A MULTIMODAL APPROACH TO SIMULATION FIDELITY

Patrice Thebault[(a)], Sangeeth saagar Ponnusamy[(b)], Vincent Albert[(c)]

[(a),(b)]Airbus Operations SAS, 316 Route de Bayonne, Toulouse-31060, France
[(b),(c)]CNRS, LAAS, 7 Avenue Colonel de Roche, Toulouse-31400, France

[(a)]patrice.thebault@airbus.com, [(b)]sangeeth-saagar.ponnusamy@airbus.com, [(c)]valbert@laas.fr

ABSTRACT

A process oriented view of a multimodal approach to simulation fidelity is discussed in the context of system verification and validation in industry. An overview of classical simulation product development is given and the problem of developing models through the inclusion of simulation objectives is presented in terms of this process. A multi modal approach for improving the syntactic fidelity of simulation by using meta-modeling techniques and semantic web principles is presented in an operational context. The domain model is briefly presented and a concept of operating mode is proposed in the model teleological framework of Structure, Behavior, Function and Interface. The relation with the study of formal techniques on semantic or behavioral fidelity is briefly discussed along with an overview on the challenges ahead and future work.

Keywords: *modeling, simulation, abstraction, process, verification and validation, fidelity, ontology*

1. INTRODUCTION

Modeling & Simulation (M&S) is being widely used as a means to conceive, develop and test complex engineering systems. In using M&S as a means for the system Verification & Validation (V&V), often the difficulty is finding and implementing abstractions of the system being simulated with respect to the simulation requirements. Such difficulties gives rise to the problem of simulation fidelity i.e. the effectiveness of simulation in reproducing the reality. However in system validation through simulation, this fidelity requirement is seldom expressed even if the context of use is well known. The effectiveness of simulation in reproducing the reality i.e. realism of simulation, motivates an important question of how to quantify the distance between a system and its simulation with respect to its V&V objectives along the product development cycle? This Paper briefly addresses this problem of fidelity in simulation for system V&V in the industrial context.

The paper is structured as follows; the concept of simulation fidelity is explained in the context of system V&V followed by a brief description of simulation product development in industry. A process oriented view on improving the syntactic fidelity of simulation by using meta-modeling techniques and semantic web principles with some illustrative concepts is presented. The relation with the study of formal techniques on semantic or behavioral fidelity is briefly presented along with a discussion on challenges ahead and future work.

2. SIMULATION IN SYSTEM V&V

In the context of systems engineering, Verification and Validation determines the compliance of simulation products with their specifications and fitness for their intended use respectively. V&V activities are usually illustrated in the classical V cycle as seen in figure 1 and this cycle can be broadly classified into two parts. The left branch of the cycle corresponds to design V&V where the system is virtual i.e. under construction and the right branch corresponds to product V&V where the system is physical i.e. built. Simulation is used to perform V&V of the specification and the design of the System Under Test (SUT) in the design verification phase, and of the integrated configuration and the SUT operational environment in extreme conditions, such as failure, in the product verification phase.

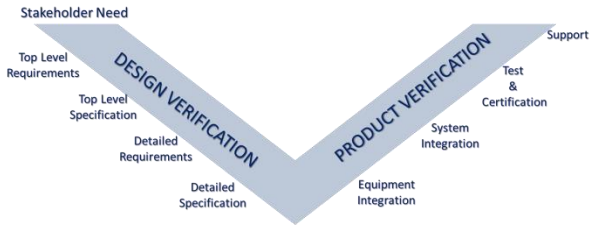

Figure 1 : System V cycle

Simulation is being increasingly used in complex system development due to its significant advantages in cost, time, safety, design tradeoff etc. However, as the systems are getting more complex so do the M&S activities. Even with the advent of powerful computing resources, the sheer complexity of phenomena to be modeled in addition to non-technical factors such as lack of rigorous and standardized process makes M&S activities challenging. There is neither an agreed standard to define or measure this complexity of model, nor a methodology for model developer to choose it [Brooks,1996].

The confluence of control, communication and computing results in cyber physical systems which are becoming ubiquitous especially in transportation systems. In V&V of avionics systems, a subclass of cyber physical systems, different modeling paradigms are being used and it is important to incorporate a multimodal approach in M&S of such systems. In addition, there is a problem of heterogeneity due to different modeling formalisms used by different stakeholders. It leads to interoperability issues especially the during model integration phase. Since simulation itself is a complex product whose development involves multiple stakeholders, resources such as IT, Platform etc., a rigorous Model Based System Engineering (MBSE) approach is needed to seamlessly develop and deploy simulation products for system V&V. The current paper is focused on an engineering process perspective on modeling and it serves as a compliment to [Ponnusamy,2014], where an MBSE approach in model assembly, integration and simulation was discussed in the context of avionic system V&V. In [Ponnusamy,2014], meta-modeling techniques were presented to develop a standard framework for information interchange and for making the domain assumptions explicit through the use of ontologies.

