used to formalize domain knowledge with concepts, attributes, relationships and instances resulting in reliable, verifiable and computer-interpretable knowledge mappings of a domain [17]. Formal engineering ontologies are described by Ahmed et al. [18] as a six-stage methodology for engineering design context. Li et al. [19] propose an Engineering Ontology (EO) based semantic framework for representing design information in documents, thus aiding their efficient retrieval. Horváth et al. [20] propose formalizing design concepts using ontologies. It is evident that ontologies not only provided formal structures for concepts and vocabularies, but they also have the potential for supporting inferences based on collective knowledge [21]. Shortly thereafter, the application of the Semantic Web in the field of knowledge management is discussed by Fensel et al. [22]. Earlier efforts in semantic mark-up languages include the Extensible Markup Language (XML), Resource Description Framework (RDF), Ontology Inference Layer (OIL) and Defense Advanced Research Projects Agency (DARPA) Agent Markup Language (DAML). Currently, the Web Ontology Language (OWL) is the de facto standard for developing and representing ontologies. OWL is recommended by the World Wide Web Consortium (W3C) as the ontology language of the Semantic [23, 24].

Ongoing work in the M&S realm, recent versions of the DEVS formalism provide for modularity and integration with HLA (DEVS/HLA), but DEVS does not spell out a formal language [26]. In addition, there are also some industrial standards, such as the language known as the SAE Architecture Analysis & Design Language (AADL). An AADL model describes a system as a hierarchy of components with their interfaces and their interconnections [25]. AADL components fall into two major categories: those that represent the physical hardware and those representing the application software. It describes both functional interfaces, and aspects critical for performance of individual components and assemblies of components. The most extensive previous research on characterizing model's behavior in engineering analyses is performed by Grosse et al. [23]. This ontology draws upon some of the analysis modeling taxonomies and concepts presented by Noy and McGuinness [24]. They organize the knowledge about engineering analyses models into an ontology, which includes both meta-data (eg, author, documentation and meta-knowledge, such as model idealizations and the corresponding justifications. A similar, although less extensive, meta-model for simulation models has been developed by Mocko et al. [13] but these taxonomies do not include detailed model behavior characteristic and any model validation and verification attributes such as Nasa's credibility assessment [27]. Based on the aforementioned standards and methodologies, we have developed MIC meta-model that might be applicable to any numerical model in the context of vehicle manufacturer. MIC meta-model will be explained in section 3.

Step 3: *Correctness Control at the Early Design Phase*

Correctness rules have been defined based on observed model interoperability and integration problems such as inconsistent in units, accuracy intervals, and model and hardware versions. Correctness control at the conceptual design phase can eliminate some of frequently faced problems.

The aforementioned three steps are evoked again through an industrial case study in section V.

III. MODEL IDENTITY CARD (MIC)

A. Motivation for M&S Meta-Model Creation

The need for standardized terminology in design artifact is often overlooked in the literature; however, it is an issue of critical importance. Our primary foundation is based upon the concept that simulation models are knowledge-based abstractions of real systems. On the other hand, the number and the diversity of the simulation models require another level of abstraction called Meta-Model that makes statements about the structure of different nature of model without making statements about their content (see Fig. 3).

Assuming that symbol **M** represents a domain model and **M (1..n)** are the different natures of domain models (i.e., 0D-3D), **α(M)** is the abstraction of a domain model,- thus **γ (α(M)) ∋M** means that the concretization of model abstraction has to contain necessary information of the model that we aim to specify (see Fig. 3).

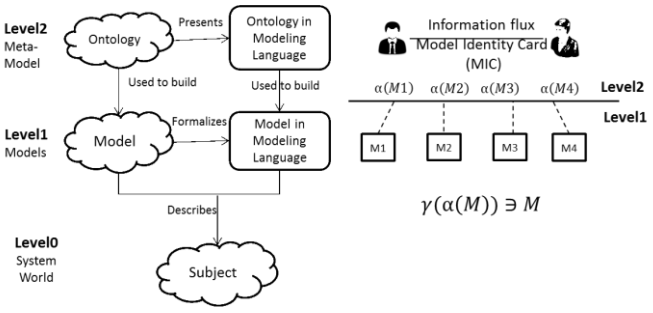


Fig. 3. MIC Concept

MIC meta-model includes some important and refined characteristics of simulation models such as modeling assumptions, behavior and interfaces specifications. This can help communicate the most objective decision in a maker's assessment of the credibility of M&S results. To reinforce this fundamental point, the concept described here does not claim to provide a measure of credibility. Its function is to enable clear communication between M&S stakeholders. We argue that formal taxonomy, or vocabulary for the representation of simulation modeling knowledge, is needed to ensure the following aims:

- to facilitate the model specification (object itself and its interfaces),
- to obtain an effective knowledge transmission and
- to reduce the rework caused by interfaces mismatches (Correctness control with MIC (Step3)).

The main long term benefits of this work include significant reductions in time, effort and interface based defect reduction throughout simulation-based decision support activities.

The MIC is developed by fifteen engineers of at least five different disciplines (Thermal Comfort, Motor, Acoustic, Electric, Vibration, etc…) in Renault automotive manufacturer company. They met more than 20 times between September and December 2013 to facilitate and standardize data collection phase using brainstorming and nominal group technique. Nominal group technique is a structured variation of small group discussion methods [28]. The process prevents the domination of discussion by a single person, encourages the more passive group members to participate, and results in a set of prioritized solutions or recommendations. All participants asked to write their ideas anonymously (or in small groups). Then the moderator collected the ideas and each is nominated on by the group based on proposed scenarios. Depending on engineer's domain knowledge and experience, we defined current MIC attributes.

Roadmap for MIC Creation

MIC is created according to the following steps:

Step1. Identification of main classes and attributes of models.

