TOWARDS A UNIVERSAL REPRESENTATION OF DEVS: USING SEMANTIC MODELS FOR BUILDING A MODELING AND SIMULATION FRAMEWORK

María Julia Blas
Silvio Gonnet

Instituto de Desarrollo y Diseño INGAR
UTN-CONICET

Santa Fe, Argentina
{mariajuliablas, sgonnet}@santafe-conicet.gov.ar

Bernard P. Zeigler

University of Arizona
Tucson, AZ
RTSync Corp.
Chandler AZ, USA
zeigler@rtsync.com

ABSTRACT

DEVS is one of the key formalisms to be included in a M&S framework due to its general systems basis. This paper presents the foundations for building an universal representation of DEVS that employs ontologies as modeling strategy by following the MDE principles. It reviews several conceptual models of DEVS developed over the years in terms of three dimensions (domain, formal specification and implementation) with aims to identify the main concepts to be included in an abstract conceptualization of the formalism. Over such concepts, it introduces the core of the DEVS M&S framework in order to illustrate how conceptual modeling can improve the design, development and implementation of discrete event simulation models by following different modeling perspectives.

Keywords: model-driven engineering, progression of abstraction to implementation, M&S framework, conceptual modeling challenges, DEVS.

1 INTRODUCTION

In the last years, the Modeling and Simulation (M&S) field has grown becoming a discipline by itself. From the modeling perspective, a *simulation model* is similar to a *software system model* because both kinds of model are developed from a conceptual system model (Guizzardi and Wagner 2010). In software engineering, this approach is called *Model-Driven Engineering* (MDE). The MDE approach follows the *Model-Driven Architecture* (MDA) proposal that has been used in the M&S field for several years.

MDE differences three types of models as engineering artifacts resulting from the activities performed in the analysis, design and implementation phases. These models are defined as *domain models*, *platform-independent design models* and *platform-specific implementation models*. Then, a *software system model* consist of a set of models defined in terms of the most important viewpoints of the system in order to crosscutting all three modeling levels (Guizzardi and Wagner 2012). However, in the M&S field there is no common understanding on how applying these modeling levels when building *simulation models*. The entire field of computer simulation is suffering from a plethora of different concepts, formalisms and technologies and from a lack of common modeling languages and standards. Our aim is to understand the field in order to improve the vision of the M&S based on the Discrete-Event System Specification (DEVS) formalism (Zeigler, Muzy and Kofman 2018).

DEVS is a modeling formalism based on systems theory that provides a general methodology for hierarchical construction of reusable models in a modular way. Over the years, DEVS has finding an

increasing acceptance in the model-based simulation research community becoming one of the preferred paradigms to conduct modeling and simulation enquiries (Wainer and Mosterman 2010). Moreover, several researchers have proposed extensions and new formalisms derived from DEVS with aims to solve different scenarios (Blas et al. 2018). In this context, the interplay of abstraction and concreteness in advancing the theory and practice of M&S can be improved with the MDE modeling levels. Given that such interplay and the associated interplay of the generic and the specific impel the development of knowledge (Zeigler, Muzy and Kofman 2018), the implicit and explicit knowledge defined in DEVS formalism can be studied employing the conceptual modeling perspective of MDE over the main M&S dimensions.

From the traditional point of view, a good conceptual model lays a strong foundation for successful simulation modeling and analysis (Robinson et al. 2010). Over such interpretation, conceptual modeling serves as a bridge between problem owner and simulation modeler. However, it can also be applied for studying the simulation field with aims to define full M&S strategies based in conceptual models as a foundation for the M&S tasks. For example, a well-defined standardized conceptual modeling technique can decrease the effort to understand conceptual models, and solve interoperability problems between components that have been implemented in different environments or that are based on different formalisms (Cetinkaya, Verbraeck and Seck 2010).