2.1. Simulation Fidelity

A general overview of this fidelity problem and a proposition for its formal assessment based on the structural and behavioral aspects is addressed in [Ponnusamy,2014] & [Ponnusamy,2015]. Ascribing a distance notion to the effectiveness of model in reproducing the required reality, fidelity is defined as 'distance to reality'. It has been long acknowledged in the M&S community that models are usually verified but seldom validated rigorously. In other words, the context of usage is not explicitly taken into account during the modeling process. Consider figure 2, where system V&V through simulation process is illustrated with associated stakeholders.

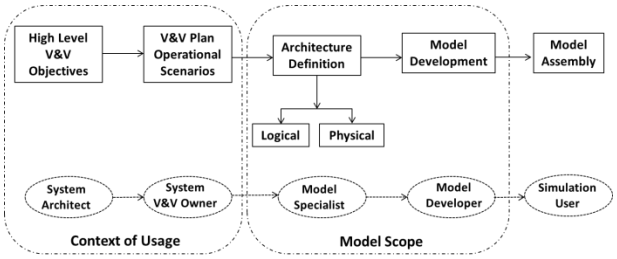

Figure 2 : Simulation Model Development Overview

A top down development of system commences with requirement collection from customers and refined to a system development and its V&V schedule by the programme management. System architect defines overall architecture and high level V&V objectives for the integrated system i.e. System of System (SoS). This drives the individual system development and the corresponding V&V plan definition i.e. operational scenario. This system definition by system designer and its V&V plan by simulation user i.e. system V&V owner serves as input for model specialist who in turn defines the simulation architecture, both physical and logical. The model functional and performance requirements drive the model development which when developed will be assembled and simulated according to the V&V plan by the simulation user. It can be seen clearly that despite the obvious need to integrate the context of usage into model development such that the model scope includes the intended usage, this proves to be a challenging task in both technical and non-technical perspectives. The distance between the model scope and its intended usage gives rise to fidelity and this phenomenon is further classified [Ponnusamy, 2014] and explained below.

2.1.1. Fidelity Classification

Fidelity can be classified in a myriad ways [Roza,1999] and in this section it is given from a process oriented view. It is classified into designed and measured fidelity which is illustrated in the following figure,

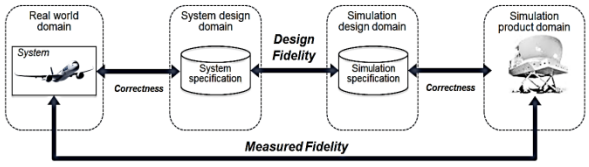

Figure 3: Design & Measured Fidelity

The classical model development process is essentially an iterative process based on measured fidelity approach due to the challenges in complexity, methodology and continuous evolution of product requirements. A paradigm shift to a design fidelity approach where the modeling process is driven by the associated validity requirements will help in better managing the fidelity.

In discussing model or simulation fidelity, the challenges are broadly classified into four groups namely, define fidelity, capture fidelity needs, manage and implement fidelity. The challenges are illustrated on the simulation product development process.

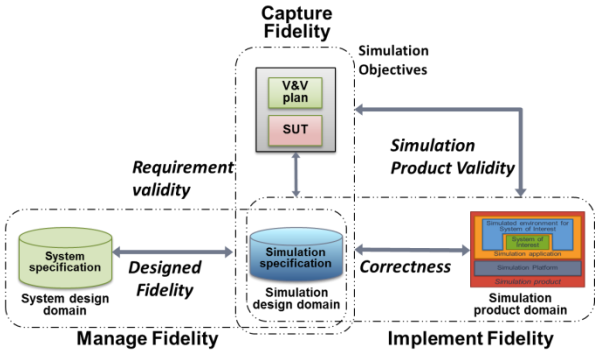

Figure 4: Fidelity Challenges

It can be seen that the requirement validity is defined between the simulation objectives given by the V&V plan over the SUT and the simulation specification, whereas the simulation product validity is defined with respect to simulation product. Similarly the concept of designed fidelity can be seen. The focus of this paper is on capture and manages fidelity aspects as fidelity definitions are already discussed in the preceding section and implement fidelity is arguably a verification problem. In the following section the simulation product [Thebault,2014] and its development process is explained in detail.

3. V&V BY SIMULATION: A PROCESS ORIENTED VIEW

A brief description of the simulation product is given followed by the development process and its relation with the problem of fidelity.

3.1. Simulation Product

A *simulation product* is generally described as a simulation application deployed on a simulation platform, and interfaced with the system of interest. In other words, a simulation product is akin to an Experimental Frame (EF) notion proposed by Zeigler [Zeigler,2000]. The EF defines the controllability and the observability means to stimulate and observe the model behavior in addition to the conditions of experimentation [Traore,2006]. The application of this standard M&S notion in the context of modeling abstractions were discussed in [Ponnusamy,2014] and [Albert,2009].

In general, a simulation product is illustrated in the following figure 5. The interface with system of interest is shown in green.