Step2. MIC attributes grouping and MIC graphical user interface creation on arKItect system engineering tool.

Development Step1: *Identify main classes and attributes of models*

MIC characterizes a model into 2 main classes: Physical Object with 3 sub classes (Methods, Usage, Validation and Verification) and Interface. Each main class consists of numerous attributes itself (see Fig. 4).

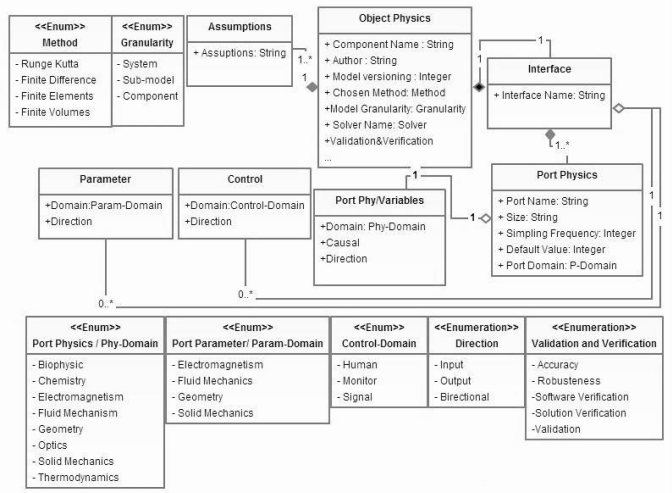


Fig. 4. A part of MIC Meta-Model

In Table 1, we identify classes and attributes of all physics based numerical models. The first column is the term employed to represent this modeling knowledge concept (attributes). The second and the third columns are the attributes of related domain and sub-domain and its type, and the fourth column gives some real examples. As shown in Table 1, the Object Physics class consists of some basic attributes such as: Specific Name, Granularity, Author and Model Version etc. Some of the attributes also have sub-attributes; for example, model granularity consists of system, sub-system, and component. The Method sub-class consists of Chosen Method, Precision, Solver, Time Step, Linearity, Continuity and Model Dimension etc. The Usage sub-class consists of Compilability, Time Computation, Scalability, Software Name, Software and Hardware Version. The Verification and Verification (V&V) sub-class attributes are based on NASA's model credibility assessment [27].

TABLEAU I
MIC CLASSES AND THEIR ATTRIBUTES

Attributes	Remarques	Type	Example	Main Classes
Generic Name *	Physical componant regroupment	String	Engine	Object Description
Specific Name *	Unique identifier	String	Compressor 7V16	
Granularity Level *	List(System/Sub-system/Componant)	String	Sub-System	
Developer Name *		String	F.Ravet	
Model Version no. *	x.x format	Float	0.1	
Creation Date		Date	14/03/2013	
Documentation	Attached technical report	String		
Image	Attached references image	Image		
Model Dimension	List (0D-3D, mix)	String	1D	Method
Chosen Method	List (Finite Volumes, Finite Elements, Finite Difference, OD…)	String	Finite Difference	
Physical Equations	List (Chemistry, Dynamic behavior of materials, Maxwell, Navier-Stokes, Strength of materials, Electric, Signal, Runge Kutta)	String	Navier-Stokes	
Integrated Solver	List (Controllable Pitch, Fixed Pitch, Without Solver)	String		
Time Step	List (Second, Minute, Mili-second, Hour, Steady state)	String	Second	
Linearity	List (No/Yes)	String	No	
Discontunity	List (Yes, No)	String	Yes	
Name of Compilator	List ()	String	Yes	Usage
Time Computation	List (Elapsed Time / Real Time)	String	Elapsed Time	
Scalability	List (Yes/No)	String	Yes	
Tool Name	List (Amesim, Matlab Simulink, GT-Power, Modelica…)	String	GT-Power	
Tool Version	x.x format	String	7,3	
Hardware Requirements	CPU, OS etc…	String		
Accuracy	Requested/Provided Accuracy	Float	%+-5	Model Quality
Robustness	Requested/Provided Robustness	String	1	
Software (Code) Verification	List (Candidat/Development/Previous/Reference)	String		
Solution (Mathematical) Verification	Level 1(Poor), Level2 (Satisfactory), Level3 (Good), Level4 (Excellent)	String		
Validation	Level 1(Poor), Level2 (Satisfactory), Level3 (Good), Level4 (Excellent)	String		

All models must have clear interface definitions that implement the communication within model components or between model components and outside environment [31, 32]. Interface does not contain any information about the internal behavior of the component. Instead, the interface encapsulates the implementation of the model, which defines the internal behavior of the component. The meaningful composition of models requires that their behavior along a number of dimensions be understood and characterized in a formal way that avoids the ambiguity of textual documentation and enables automated processes to configure, compose, and mediate component-based simulations [6].

Interface's attributes are developed for respecting laws of conservation. More precisely, all physical systems have in common their conservation laws for energy and mass. Bond graphs concern themselves intimately with the conservation of energy in a physical system. Firstly, the workgroup creates a tree of diagram of Interface description. We distinguish the nature of interface into three main classes which are parameters,

control and physics and each main interface class attributes can be divided into domain, sub-domain and unit. This tree diagram provides a good level of abstraction for domains and sub-domains (see Fig. 5).