This paper reviews several conceptual models of DEVS developed over the years in terms of three dimensions: *domain*, *formal specification* and *implementation*. Each dimension depicts a MDE modeling level. Therefore, they can be used as foundation of a broad conceptualization of DEVS. Here we identify a set of desired concepts for each dimension in order to define the main characteristics required in the MDE models. The final goal of this research is building an universal representation of DEVS based on MDE modeling levels that provides a conceptual model useful for developing an ontology-based M&S framework for DEVS.

The remainder of this paper is structured as follows. Section II describes the main M&S motivation for building a universal representation of DEVS using conceptual modeling as a foundation. Section III summarizes the set of DEVS conceptualizations studied in order to define the main concepts required in each dimension. It also introduces the notion of the M&S framework that will employ such representation as core of the semantic DEVS description. Finally, Section IV is devoted to conclusions and future work.

2 THE NEED FOR UNIVERSAL REPRESENTATION OF DEVS

DEVS is a popular modular and hierarchical formalism for modeling complex dynamic system using discrete-event abstraction. Because of the ease of model definition, model composition, reuse, and hierarchical coupling, DEVS has always been successfully applied in a variety of applications (Risco-Martín et al. 2017). Recently, it has been proved than DEVS is equivalent to the Iterative System Specification (ISS). In this way, the ISS has been used as an intermediate specification between DEVS at the lower (computational) level and general system behavior at the higher level (Muzy, Zeigler and Grammont 2017). Then, DEVS can be used to specify general system structure and compute its behavior as a computational implementation of the ISS.

In M&S, abstractions lead to new model representations (i.e. new computational environments). Figure 1 shows the instance of the *progression of abstraction to implementation* for the evolution of DEVS. In such progression, an *abstraction* focuses on an aspect of reality (*phenomenon*) and greatly reduces the complexity of the reality being considered. Then, *formalization* makes it easier to work out implications of the *abstraction* and implement in reality. Finally, *implementation* can be considered as providing a concrete realization of the *abstraction*. In this sense, Figure 1 shows how the abstraction of *discrete events, states, and components* can be formalized in *DEVS* and, then, it can be implemented in some *simulation environment based on DEVS*. Hence, *DEVS* formalizes the *discrete event system* abstraction and became the foundation for a *family of simulation system*.

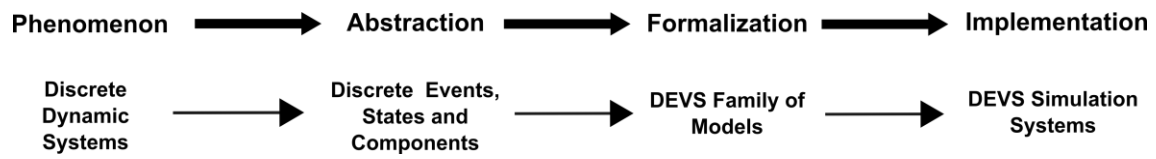

Figure 1: Progression of abstraction to implementation for the evolution of DEVS (adapted from (Zeigler, Muzy and Kofman 2018)).

The main concepts of this progression (*abstraction*, *formalization*, and *implementation*) help to understand the progress of knowledge in M&S. Therefore, a full study of DEVS formalism must consider such dimensions in order to get a complete understanding of the discrete event simulation domain.

According to (Guizzardi and Wagner 2012), Discrete Event Simulation (DES) is concerned with the simulation of real-world systems that are conceived as discrete event systems (or 'discrete dynamic systems'). The behavior models of this type of systems can be described at different levels of abstraction and, therefore, by means of different formalisms. The particular formalism and level of abstraction depends on the background and goals of the modeler as much as on the system modelled (Vangheluwe 2008). In this context, a formalization based on DEVS is always applicable.

DEVS is formalized using set theory and systems theory. The formalism includes two types of DES models: *atomic models* (behavior description) and *coupled models* (structural description). In both cases, the models are described using equations, functions, sets, etc. Therefore, DEVS is an abstract formalism for the specification of simulation models that is independent of any particular implementation. However, when engineers want to simulate these models they need to program them in the input language of a concrete simulator, which means writing code in Java or C++ or another general-purpose programming language (Cristiá, Hollmann and Frydman 2019). Such implementation is often called "reduction to concrete form" (Zeigler, Muzy and Kofman 2018).