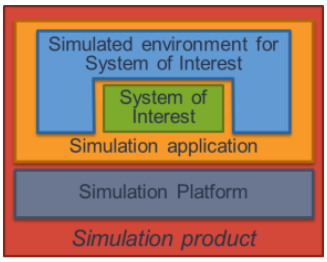

Figure 5: Simulation product architecture

In order to enable such an architecture representation, an internal standard commonly exists in the industry to define a common understanding on how the simulator platform shall execute the simulation application. The simulation application development is based on the knowledge of the operational environment of the system of interest, which is, in the avionics context, composed of equipment whose behaviour is governed by physical laws such as aircraft natural dynamics and other avionic systems.

The *simulation application* comprises a set of standard simulation models and associated configuration files which specify the connections between models, and their scheduling properties.

The *simulation platform* usually consists of an IT infrastructure and the simulation software. The platform schedules and monitors the execution of the models with respect to time constraints of logical or real time simulation. It enables communication between the models and provides the end user with control and observation facilities to operate the simulation [Thebault,2014].

3.2. Simulation Product Development

A simulation product needs to be updated continuously to follow each high level design change and also new simulation capabilities for V&V objectives to the end user. In general, this simulation application development process is performed by simulation platform teams, who consistently interact with the component system developers and simulation users. The simulation application development process is briefly illustrated in the following figure,

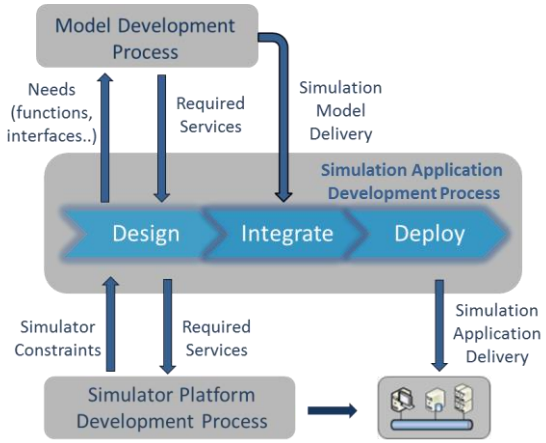

Figure 6: Simulation product development process

It can be seen that the process can be broadly classified into design, integrate and deploy. In the 'design' phase, functional and performance objectives of the simulation models are defined in addition to their interface definition. The second phase, 'Integrate', is model assembly phase to ensure the consistency for its 'deployment' on the simulation platform.

In [Thebault,2014], in a Model Driven Engineering (MDE) framework using SysML, this consistency in 'integrate' and 'deploy' phase are discussed. However, owing to the complex nature of this process, the context of usage is not always captured in accordance with modeling abstraction employed in the 'design' phase. This paper deals with consistency in the 'design' phase using similar such domain model techniques. In the design phase i.e. modeling, models are developed using system knowledge and its context of usage by model developers. This model development usually involves three stakeholders namely model developer, system designer and simulation user. The simulation user is usually the V&V task owner who defines the simulation requirements derived in turn from the high level V&V objectives in terms of functional, non-functional and

behavioral requirements. In practice, there is a model specialist who translates simulation requirements and system specification into functional and behavioral requirements of the model to be built. The model developer builds the model based on this requirement using knowledge from existing library of abstractions. The built model assembled and then verified against their requirements by simulation according to the V&V plan. The following section briefly outlines the challenges in selecting modeling abstractions employed in the process which gives rise to the fidelity problem.

3.2.1. Challenges in Modeling Abstractions

A model is essentially an abstraction of reality it is intended to represent and formally it is defined by Brade [Brade,2004] as follows, A model is an abstract and idealized representation of a real system, which reflects all the relevant properties with sufficient accuracy with respect to its intended purpose. The key element in this definition is abstraction with respect to its intended purpose. A model must be built such that it is fit for the intended usage, in other words, it must be valid [Traoré,2006]. Owing to the fact that most of the models are rigorously verified but seldom validated, the onus must be on inclusion on validation objections a priori in model building process. However, there exists no agreed standard or a guideline [Brooks,1996] to choose the level of model complexity owing to the innate nature of problem in quantifying this complexity i.e. abstraction level vis à vis model performance. Some of the other challenges in modeling abstractions are lack of common understanding between the stakeholders, semantic inconsistency, and interoperability [Benjamin,2009].

In addition to the inherent technical complexity of modeling, there exists non-technical complexity in the development of complex systems such as constraints on the time, cost, human resources and infrastructure. In an industrial context, each component systems are developed by different teams often working transversely and transnationally. These component systems interact with each other to perform a Multi System Function (MSF) in an integrated system. For example, '*calculate and display aircraft position*' function is performed by GPS and inertial data system and communicated to Cockpit Display System (CDS) for the pilot's view. However, identification of functional contribution of each such system to MSF could be difficult and this equally true for identification of simulation requirements. Thus, formalization of such system functions and test objectives is often a tedious task for the system designer and simulation user respectively.