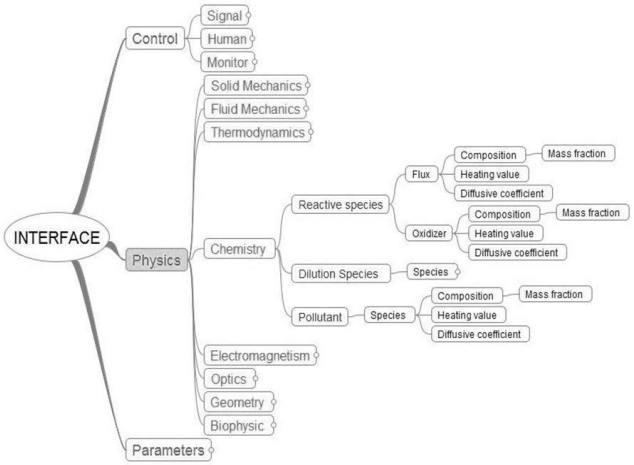


Fig. 5. Interface Characterization

TABLEAU II INTERFACE ATTRIBUTES

Attributes	Sub-Attributs	Type	Example	Main Class Name
Port Name		String	ComPression-Input	Interface
Nature	Control (I/O), Parameter, Physical	String	Physics	
Domain	Solid Mechanics, Fluid Mechanics, Thermodynamics, Chemistry, Electromagnetism, Optics, Geometry, Biophysic, Signal, Human, Monitor, Geometry, Durability, Solid Mechanics…	String	Fluid Mechanics	
Causality	List (Acausal, Causal X2, causal X3,Causal X4)			
Direction	Input, Output, Bidirection			
Sub-Domain	Fluid Mechanics (Acoustics,External aerodynamics,Reactive/ diphasic flow), Thermodynamic, Chemistry…	String	Reactive/diphasic flow	
Variable	Digital(CAN,Ethernet,Optic Fiber), Analogic (Filaire,Radio), Evaluation(Acoustic Comfort,Vibration Comfort,Thermal Comfort,Performance,Durability,Drivability, Ergonomy,Consomption), Pressure…	String	Pressure	
Unit	List (°C, K, kW, W, bar, Pa)	String	Mpa	
Deterministic /Statistical		String	Deterministic	
Offset		String		
Size *	List(Scalar, Vector, Matrix)	String	Scalar	
Min *		String	1,00E+0,5	
Max *		String	2,00E+0,6	

The system interface specification identifies input and output port by name, data and communication type (i.e., input, output, bidirectional, or causal). We identified causal and non-causal interfaces based on bond graphs representation [29].The connection mechanism of model specifies the interface definition and connections. If the connection is a causally coupled relationship, it is called causal connection. Causality is the ability of the model to help establish causal relationships between output parameters and input parameters. The causal connection expresses the interface as input/output relationship. If the connection is non-causal relationship, it is called non-causal connection. The non-causal connection expresses the interface as variables shared relationship [6].

As shown in Table2, Interface class consists of several attributes and sub- attributes. Efficient methods and tools are anticipated for extracting analysis modeling knowledge from engineers, and incorporating this knowledge into a computational environment. Finally, we need methods and associated tools that can exploit the existence of such knowledge in a computational environment, to improve complex model integration and design processes. Therefore, the fundamental concepts that form the basis of all numerical models are identified, described, and typed for implementation into a computational environment. An industrial tool arKItect is used to instantiate the MIC meta-model and illustrate, how common vocabulary usage such as MIC, might improve the ability of model characterization. We integrate a graphical user interface based on MIC vocabulary to create a semantically-rich model characterization support.

Development Step 2: *MIC attributes grouping and integration to the arKItect system engineering tool*

We suppose that each numerical model is as an object and it has various attributes that describe and frame it. Most objects are physically part of a system, and they interact with other objects to create a larger object. In order to extract objects in a system as a unit and understand them, such as, object interfaces, it is necessary to characterize and reuse an object and its interactions with other objects in the system. Here, we decompose the object into three levels which are object itself, object interface and object context (see Fig. 6).

In order to characterize the object itself, we have grouped 'Object physics', 'Method's' and 'Model quality' attributes under the heading: '**Object**'. **The object interface** consists of all Interface class attributes, additional assumptions and dependencies. Whereas, the object and object interface dimensions provide a snapshot of the object at hand. **The object context** dimension provides historical information such as: its usage and software version etc (see Fig. 6).

ArKItect system engineering tool is used to characterize the models via a MIC in a hierarchical manner, and to generate semi-automatically structural model architecture as a block diagram. This tool is used also for demonstrating the applicability of MICs to a particular case study.

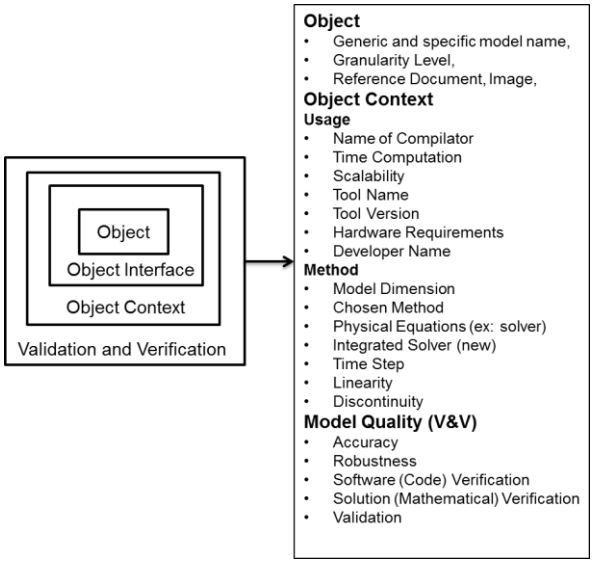


Fig. 6. Object Model classes grouping

(Adapted from Basili and Rombach [33])

IV. CASE STUDY

A. Demonstration Scenario and MIC Validation

Today, the simulation model supply especially from an external provider is a bottleneck activity. Automotive manufacturers ask for having a new model or a customized existing model from the suppliers. In the case of a new model supply, from requirement elicitation phase to model integration tests, as there is not a common vocabulary; the probability to fail during the model integration is very high. The source of problem is mostly based on wrong or insufficient knowledge transmission from automotive manufacturer to model suppliers. As the assumption of common understanding is incorrect, the provided model does not totally conform to the requirements and the mentioned activities take a lot of time, typically 1 to 6 months after several meetings and integration tests. In the example scenario, the system architect wishes to know the behavior of after treatment model with defined boundary conditions. The scenario is realized based on the **3 model design steps** that we explained in section 2.