Nowadays, there are multiple software tools and simulators for DEVS models (Van Tendeloo and Vangheluwe 2017).Though, modelers must represent DEVS model as programming code using predefined libraries offered by each simulator (Nikolaidou et al. 2008). Each simulator has its own input language. Generally, these input languages are different, hindering the interoperability between simulation tools (Hollmann, Cristiá and Frydman 2015). Then, DEVS models can be mathematically described but its simulation is performed by concrete DEVS simulation systems (DEVS simulators). When concrete DEVS models are developed using programming languages, it is difficult to ensure they conform to their formal model (Sarjoughian, Alshareef and Lei 2015). Moreover, the mathematical properties and constraints defined in DEVS models must be guaranteed in any implementation of it. Hence, it would be desirable to be able to describe the formal model and then automatically translate it to one or more DEVS simulation system by selecting an input language or an specific software tool.

In this case, an *universal representation of DEVS* can help to study the relationships among DEVS *formalization*, DEVS most commonly used *implementations* and the *discrete event system domain*. Sarjoughian, Alshareef and Lei (2015) state that "it is useful to have a framework that can not only capture the formal specification of DEVS atomic models but also enforce its syntax and semantics for domain specific metamodels". Now, if such *universal representation of DEVS* is *ontology-based*, then will be possible to build a *M&S framework* that provides *computational reasoning*, *concepts alignment* and *automatic translation* among the components included in each dimension. Moreover, since DEVS can serve as a simulation "assemble language" to which models in other formalism can be mapped (Vangheluwe 2000), an ontological representation can be used as support language to support such assemble through a computational process. Hence, the DEVS formalism can be used as standardization specification from two different points of view: *i)* as a common language for other discrete-event formalisms, or *ii)* as a specification platform for unifying multiple models developed with different abstraction levels.

Figure 2 illustrates how the general dimensions of DEVS (detailed in Figure 1) can be interpreted in terms of an ontology-based conceptualization.

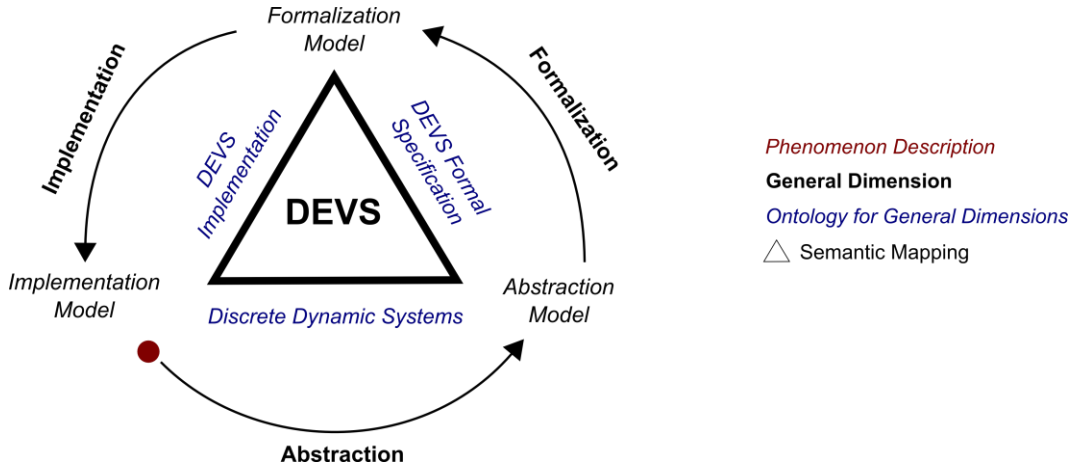

Figure 2: Core of DEVS universal representation.