This is compounded due to the lack of a consistent derivation of low level V&V requirements from high level V&V objectives as illustrated in the figure 2. The requirements traceability between these two domains is seldom one-to-one and the inclusions of low level requirements in the high level requirements are traditionally managed by heuristics, domain expertise, margins and experience. This issue permeates down to all phases of the V cycle and often the model architect who defines abstraction rules to be implemented by the model developer faces a difficult choice of selecting abstractions consistent with its V&V objectives. This not only leads to model validity problems at the simulation runtime but also to over specification and sub optimal design and development. These challenges necessitates a domain model approach which mitigates such problems and enable a common understanding by making domain assumptions explicit and separate domain knowledge from the operational knowledge. Ontologies, in addition to classical MBSE tool such as SysML, serve as a good candidate due to their standardization in terms of OWL language, scalability, query capability and availability of tools such as Protégé[1]. However, ontologies have limitations in capturing the dynamics of reactive systems and a fidelity approach will only be complete if the fidelity requirements usually expressed as tolerances over desired behavior are adequately captured in modeling and this *semantic* i.e. behavioral perspective [Ponnusamy,2014] & [Ponnusamy,2015] is given by approximate bisimulation [Girard,2007]. Such a multimodal approach is discussed in the following section with emphasis on model teleological view in terms of operating modes ontology.

4. A MULTIMODAL APPROACH TO MODELING

A simulation product development involves requirement collection, conceptual modeling, model formulation, model construction, assembly and deployment on the platform. The simulation involves experimentation according to V&V plan, data collection, analysis and conclusion. All such activities involve multiple levels of abstraction, stakeholders, formalisms and tools. A single approach to tackle this abstraction consistency problem is neither feasible nor practical and a multi modal approach is needed. This problem is broadly classified on structural and behavioral aspects and methodologies are discussed in [Ponnusamy,2015]. In [Fishwick,1993], a multimodal approach to simulation is discussed in terms of reasoning, functional modeling, qualitative modeling and visuo-spatial reasoning. However in the current paper, the focus is on simulation domain ontology spanning all steps of simulation product development in the context of capture and manages fidelity as described in section 2.1.1. The ontologies based on the model teleological and theory of M&S concepts developed in OWL using the standard Protégé tool is presented in the following section. This is used for collecting and exploitation of system knowledge and context of usage by the model architect to formalize structural consistency aspects of models such as architecture, interface, data type, quantities, units etc.

A significant advantage of such an approach besides providing a standardized information exchange

[1] http://protege.stanford.edu/

platform is its reasoning and query ability [Jenkins,2012]. They could be important in the industrial context as the information is incrementally and simultaneously being updated by different stakeholders; many linked concepts could be identified, classified and checked upon to ensure consistency all along the product development.

4.1. M&S Ontology: Operational view

In an industrial setting an essential prerequisite for any method proposed to improve the model fidelity is that it must be amenable for integration into the system development process and also need to be user friendly for the practicing engineers. It is thus important to illustrate how the proposed method will be operated and quantify its effect on the 'as is' process. In this context, the operational perspective of the proposed domain model is presented in the figure 7.

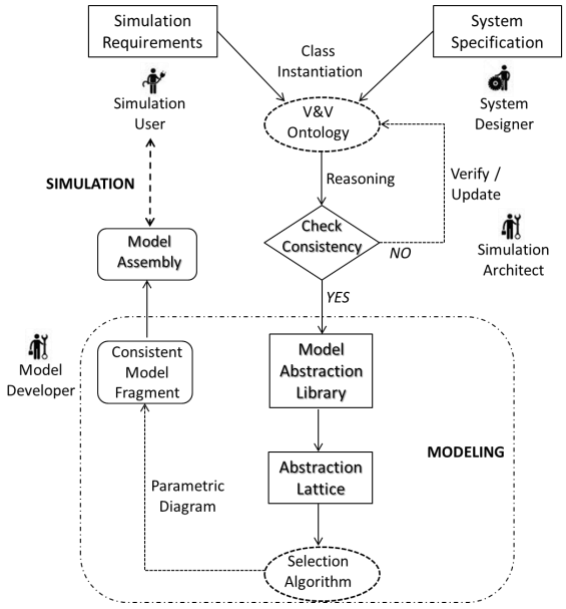

Figure 7: Operational View of M&S domain model

It may be seen that a single ontology is instantiated by the user and designer separately whose consistency and inclusion relations are ascertained by the model specialist. Here consistency refers to classical consistency checks such as incorrect instantiation, constraint violation etc. Inclusions relations are better understood through partial order relations defining a finite lattice [Lickly,2011]. The consistent lattice is input to the selection algorithm [Levy,1997], which selects the consistent yet simplest model abstraction. This selection through a parametric diagram is built in a classical simulation tool such as Modelica or Matlab SIMULINK. All such consistent models are assembled by the user into one integrated simulation application deployable on a platform and then simulated. The reasoning capabilities of ontology are exploited in conjunction with query capabilities to perform theses activities. It may be noted that the classical but often tedious document and discussion centric process used by model specialist as discussed in section 3.2 is being replaced by the domain model approach. Some of the important concepts of this domain model approach are briefly discussed in the following section for better understanding of this process vis à vis V&V activities.