Thus, the aims of this Case study are illustrated as below:

- *Formal Vehicle Architecture Design based on non-formal model architecture.*
- *Vehicle and Domain Models Specifications with Model Identity Card (MIC).*
- *Correctness Control at the Early Design Phase (MIC).*

Step 1: *Formal System Architecture Design based on non-formal model architecture*

First, the System Architect provides a draft version of the model plan which is most of the time on MS PowerPoint tool and the study objective.

Example:

Technological objectives: Optimizing a combination of after-treatment components. List of the components to be simulated: Catalyst, SCR, Particles filter. Engine applications: K9, R9M, M9R.

Decision: Volume, Mass and Cost analysis of each after-treatment components.

Application: Transient simulation on homologation cycles for passenger car vehicles.

Expected accuracy: 10 % on the volume, 5% on the mass and 1% on the cost.

Once the model architect receives and checks the consistency of the model and study objective (see Fig. 7), he transforms this model plan into a semantically rich block diagram on the arKitect tool. Formal model architecture integration to the early design process facilitates model use activity as a plug-in manner by providing a holistic sub model integration schema with its interface connections.

The model architect characterizes each incoming and out coming ports of related sub models and their connections with each other's. Each sub model has a MIC in which there are enough and necessary information about related model and its port connection with other models. As an example, we characterize "Combustor/Chamber" sub model's interface by using MIC GUI (Graphical User Interface), see Fig. 7 and 8. Each port definition consists of Port Name, Port Nature, Direction, Domain, Sub-domain, variables, Units, Size, Min, Max values, Resolution, Accuracy etc (see Table2). Once we define the incoming and outgoing ports and their connections with other sub models one by one, arKItech semi-automatically generates a block diagram, (see Fig. 9).

Step 2: *Vehicle and Domain Models Specifications with Model Identity Card (MIC)*

The attributes that we use in MIC GUI decrease the ambiguity and misunderstanding between model architect and model provider. This communication problem which might happen has an effect on model quality and decision. Some interoperability problems based on misunderstanding of software versioning, undated libraries and model units, can be decreased by using a MIC. By doing this it minimizes time, cost and the expertise required to construct comprehensive models within the context of their organization. Once the model architect sends on-demand requests to the model provider, he/she has to find or develop the requested model. In the context of after treatment example, the model architect sends the request to different engineering domain to be able to create a complete high level model.

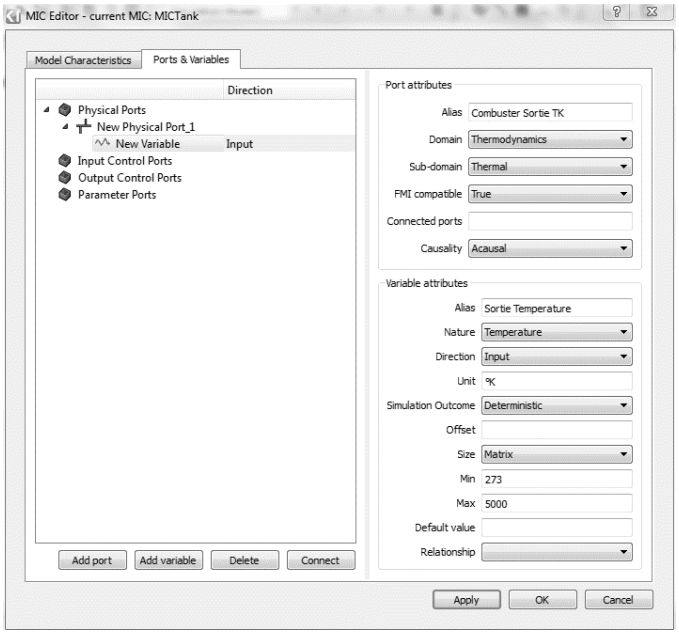


Fig. 7. GUI MIC Interface description / Port creation

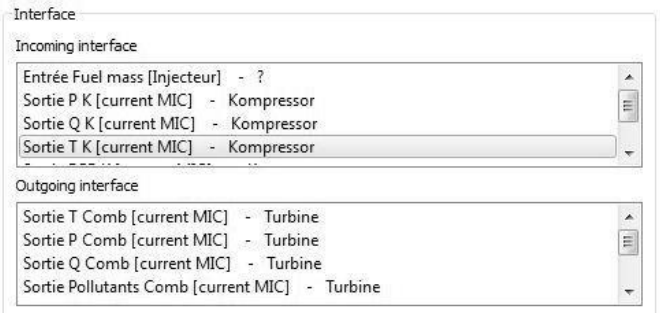


Fig. 8. GUI In-coming, out- coming ports

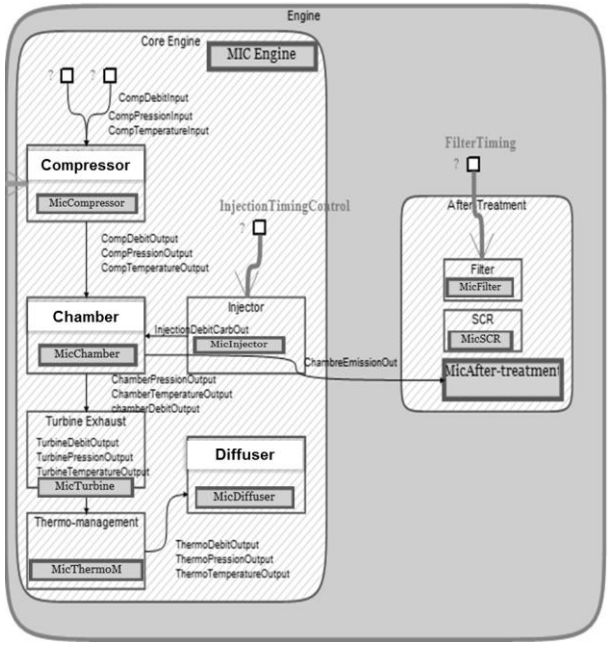


Fig. 9. Generated block diagram (formal version)