The *Abstraction* is the base for *Formalization* and *Implementation* (starting from a *Phenomenon* description). For each general dimension, the Figure 2 proposes the following ontologies:

- *Discrete Dynamic Systems* ontology for supporting the *Abstraction* dimension;
- *DEVS Formal Specification* ontology for supporting the *Formalization* dimension;
- *DEVS Implementation* ontology for supporting the *Implementation* dimension.

Each ontology acts as intermediary between two specific models. For example, in order to get a *Formalization Model* from an *Abstraction Model*, the *Formalization* dimension uses the *DEVS Formal Specification* ontology. Then, the *Formalization* of an *Abstraction* (derived from a *Phenomenon*) can be seen as an instance of the *DEVS Formal Specification* ontology. The same assertion is true for the other dimensions.

In this context, the main benefit obtained from establishing the ontological foundations of the core concepts of a conceptual modeling language is a clarification of its real world semantics (Guizzardi and Wagner 2010). Even when a single formalism, thought free from semantics alignment issue by construction, is alone rarely appropriate for all levels of abstraction (Zhang, Zeigler and LaiLi 2019); the goal of the DEVS universal representation is to explicitly define a common semantic description of DEVS that can be used as description of the discrete-event specification for enabling a broad modeling and simulation approach. Therefore, the use of ontologies to support the progression of DEVS simulation models provides a solid common language for describing and handling the transformation among the main M&S dimensions.

3 FINDING ONTOLOGY CONCEPTS: STUDY OF EXISTING CONCEPTUAL MODELS OF DEVS FOR THE M&S DIMENSIONS

An ontology is an explicit specification of a conceptualization, that is, an abstract, simplified view of the world that includes the objects, concepts, and the relationships between them in a domain of interest (Gruber 1993). Ontologies definition allows an unambiguous specification of the structure of knowledge in a domain, enables knowledge sharing and reuse and, consequently, makes automated reasoning about ontologies possible (Orgun and Meyer 2008). However, in order to define an ontology, it is necessary to identify the set of entities to be included as concepts. Such identification should be obtained from the existing knowledge of the domain under representation.

Over the years, several researchers have developed conceptual models related to DEVS formalism. But not all models are founded in the M&S dimensions. Most conceptual models are based on domain-solution strategies for building DEVS simulation models. Therefore, a literature review was performed in order to identify an initial set of conceptualizations based on the *Abstraction*, *Formalization* and *Implementation* dimensions. The following subsections resumes the DEVS conceptual models obtained as result of this review process (including the set of main concepts identified for building the M&S ontologies).

3.1 Abstraction Dimension: Domain of 'Discrete Dynamic Systems'

Guizzardi and Wagner (2010) present DESO, a foundational ontology for discrete event system modeling derived from the foundational ontology UFO. The main purpose of such ontology is to provide a basis for evaluating discrete event simulation languages.

The authors claim that "a discrete event system model may be expressed at different levels of abstraction". In this context, they preset two ontologies derived from UFO: *i)* the *Design-Time Ontology DESO-U* that describe a discrete event system by defining the entity types (i.e. the instances which are part of the system), and *ii)* the *Run-Time Ontology DESO-I* that deals with individuals of different types from the simulator perspective.

DESO does not contain all the foundational concepts of UFO. Rather, it just contains the leaf nodes of UFO that are relevant for describing discrete event systems. Therefore, if the concepts of UFO are removed from DESO, the remaining concepts are the ones that mainly define the domain of discrete event systems. These concepts include descriptions of *Event*, *Entity*, *Entity Type*, *Attribute*, *Property*, *Object*, *Object Type*, *Situation* and *State*.

On the basis of DESO, the *Discrete Dynamic Systems* ontology will be defined. The ontology will provide a feasible representation of our conceptualization of discrete event systems from the *Abstraction* dimension without dealing with the simulator perspective. Then, the conceptualizations included in *Run-Time Ontology DESO-I* will not be used as part of the *Abstraction* perspective (because they deal with entities related to the *Implementation* dimension). Given that we are not really interested in considering all the things that constitute a real-world system, the ontology will be only focused on the aspects of reality that help to define abstractions of discrete events, states, and components. Therefore, the *Discrete Dynamic Systems* ontology will contain a smaller number of concepts than DESO.