4.2. M&S Domain Model – A Syntactic View

There have been various studies on the need for a domain model approach in system development [Zayas,2010], [Jenkins,2012] and the need for shared domain knowledge, i.e. ontologies, which is usually mastered by the engineers but not formalized. In addition, modeling as a reasoning problem was posed by et al in [Levy,1997] since a model developer reasons about a given physical system at different levels of abstraction. In the field of Artificial Intelligence (AI), qualitative simulation has been proposed by Kuipers et al which is based on qualitative reasoning about systems [Kuipers,2001]. Similarly, in the state of the art ontology document [CRYSTAL,2014] of the CRYSTAL project, five different levels of abstraction levels are proposed namely, operational, functional or non-functional, logical, physical and implementation and discussed in the context of engineering phases and viewpoints.

A domain model for M&S needs to incorporate all such possible paradigms of modeling concepts and in [Ponnusamy,2015], a domain model of modeling abstractions is proposed on four axes of scope, computation, data and time in and instantiated with model fragments data. This ontology was built using Protégé and the reasoning capabilities of ontology were exploited to build and fill the model abstraction library. An algorithm based on [Levy,1997], has been implemented as SysML activity diagram to select an abstraction consistent with requirements from this library. In discussing this preliminary attempt at an automated model selection approach, the need to be coherent with behavioral modeling and fidelity needs capture has been emphasized in it. This paper attempts to address this problem from the context of industry. In the following section, the initial version of ontology based on Structure, Behavior, Function (SBF) framework of [Gero,2004] is extended with Operational modes concept and discussed. This operational modes ontology proposed here corresponds to operational abstraction and is linked with behavioral and functional abstraction.

4.2.1. SBFIO Framework

The notion of Structure, Behavior & Function, originally proposed in the context of design studies has been used in system engineering studies as well for better understanding and decomposition of complex systems [Gero,2004]. In broader terms, Structure refers to the architectural notion of the system whereas Behavior refers to dynamic response of the system under input stimuli. Function in our context is the purpose or goal of the system i.e. availability of a service. In addition to SBF, the notation of interface (I)

and Operating mode (O) are more pertinent for interconnected system description and different ways of operation respectively. Interfaces can be interpreted as ports of exchange, energy and data, between physical and cyber systems respectively. The following section briefly illustrates the Operating mode concept of the SBFIO framework. For fidelity approach for other concepts the readers are encouraged to refer to [Ponnusamy,2015].

The mapping of SBFIO framework is given as follows,

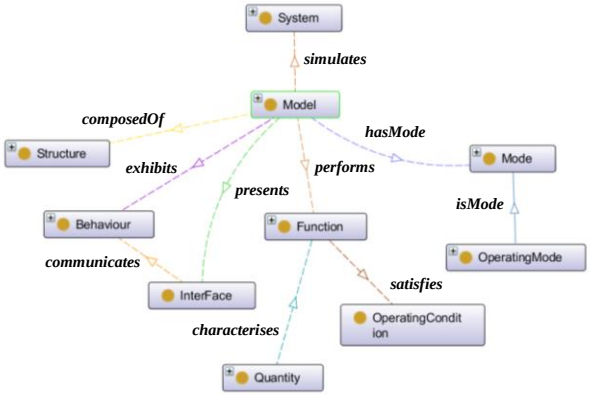

Figure 8: SBFIO Framework

It must be noted that not all the concepts and relations are explicitly given in the figure and this ontology is only a teleological view on models and it does not discuss ontology of modeling formalism, class of abstractions etc. [Albert, 2009], [Ponnusamy,2015].

4.3. Operating Modes &Transitions

A system may exhibit different ways of operation called *mode* and operating mode typically refers to such a phenomenon in the context of interconnected systems where one mode causally affect the other. A simple example would be light switch which could be either 'on' or 'off' connected to a light bulb which could be 'glow' or 'dark' depending on the mode of the switch. This mode definition could be physical or logical and it can be ascribed to logical architecture i.e. system or physical architecture i.e. equipment. Denoting such architecture constituents by a generic notion called '*block*', an operating mode could be associated to a block's modes of operation. In general, a block can function in different modes and for each mode and their combination, there exists an associated functionality and behavior. Two interconnected blocks essentially implies interconnection of their respective modes. The mode of the source component is called guard and the destination component is called mode. Together, the pair of causative and resulting mode, along with their blocks forms an operational mode. The interest of such a description is the correspondence with their functionality and behavior. It helps to envisage the modes of operation, its complexity causality, and inter dependencies.