After having established the communication via MIC, the model provider supposed to fill out also some important attributes before pointing out the relevant model and sending the complete MIC to the model architect. As shown in Fig. 10, the model provider must fill important attributes, such as, the name of method that the model provider used to develop his or her model and the model precision, solver name, dimension, time step, software tool name, versioning etc…

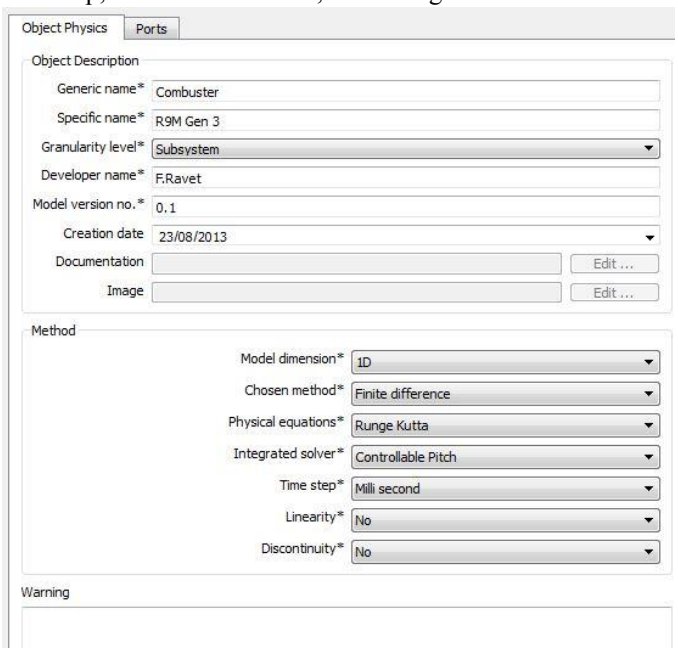


Fig. 10. GUI MIC Object Physics description1

Step 3: *Correctness Control at the Early Design Phase*

We define correctness as a measurement of how well the system design meets or exceeds its requirements. Due to uncertainty arising from design abstractions as well as system interactions with the environment and its users, correctness is specified over multiple dimensions. These dimensions include the design hierarchy (i.e., subsystems and components), environment, use conditions, functions, functional failures, and adherence to design rules. After having established the domain level signals (interfaces and domain level model specifications), the model architects integrate these sub models into a full virtual prototype. The model architect can evaluate alternative architectures against different model accuracy constraints. She or he has to detect any potential problem before the IVVQ (Integration, Verification, Validation, and Qualification) phase. The virtual prototype contains various correctness checks for interoperability problems such as domain models software names, versions, models' min/max values, units, the direction of acausal connections, models' accuracy levels, etc [see Fig. 11].

In the literature there are some related works such as the early stage virtual prototype verification and validation. To address this issue, Van der Velden *et al.* [34] recently developed a virtual prototype metric called the Probabilistic Certificate of Correctness which computes the probability that the actual physical prototype will meet its benchmark acceptance tests, based on virtual prototype behavior simulations with known confidence and verified model assumptions. This works

however does not use Probabilistic Certificate of Correctness methods, it checks basically if there is some incoherence in component level models' interfaces.

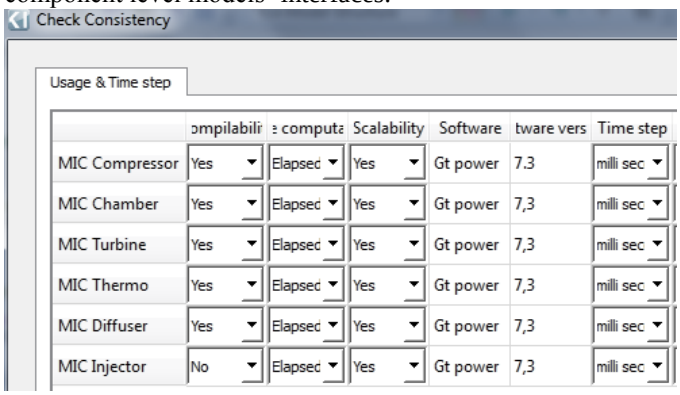


Fig. 11. Interoperability Check

B. Validation Protocol

The validation of the proposed methods consists of two interrelated steps:

(i) scalability of MIC: capacity to cover different nature of analysis models and

(ii) heuristic observation to estimate the rate of model rework and ambiguity reduction

(i) As shown below in Fig. 12, a validation plan is established to test MIC's model specification capacity, proposed tool and the GUI's functionality. To be able to cover different natures of models, we tested MIC with 0D (engine/after treatment, and electric transmission) and 3D (crash) complex model test cases with some selected domain experts.