3.2 Formalization Dimension: Domain of 'DEVS Formal Specification'

A DEVS conceptualization based on the *Formalization* dimension needs to be focused on the entities that describe DEVS simulation models. In this context, Hu et al. (2013) propose an ontology-based model representation named DEVSMO (DEVS math ontology) to support model reuse. DEVSMO is composed of three ontologies: *i)* the *DEVS model ontology* that describes the classification of DEVS models according to DEVS formalism, *ii)* the *model structure ontology* that consists of atomic model structure and coupled model structure, and *iii)* the *model behavior ontology* that defines the behavior of a DEVS model. In DEVSMO, the concepts related to the *Formalization* dimension are mainly defined in *i)* and *ii)*. A similar approach is proposed in (Touraille 2012) where the author proposes a metamodel of DEVS formalism that provides a pivot format for building DEVS simulation models compatible with the available software tools. This metamodel conceptualizes DEVS models using two parts: *i)* a *structural part* that deals with state variables, ports, components, connections, etc., and ii) a *behavioral part* that describes the temporal evolution of models in terms of transition functions, time advance function, etc. A DEVS *behavioral metamodeling* is also proposed in (Sarjoughian, Alshareef and Lei 2015). However, in this case, the authors use metamodeling with aims to generate concrete models from domain-specific metamodels. Therefore, the metamodel behavior specification identifies abstractions for state transitions in the external, internal, and confluent transition functions. Similarly appropriate abstractions are proposed for the output and time advance functions.

Hollmann, Cristiá and Frydman (2015) employs a different point of view for modeling DEVS formalization. Authors propose a formal modeling language called CML-DEVS (Conceptual Modeling Language for DEVS) that allows abstract description of DEVS models in terms of logical and mathematical expressions without involving programming concepts. CML-DEVS can be seen as an improved version of the specification language for DEVS models named DEVSpecL, developed by Hong and Kim (2006).

Table 1 summarizes the main concepts obtained from literature that will be used for building the *DEVS Formal Specification* ontology.

Table 1: Concepts to be used for modeling the *Formalization* dimension.

Concept	**Description**	**References**
Atomic Model	Entity composed of all the elements that specify a DEVS atomic model.	(Hu et al. 2013; Touraille 2012; Hollmann, Cristiá and Frydman 2015)
Coupled Model	Entity composed of all the elements that specify a DEVS coupled model.	(Hu et al. 2013; Touraille 2012)
State	Entity defined as the state set of elements that describes the state definition of DEVS models.	(Hu et al. 2013; Sarjoughian, Alshareef and Lei 2015)
State Variable	Entity that describes a variable that composes the state definition in terms of its data properties.	(Hu et al. 2013; Touraille 2012)
Internal Transition Function External Transition Function Output Function Time Advance Function	Entities that model functions defined as part of DEVS atomic models.	(Hu et al. 2013; Sarjoughian, Alshareef and Lei 2015; Hollmann, Cristiá and Frydman 2015)
Port	Entity that models a port that belongs to a component which refers to a DEVS model.	(Hu et al. 2013; Touraille 2012)
Port Coupling	Entity that define a coupling as a composition of source port and target port.	(Hu et al. 2013; Touraille 2012)
EIC EOC IC	Entities that define specific types of port couplings.	(Hu et al. 2013; Touraille 2012)

Other concepts to be included in the ontology are *State Set*, *Input Set*, *Output Set*, *Input Port*, *Output Port*, *State Transition*, *Model Structure* and *Model Behavior*. Moreover, the *Function* concept may be also included as a super concept of *Internal Transition Function*, *External Transition Function*, *Time Advance Function* and *Output Function*. Some of these concepts already exist in the conceptual models studied but their application depends on the modeling approach used by authors.