This definition of mode as a control information of the system operation is similar to the concept of mode charts proposed in [Jahanian,1994]. Mode charts is a specification language for real time systems whose semantics are given by Real Time Logic (RTL) [Jahanian,1994], a logic for reasoning about the absolute timing of events. Our definition of operational mode is akin to series and parallel mode classification proposed by Jahanian et al, however, in our approach the parent-child modes given by guard-state are formalized as a single mode. This definition is based on simple causality relation i.e. mode A causes mode B and is amenable to ascribe functional behavior or a semantic behavior. A similar notion is mode automata proposed in [Maraninchi,1998], which is essentially an automaton whose states are labeled by dataflow programs. However, in our case, the modes refer to operational manifestation of a block under a given scenario. An analogy with state transition diagram [Jahanian,1994] is when mode could be interpreted as a grouping of states.

Definition 1: Let us denote a block, B_i where *i* is an identifier, exhibiting modes, M_j^i where *j* refers to the numbers of modes. An interconnection between blocks B_i and B_{i+1} and their modes are represented by an operating mode which is defined by the following tuple,

$$OM_n =< M_j^i \rightarrow M_j^{i+1} > \quad (1)$$

where M_j^i, M_j^{i+1} are the connected modes of B_i and B_{i+1} respectively.

The direction arrow defines the causality connection, here it means when B_i is at mode M_j^i then B_{i+1} is at M_j^{i+1}. The tuple $<B_i,M_j^i>$ is called the guard (i.e. source) and $<B_{i+1},M_j^{i+1}>$ is called the state (i.e. destination).

The transition $T_{n\rightarrow n+1}$ defines transition from one operating mode, OM_n to another, $OM_{n+1}$when the guard conditions changes i.e. becomes true denoted by ⊦ symbol. Interpreting OM_n as source operating mode and OM_{n+1} as destination operating mode, then transition occurs when the guard mode of the destination operating mode becomes true i.e. enabled. It is defined by,

$$T_n: M_j^i \times \tau \rightarrow M_{j'}^i \,|\, \tau : M_{j'}^i \vdash \quad (2)$$

where $M_j^i \in OM_n$ and $M_{j'}^i \in OM_{n+1}$

The mode description and its transition is represented in the following figure,

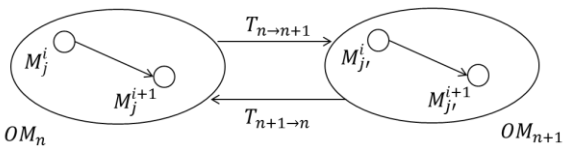

Figure 9: Operating Mode

These modes can be interpreted as reachability space where each state is a possible operating mode of the block and preceding state is the guard operating mode. In other words, when the guard condition is true the entailment relation between these two modes result in final mode. The evolution from guard to the mode involves dynamics and at each mode too there could be an associated behavior. In the next section, a brief discussion on how these concepts are useful in developing an abstraction library is discussed.

4.3.1. Operational Modes Inclusion & Abstraction

An important use of this definition is organizing such operational information of systems into interconnected modes which not only renders a lucid description but also yields information about independent modes which could be abstracted. Modes which have no transition relation with a particular mode i.e. unreachable are independent modes. Such enumeration of transition based on mode definition helps in determining necessary abstractions.

Thus depending on the user requirement on the operational mode of the system, a model specialist can navigate through the mode description given by the system designer to identify and segregate the necessary modes and thereby the associated sub systems or equipment (refer Fig 7). Only the system and the corresponding modes matching with user requirements will be retained. This also enables more autonomy for the modeler, who often has to rely on system designer to identify these mode transitions. A semi-automated way to extract such information will make operational modeling more lean and coherent with simulation objectives

This is akin to an automaton but the only difference being that the concept of time is abstracted. The modes are operational modes with causality relation between them and the detailed semantics of this approach is being formalized. However, for systems based approach, an abstract notion such as in Eq (1) & (2) may suffice and this definition should be seen from the operational context by the user and designer.

An ontology instantiation will help analyze the consistency through inclusion relations [Ponnusamy, 2015] and SPARQL[2] queries identify and extract desired information. For example, the model architect queries states having same guards, in other words, this state could be reached by two different sources and based on the simulation objectives any one branch and its associated block definition can be abstracted. Similar such extensions and analyses are possible and are not discussed here. An analogy for this approach would be fork-join definitions in reachability tree of petri net. In other words, there exists two ways to reach this mode and these two guards can be combined as follows. The semantics and its transformation to other formalism is being formalized and will be subject of other paper whereas the current objective is to illustrate this concept in a process based context of V&V by simulation. An application of these formalism on the V&V of aircraft nacelle anti ice system failure case has been discussed in [Ponnusamy,2016].

4.3.2. Mapping to Behavior – A Semantic View

In a SBFIO framework, these operational modes can be mapped to automata which model the system behavior and this is applicable to hybrid automata defined by invariants, guards and resets as well [Tomlin,2003]. Such behavior can be formally verified by reachability analysis and significant progress has been made in the control community in developing various geometric abstractions such as zonotopes [Girard,2007], polyhedrons etc. to perform reachability analysis over dynamic systems [Stursberg,2003]. In addition to verification, syntheses of abstractions are also studied with the help of approximate bisimulation techniques in [Girard,2007] & [Pappas,2003]. Alternatively, such a model can be executed using a discrete event system (DEVS) simulator such as ProDEVS for which there exists a meta model of execution semantics in SysML [Hung,2015]. This domain model for model execution complements the domain model for model building. Such an integrated domain model approach helps in standardizing M&S activities and thereby improves the overall fidelity. Another approach would be using such definition as a language for a formal system specification.