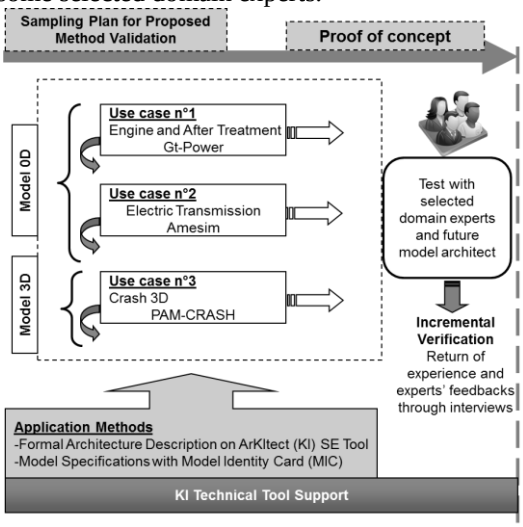
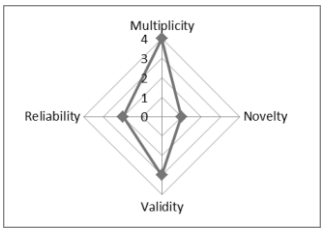


Fig. 12. Proof of Concept

This kind of sampling experimentation is useful to be able to understand the proposed methods' functionality and capacity. Our aim is to make iterations with domain experts in terms of MIC and tool improvement until they arrive at that meets design requirements. Based on return of experience and experts interviews, we can say that MIC is potentially a useful concept which contains ample information about component and system behavior. In addition, they think that robust conceptual designs in an automated fashion in far less time than the manual system engineering approach. Seeing that the usage of simulation models in the other multidisciplinary systems is similar, experts from different companies and domains face also with similar design challenges. Thus, MIC is potentially generalizable concept to other domains outside of automotive engineering such as aeronautics and transport.

(ii) We argue that formal taxonomy or vocabulary for the representation of simulation modeling knowledge is an essential component of ambiguity reduction. Ambiguity arises as multiple interpretations, and interpretations can be understood as hypotheses. Weick [36, 37] introduces the term ambiguity, which he defines as a combination of two underlying terms: equivocality and lack of clarity. Lack of clarity, according to Weick, stems from ignorance, and is similar to uncertainty, which will be reduced by the availability of more information. Equivocality, on the other hand, stems from confusion, where two or more meanings can be assigned to the same cue. As showed in figure 14, today, 4/10 ambiguity problems resulting from multiplicity as variety interpretations of the same things. For example the terms "parameters and uncertainty" have multiples interpretations based on different perceptions towards engineering domains. Resolving equivocality is possible by discarding alternatives interpretations in a collaborative design environment (see Fig. 13 and 14).



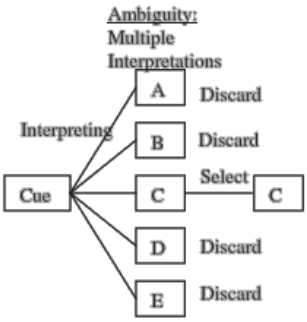


Fig. 13. Sources of ambiguity
Fig. 14. Ambiguity reduction [36]

Sharing a common vocabulary between the designers and manufacturers allows both sides to identify potential misunderstanding problems before they start. Another contribution of this work is reducing the number of the rework while detecting potential inconsistency problems at early design stage. Avoidable rework consumes a large part of development projects, i.e. 20-80 percent depending on the maturity of the organization and the complexity of the products. Therefore, typical rework anomalies may be classified such as

• **avoidable rework** which consists of the effort spent on fixing difficulties that could have been discovered earlier or avoided altogether. In M&S context, most of rework anomalies are caused by interface consistencies and software and hardware versioning problems (see Fig. 15). These anomalies would be detected by correctness control at the early design phase (see section 4 for case study). And

• **unavoidable rework** is work that could not have been avoided because the developers were not aware of or could not

foresee the change when developing the software, e.g. changed user requirements or environmental constraints.

Today, 4/10 rework anomalies are caused by inconsistent interfaces values and hardware and software versioning mismatch problems which are potentially avoidable rework anomalies. Since each defect finds after the product was released to OEM, these reworks create on average, 2 or 3 supplementary staff work per project and 1 to 2 months of delay. Early correctness control aims to reduce the number of these anomalies by a factor of 2. With the provided method, it would take approximately less than one staff hour of correctness check time for each defect found (see Fig. 15).

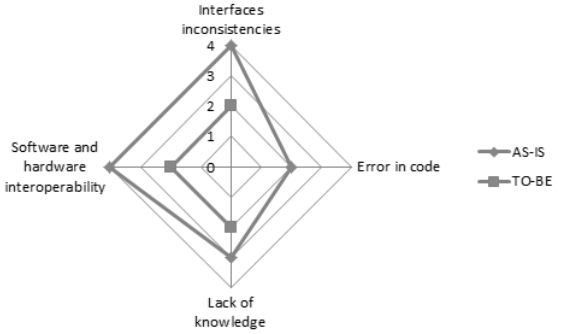


Fig. 15. Sources of rework and expected improvement

V. LIMITATIONS AND FUTURE CONSIDERATION

The overarching goal is to manage the creation of full-vehicle system model by integrating associated domain models to deliver a viable model in less time. To meet this target, this work presents two approaches. First; utilizing architecture based model design phase at the early stage of model development process and the second is to standardize the engineering knowledge transfer thanks to Model Identity Card (MIC). The first characteristic desired for an effective approach is that model architecture integration should be orchestrated by an actor (i.e., Model Architect) with a more formal manner in a precise process.

In this process, we introduce as a novel approach, a detailed design phase with interface consistency checking before the IVVQ (Integration, Verification, Validation, and Qualification) phase. The second desired characteristic is to create M&S common vocabulary and its integration to arKItect systems engineering tool for capturing and sharing engineering domain knowledge between OEMs and model providers. To be able to facilitate editing of model characterization, we created a GUI based on MIC attributes. These techniques described in this paper allow M&S stakeholders' to collaborate quickly and easily.