3.3 Implementation Dimension: Domain of 'DEVS Implementation'

Since there is no common DEVS format used by all tools, modelers are tied to their tool (Van Tendeloo and Vangheluwe 2017). Then, each conceptualization used for building DEVS models in specific software tools is unique. Hence, the *DEVS Implementation* ontology proposed as description of the *Implementation* dimension needs to be focus on the common-concepts included in most software tools. In this way, the ontology will be developed keeping in mind that many DEVS tools are already available and, also, it will help to provide as much integration as possible among them.

Several metamodels of DEVS have been developed for specific purposes. For example, Nikolaidou et al. (2008) propose a SysML (Systems Modeling Language) profile supporting the description of DEVS simulation models. Authors build DEVS SysML profile employing as basis the entities defined in a DEVS metamodel. The aim of this research was to offer a graphical, standardized environment for the definition of DEVS models using a SysML profile. A similar approach is presented in (Gonzalez et al. 2014) where authors propose an Ecore metamodel for DEVS that is used with aims to map UML states machines over DEVS atomic models.

Applying the same point of view that the one proposed for the *DEVS Implementation* ontology, Garredu et al. (2012a) propose a platform independent meta-model for DEVS formalism that is enriched with Object Constraint Language (OCL) constraints in (Garredu et al. 2012b). Authors state that such meta-model is "accurate enough to specify several DEVS models". A similar research related to SysML is presented in (Kapos et al. 2014). In such research authors discuss on the adoption of Model Driven Architecture concepts in order to seamlessly transform SysML models to executable DEVS models. Therefore, they build a MOF metamodel for DEVS in order to provide a standard representation for the simulation-specific domain. In (Cetinkaya, Verbraeck and Seck 2010) authors introduce a metamodel for component based hierarchical simulation based on Zeigler's Hierarchical DEVS. This metamodel initiates the definition of formal component based conceptual modeling technique to overcome the problems in hierarchical simulation.

Table 2 resumes the concepts obtained from literature that will be used to abstract most commonly-used DEVS implementations with aims to build the *DEVS Implementation* ontology.

Table 2: Concepts to be used for modeling the *Implementation* dimension.

Concept	**Description**	**References**
DEVS Model	Entity that abstracts the definition of both types of DEVS models.	(Garredu et al. 2012a; Garredu et al. 2012b; Nikolaidou et al. 2008; Kapos et al. 2014)
Atomic DEVS	Entity that defines the implementation of the DEVS atomic model.	(Garredu et al. 2012a; Garredu et al. 2012b; Gonzalez et al. 2014; Kapos et al. 2014; Cetinkaya, Verbraeck and Seck 2010; Nikolaidou et al. 2008)
Coupled DEVS	Entity that defines the implementation of the DEVS coupled model.	(Garredu et al. 2012a; Garredu et al. 2012b; Cetinkaya, Verbraeck and Seck 2010; Nikolaidou et al. 2008; Kapos et al. 2014)
State	Entity that defines the state of DEVS atomic models as an implementation based on variables, data types and parameters.	(Gonzalez et al. 2014; Nikolaidou et al. 2008; Kapos et al. 2014)

Coupling	Entity that defines the implementation of DEVS couplings in terms of its source and destination ports.	(Garredu et al. 2012a; Garredu et al. 2012b; Cetinkaya, Verbraeck and Seck 2010; Nikolaidou et al. 2008)
EIC EOC IC	Entities that describe specific implementations of DEVS coupling types.	(Garredu et al. 2012a; Garredu et al. 2012b; Kapos et al. 2014)
Port	Entity that defines a DEVS port using a name.	(Garredu et al. 2012a; Garredu et al. 2012b; Cetinkaya, Verbraeck and Seck 2010; Nikolaidou et al. 2008)
Input Port Output Port	Entities that describe specific types of DEVS ports in order to receive/send events.	(Garredu et al. 2012a; Garredu et al. 2012b; Cetinkaya, Verbraeck and Seck 2010)
Event	Entity that specifies the implementation of the basic unit to be exchanged among models.	(Gonzalez et al. 2014; Kapos et al. 2014)