4.4. A Unified framework

A syntactic approach based on domain modeling and a semantic approach based on bisimulation and its approximations could lead to a unified framework encompassing high level fidelity needs capture to low level implementation. It may be noted that as the simulation product development process progresses, the method and tool used will become more formal and this multi modal approach of using a combination of formal and semiformal techniques will help managing the fidelity of models better. In addition, creation of such repository due to incremental addition of knowledge on modeling, systems and their usage will be a significant value addition for enterprise in terms of knowledge capitalization and reuse.

5. OUTLOOK & CONCLUSION

The preliminary process described above needs to be further developed, automated and integrated with the engineering process with user friendly GUI for instantiations by different actors. The SBFIO ontology is being improved with additional concepts based on naïve and basic physics and other domain specific concepts. The preliminary results demonstrate the flexibility of this approach in archival and exploitation of domain knowledge [Ponnusamy,2016]. However future challenges include integration, management and deployment of this process in the industry. Currently studies are being carried out to demonstrate the

[2] http://www.w3.org/TR/rdf-sparql-query/

feasibility and industrial readiness of this approach through a real aircraft system application case.

Another axis of future work is tool development for integration of SysML and OWL. As remarked by [Greves,2009] and [Jenkins,2012], mutual transformation between SysML and ontology will help engineers to capitalise on their graphical syntax and reasoning capabilities respectively and thereby ensuring seamless design and product V&V activities. In the semantic approach, further work is needed in synthesis of abstractions with respect to fidelity requirements. In [Ponnusamy,2015], a preliminary approach based on bisimilarity preserving surjection maps is presented in the context of experimental frame. However, significant work needs to be done in this direction especially due to the increasing usage of cyber physical systems which exhibit hybrid dynamics where choice of abstraction is crucial in representing their dynamics to sufficient accuracy.

An important area to be addressed in the overall V&V process is the synthesis of requirements. Requirements are usually written in natural language text and unless they are managed by tools such as DOORS, it becomes a tedious task to consistently update, trace or modify the requirement database. An active area of research is to move from informal natural language description to a more semi-formal MBSE approach and in some cases formal description in some temporal logic such as Linear Temporal Logic (LTL) or Signal temporal Logic (STL) etc. In [Ponnusamy,2014], Ponnusamy et al discusses inclusion of simulation objectives described in LTL in modeling. The need to parse requirements into various classes has also been of considerable interest and NLP tools such as TXM[3] aids in some text analysis. However, automated transition from natural text to a class description is far from being done though there have been some initial attempts [Ileiva,2005]. A more formal definition of requirements would enable better rapid prototyping of systems. Such a method will help in coherent model development and deployment from top level requirements capture to low level behavioral modeling by mapping the related concepts at each intermediary step.

REFERENCES

Albert V., 2009. Simulation validity assessment in the context of embedded system design, Thesis (Phd). LAAS-CNRS, University of Toulouse, Unpublished.

Benjamin P., Akella K., 2009. Towards ontology-driven interoperability for simulation-based applications. Proceedings of Winter Simulation Conference, pages 1375-1386, December 13-16, Austin, USA.

Brade D., 2004. VV&A Global Taxonomy (TAXO). Common Validation, Verification and Accreditation Framework for Simulation, REVVA.

Brooks R.J., Tobias A.M., 1996. Choosing the best model: Level of detail, complexity, and model performance. Journal of Mathematical and Computer Modeling, Vol 24 Issue 4, Pages 1-14.

Fishwick P.A., Narayanan N.H., Sticklen J., Bonarini A., 1993. A Multimodal Approach to Reasoning and Simulation. IEEE Transactions on Systems, Man and Cybernetics, Vol 24, No 10.

Frantz F.K., 1995. A taxonomy of model abstraction techniques. Proceedings of the 27th conference on winter simulation, pages 1413-1420, Arlington, Virginia, United States.

Gero J.S., Kannengiesser U., 2004. The situated function-behaviour-structure framework. Design Studies, 25(4), 373–391.

Girard A, Pappas G.J., 2007. Approximate bisimulation relations for constrained linear systems. Automatica, Volume 43 Issue 8, pages 1307-1317.

Girard A, Pappas G.J., 2007. Approximation Metrics for Discrete and Continuous Systems. IEEE Transactions on Automatic Control, Volume 52, Issue 5, pages 782-798.

Gross D., 1999. Report from the Fidelity Implementation Study Group, Simulation Interoperatbility Workshop, USA.

Greves H., 2009. Integrating SysML & OWL. Proceedings of OWL:Experiences and Directions, 2009.