In the literature, there are some similar works related to numerical model capturing, reuse, characterization and integration but to the best of our knowledge, the development of a systematic process and a complete and detailed M&S vocabulary was not exist.

To demonstrate the applicability of our proposed solution, engine-after treatment, crash and electric transmission models are tested. Engine-after treatment model is explained as a case study. During this case study observation, models' stakeholders (external/internal provider) have participated to the test scenario. Within model development process, the model and the system architects, characterize and create well-defined on-demand model requests. Based on this model request, the model provider selects or creates the model that (1) is appropriate for their desired simulation context and (2) represents the assumptions and limitations of the model. Thus, according to engineers' return of experience based on case study, the knowledge gap between model architect and provider is decreased by providing M&S common vocabulary. Also, knowledge capturing and understanding about numerical modeling is increased. The MIC is locally integrated to the company and tested by different engineering teams. Following the test results, we say that MIC's attributes are accurate and containing sufficient information for characterization different nature of models (0D reduced, and 1D, 2D and 3D).

MIC is applicable to another context such as aeronautic but it requires more work, and thus it is important to extend it to support different specific domains of interest. Model Development Process and MIC concepts will be used in next generation Company's multidisciplinary vehicle modeling strategy. Future works include (1) increasing the number of MIC tests with different engineering teams for testing its capacity (e.g., HVAC and Battery Aging Model are envisaged) and to extend its usage to support different specific domains of interest, (2) an extended validation protocol for proposed concepts in terms of value addition to company's current situation. (3) We would like to also complete the Vehicle Reference Architecture to be modeled by using mentioned tool and methods and finally (4) we would like to align of different views and viewpoints in the same tool.

A long-term vision is to integrate semantically rich domain model libraries and model of behavioral intention [35] concept to our MIC to be able to increase the probability to get the right model from supplier.

ACKNOWLEDGMENT

The authors thank Renault SAS Techno-Centre MIC workgroup especially to Fréderic Ravet, and Marija Jankovic, assistant professor at Ecole Centrale Paris for valuable discussion.

REFERENCES

[1] G. Sirin, B. Yannou, E. Coatanéa and E. Landel, "Analysis of the simulation system in an automotive development project". Int. Conf. of Complex Systems Design & Management CSDM, Paris, France, Dec, 12-14, 2012.

[2] J.M. Branscomb, C.J.J. Paredis, J. Che, and M.J. Jennings, "Supporting Multidisciplinary Vehicle Analysis Using a Vehicle Reference Architecture Model in SysML". Systems Engineering Research CSER, Atlanta, US-GA, March, 12-22, 2013.

[3] G. Sirin, B. Yannou, E. Coatanéa and E. Landel, "Creating a Domain Ontology to Support the Numerical Models Exchange between Suppliers and Users in a Complex System Design". ASME (IDETC/CIE), Portland, USA, Aug., 4-7, 2013

[4] C.J.J. Paredis, A. Diaz-calderon, R. Sinha and P.K. Khosla, (2001, Apr). Composable models for simulation-based design. Engineering with Computers Journal, Vol.17, No. 2, p. 112-128.

[5] L. Yilmaz. (2004). On the Need for Contextualized Introspective Models to Improve Reuse and Composability of Defense Simulations, JDMS Volume 1, Number 3.
[6] R.J. Malak, and C.J.J. Paredis, (2007). Validating Behavioral Models for Reuse. Research in Engineering Design, Vol.18, No.3, p.111-128.
[7] Systems Engineering Fundamentals, DOD Supplementary text prepared by the defense acquisition university press fort Belvoir, Virginia 22060-5565, USA, 2001, http://123management.nl/0/070_methode/072_kwaliteit/Dod%20Systems%20Engineering.pdf
[8] O. De Weck, "Feasibility of a 5x Speedup in System Development due to Meta Design". ASME/DETC2012-70791, 2012.
[9] D. Ward, S. Helton, and G. Polari, (2011). ROI Estimates from SAVI's Feasibility Demonstration, Systems Engineering Conference.
[10] G. Tassey, S.B. Brunnermeier, et S.A Martin. (1999). Interoperability cost analysis of the US automotive supply chain. Research Triangle Institute Final Report. RTI Project 7007-03, Available: https://www.rti.org/pubs/US_Automotive.pdf
[11] C. Belton, "Vehicle Model Architecture for Vehicle System Control Design". SAE World Congress & Exhibition, Detroit, Michigan, 2003.
[12] E. H. Miller, "A note on reflector arrays," *IEEE Trans. Antennas Propagat*., to be published.
[13] G. Mocko, R.J. Malak, C.J.J. Paredis, and R. Peak, "A Knowledge Repository for Behavioral Models in Engineering Design". 24th ASME Computers and Information in Engineering Conference, Salt Lake City, UT, Sept, 2004.
[14] AUTOSAR, www.autosar.org.
[15] Belton, "Vehicle Model Architecture for Vehicle System Control Design". SAE World Congress & Exhibition, Detroit, Michigan, 2003.
[16] M. Pratt, Introduction to ISO 10303—the STEP standard for product data exchange, J. Comput. Inf. Sci. Eng. 1 (1) (2001) 102.
[17] W. Verhagen, R. Curran, Ontological modelling of the aerospace composite manufacturing domain, in: 18th ISPE International Conference on Concurrent Engineering, 2011, Advanced Concurrent Engineering, 2011, pp. 215–222.
[18] S. Ahmed, S. Kim, K. Wallace, A methodology for creating ontologies for engineering design, in: Proceedings of the ASME IDETC/CIE Conference, Long Beach, CA, 2005.
[19] Z. Li, M. Yang, K. Ramani, A methodology for engineering ontology acquisition and validation, Artif. Intell. Eng. Des., Anal. Manuf, Special Issue on Developing and Using Engineering Ontologies (23) (2009) 37–51.
[20] I. Horváth, J. Vergeest, G. Kuczogi, Development and Application of Design Concept Ontologies for Contextual Conceptualization, Delft University of Technology, 1998.
[21] B. Chandrasekaran, J. Josephson, V. Benjamins, What are ontologies, and why do we need them?, IEEE Intell Syst. Appl. 14 (1) (1999).
[22] D. Fensel, C. Bussler, Y. Ding, V. Kartseva, M. Klein, M. Korotkiy, B. Omelayenko, R. Siebes, Semantic Web Application Areas, 2002. <http://www.few.vu.nl/~ronny/work/NLDB02.pdf> (retrieved 17.09.12).
[23] I.R. Grosse, J.M. Milton–Benoit and J.C. Wileden (Juil, 2005) Ontologies for supporting simulation models. AI EDAM, Vol.19, No.1 p.1-18.
[24] Noy, F. N. and McGuinness, (2001) Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. Stanford, CA.
[25] P. Failer. (2010). SAE AADL V2, An overview. Software Engineering Institute Carnegie Mellon University Pittsburgh, Available: https://wiki.sei.cmu.edu/aadl/images/7/73/AADLV2Overview-AADLUserDay-Feb_2010.pdf
[26] B.P. Zeigler *et al*., The DEVS Environment for High-Performance Modeling and Simulation. IEEE C S & E, 1997. 4(3): p. 61-71.
[27] S.R. Blattnig *et al*, Towards a Credibility Assessment of Models and Simulations. 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials, At Schaumburg, IL, 2008.
[28] Delbecq, Andre L., and Van de Ven, Andrew H. (1971), "A Group Process Model for Problem Identification and Program Planning," Journal of Applied Behavioral Science, 7, 4.
[29] D. Karnopp and R.C. Rosenberg. (1968). Analysis and Simulation of multiport Systems. The Bond Graph approach to physical systems dynamics, MIT Press.
[30] S.D. Eppinger, and V. Salminen, "Patterns of Product Development Interactions". 13th International Conference on Engineering Design ICED, Aug., 21-23, Glasgow, Scotland, 2001.
[31] Nasa Technical Standard (2008), Standard for models and simulations National Aeronautics and Space Administration Washington, DC 20546-0001, NASA-STD-7009.
[32] S. Halbach, P. Sharer, S. Pagerit, C. Folkerts, A. Rousseau, "Model Architecture, Methods, and Interfaces for Efficient Math-Based Design and Simulation of Automotive Control Systems", Argonne National Laboratory, 2010.
[33] V. Basili and H.D. Rombach. (1990). Towards A Comprehensive Framework for Reuse: Model-based Reuse Characterization Schemes. University of Maryland, CS-TR-2446, UMIACS-TR-90-47.
[34] A. Van der Velden, P. Koch, S. Devanathan, J. Haan, D. Naehring and D. Fox, "Probabilistic Certificate of Correctness for Cyber Physical Systems", Volume 2: 32nd Computers and Information in Engineering Conference, Parts A and B Chicago, Illinois, USA, August 12–15, 2012.
[35] F. Retho, H. Smaoui, J.C. Vannier, P. Dessante, A model-based method to support complex system design via systems interactions analysis. Int. Conf. of Complex Systems Design & Management CSDM, Paris, France, Dec, 12-14, 2013.
[36] *E. Brun and A.S. Saetre*, Ambiguity Reduction in New Product Development Projects. International Journal of Innovation Management Vol. 12, No. 4 (Dec. 2008) pp. 573–596.
[37] Weick, K (1995). Sensemaking in Organizations. Thousand Oaks, CA: Sage Publications.

Göknur Sirin received the B.S. degree in Computer Engineering from the Galatasaray University Istanbul (Turkey), and an M.S. in Management and Information & Knowledge Systems Engineering from IAE de Paris Sorbonne (France). Since 2012, she has been an Industrial PhD candidate in Ecole Centrale Paris and in Renault SAS.

Chris Paredis is a Professor and Woodruff Faculty Fellow in the G.W. Woodruff School of Mechanical Engineering at Georgia Tech. He has an M.S. degree in Mechanical Engineering from the Catholic University of Leuven (Belgium), and an M.S. and Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University. He received the 2007 CETL/BP Junior Faculty Teaching Excellence Award, the 2007 SAE Ralph R. Teetor Educational Award, and the 2011 ASME CIE Excellence in Research Award. In 2007-2008, he was the Chair of the ASME Computers and Information in Engineering (CIE) Division.

Bernard Yannou is a Professor of Industrial and Mechanical Engineering, deputy director of the Industrial Engineering Laboratory of Ecole Centrale Paris, France, where he manages the Design Engineering Team. His area of expertise is design engineering, more specifically: design automation, artificial intelligence in design, innovation engineering, and design under uncertainty, decision-based design, ecodesign, design optimisation, design processes and organisation. He is member of the Advisory Board of the Design Society, member of the ASME and Associate Editor of Journal of Mechanical Design.

Eric Coatanéa is a Professor in Engineering Design, and manufacturing in Aalto University Finland. He received a PhD (2005) from TKK Finland. Eric has been managing a research group, researching and teaching Engineering design and production for 20 years both in France and Finland.

Eric Landel is an expert leader in Numerical Modelling & Simulation department in Renault Company. He received his MSc in Mechanical Engineering and a PhD in Naval hydrodynamics from Ecole Centrale Nantes (ECN). He has conducted a lot of research teams and industrial projects for a number of industrial companies: REVA, RENAULT, PSA, DCN, IFP. He is the president of the Simulation Section SIA since 2008.