Hung V.L., Foures D., Albert V., 2015. ProDEVS: Simulation and visualization of DEVS specification for hybrid modeling using UML state machine. International Conference on Simulation Tools and Techniques, Athens, Greece, Submitted.

Ilieva M.G., Ormandjieva O., 2005. Automatic Transition of Natural Language Software Requirements Specification into Formal Presentation, Natural Language Processing and Information Systems. Lecture Notes in Computer Science Volume 3513, pp 392-397.

Jahanian F., Mok A.K., 1994. Modechart: a specification language for real-time systems, IEEE Transactions on Software Engineering, vol.20, no.12, pages 933-947. doi: 10.1109/32.368134

Jenkins S., Rouqette N., 2012. Semantically-rigorous systems engineering using SysML and OWL. International Workshop on System & Concurrent Engineering for Space Applications, Lisbon, Portugal.

Kuipers B., 2001. Qualitative Simulation. Encyclopedia of Physical Science and Technology, Third Edition, NY: Academic Press.

Levy A.Y., Iwasaki Y., Fikes R., 1997. Automated model selection for simulation based on relevance reasoning. Artificial Intelligence, Vol 96, Issue 2, 351–394.

Lickly B., Shelton C., Latronico E., Lee E., 2011. A Practical Ontology Framework for Static Model Analysis. Proceedings of the Ninth ACM international conference on Embedded software, NY, USA, pages 23-32.

[3] http://textometrie.ens-lyon.fr/

Maler O., 2002. Control from Computer Science. IFAC Annual Review in Control 26(2), pages 175-187.

Man-Kit-Leung J., Mandl T., Lee E., Latronico E., Shelton C., Tripakis S., Lickly B., 2009. Scalable semantic annotation using lattice based ontologies. Lecture Notes in Computer Science, Vol 5795, pages 393-407.

Maraninchi F., Rémond Y., 1998. Mode-Automata: About Modes and States for Reactive Systems. Proceedings of European Symposium on Programming (ESOP), Lisbon, Portugal.

Ponnusamy S.S., Albert V., Thebault P., 2014. A simulation fidelity assessment framework. International Conference on Simulation and Modeling Methodologies, Technologies and Applications 2014, pages 463-471, Vienne, Austria.

Ponnusamy S.S., Albert V., Thebault P., 2014. Modeling and simulation framework for the inclusion of simulation objectives by abstraction. International Conference on Simulation and Modeling Methodologies, Technologies and Applications 2014, pages 385-394, Vienne, Austria.

Ponnusamy S.S, Albert V., Thebault P., 2015. A Meta-Model for Consistent & Automatic Simulation Model Selection. International Conference on Simulation Tools and Techniques. Athens, Greece, Submitted.

Ponnusamy S.S, Albert V., Thebault P., 2015. Consistent behavioral abstractions of experimental frame. AIAA Modeling & Simulation Technologies Conference. Dallas, USA, Accepted.

Ponnusamy S.S, Thebault P., Albert V., 2016, Towards an ontology driven framework for simulation model development, European congress on Embedded real time software and systems. Toulouse, France, Submitted.

Pappas G.J., 2003. Bisimilar linear systems. Automatica, Vol 39, Issue 12, pages 2035-2047.

Roza M., 1999. Fidelity Requirements Specification: A Process Oriented View. Fall Simulation Interoperability Workshop.

State of the art for Aerospace ontology. 2014. CRYSTAL Project, D209.010.

Stolle R., Bradley E., 1998. Multimodal Reasoning for Automatic Model Construction, Proceedings Fifteenth National Conference on Artificial Intelligence, July, 1998, pages 181-188, Madison, Wisconsin, USA.

Stursberg O., Krogh B.H., 2003. Efficient Representation and Computation of Reachable Sets for Hybrid Systems. Hybrid systems: Computation and Control, Lecture Notes in Computer Science, Vol 2623, pages 482-497.

Thebault P., Suquet T., Duffy M.,Viaud B.,Will P., Cordon J., Lamoliate S., 2014, Engineering of complex avionics systems simulations using a model based approach. Proceedings of Embedded Real-time Software and Systems (ERTS) 2014, Toulouse, France.

Tomlin C.J., Mitchell I., Bayen A., Oishi M., 2003. Computational Techniques for the Verification of Hybrid Systems, Proceedings of the IEEE, Vol. 91, No. 7, pages 986-1001.

Traoré M.K., Muzzy A., 2006. Capturing the dual relationship between simulation models and their context. Simulation Modeling Practice and Theory. 14(2): 126–142.

Wagner D.A., Bennett M.B., Karban R., Rouquette N., Jenkins S., Ingham M., 2012. An ontology for State Analysis: Formalizing the mapping to SysML. IEEE Aerospace Conference, USA.

Zayas D.S., Monceaux A., Ameur Y.A., 2010. Models to Reduce the Gap between Heterogeneous Models: Application to Aircraft Systems Engineering. ICECCS, pages 355-360.

Zeigler B.P., Praehofer H., Tag G.K., 2000. Theory of modeling and simulation, San Diego, California, USA: Academic Press.