As in the *Formalization* dimension, there is a set of concepts that each conceptualization uses with aims to achieve its implementation modeling goal. We believe that such concepts should be (at least) considered as potential entities of the *DEVS Implementation* ontology. For example, a full description of the *State* concept implementation is defined in (Kapos et al. 2014). This conceptualization includes *State Variable* and *State Set* as components, but also details *Time Advance Function*, *Internal Transition Function* and *External Transition Function* as states handlers. In (Garredu et al. 2012b) authors include *Rule*, *Expression*, *Type*, *Variable*, *Action* and *Condition* as metamodel concepts. Similarly, the model proposed by Cetinkaya, Verbraeck and Seck (2010) include *Parameter*, *Rule*, *Entity* and *Component*. Moreover, Gonzalez et al. (2014) incorporate *Transition*, *Internal Transition*, *External Transition*, *Input Event* and *Output Event*; while Nikolaidou et al. (2008) employ the notion of *Message*.

Table 2 shows that there is a broad set of concepts to be considered in the *Implementation* dimension. This issue is given by the complexity of modeling an universal representation of DEVS models implementation that includes all possible deployments.

3.4 M&S Ontology-based Framework for DEVS

Yang et al. (2019) have studied the current state of art of ontology-based system engineering. They have identified the following contributions of ontology-based systems: *1)* enabling interoperability and communication among multiple disciplines or across different stakeholders; *2)* integrating, mapping, exchanging and reusing knowledge; *3)* describing concepts and their relationships explicitly and accurately to avoid incompleteness and ambiguity; *4)* developing a domain knowledge representation; *5)* unifying a controlled vocabulary or semantics for capturing declarative knowledge; *6)* providing core and basic concepts as a reference to describe other concepts; *7)* defining a homogeneous terminology to eliminate inconsistency; *8)* sharing a common understanding of a domain; *9)* capturing knowledge in a formal language; *10)* allowing, expressing and reasoning about machine-readable programmable complex logical axioms; and *11)* visualizing and navigating knowledge repository.

In this context, the ontologies proposed as core of the DEVS universal representation can be used to build a software M&S framework for DEVS. These ontologies can be seen as conceptual models that define abstraction levels of DEVS formalism from different (but related) perspectives. Moreover, the MDE

principles can be applied to such software framework in order to improve its development. The M&S field has applied the MDE approach for years. MDE is based on a number of principles that involve the concepts of model, metamodel, meta-metamodel and model transformations in order to provide a process that enables the automated development of a system (Cetinkaya, Verbraeck and Seck 2011). From the MDE perspective, MDE produces well-structured and maintainable systems and increases the level of abstraction.

Figure 3 shows how the MDE modeling levels can be mapped into the ontologies proposed for the M&S dimensions with aims to build a software M&S framework. A full representation of a *DEVS model* is given by the set *{Abstraction Model, Formalization Model, Implementation Model}*. Each kind of model is defined as an instance of the ontology that represents its dimension. In the Figure, each ontology is sketch using a set of related concepts. As interoperability example, all the ontologies include the *State* concept. However, even when all the *State* concepts refer to the same *DEVS model*, each conceptualization depicts the state of the model from a different perspective (M&S dimension). Then, a set of *traceability relationships* can be defined among all the conceptualizations of the same concept (in Figure 3 such relationships are depicted only for the *State* conceptualizations). This traceability allows to align concepts from different ontologies (that is, the M&S dimensions) with aims to get more information about the model definition. Such information can be interpreted as new knowledge referred to the structure of DEVS dimensions. Moreover, it provides a solid basis for defining *mapping relationships* among concepts defined in different dimensions. Therefore, inference rules can be designed over the ontology network in order to get new knowledge. In both cases, for each pair of M&S dimensions, a software framework based on the proposed ontologies could implement several *model-to-model* transformations following the MDE approach. Then, it is useful to consider the M&S dimensions in the same way than the MDE approach.

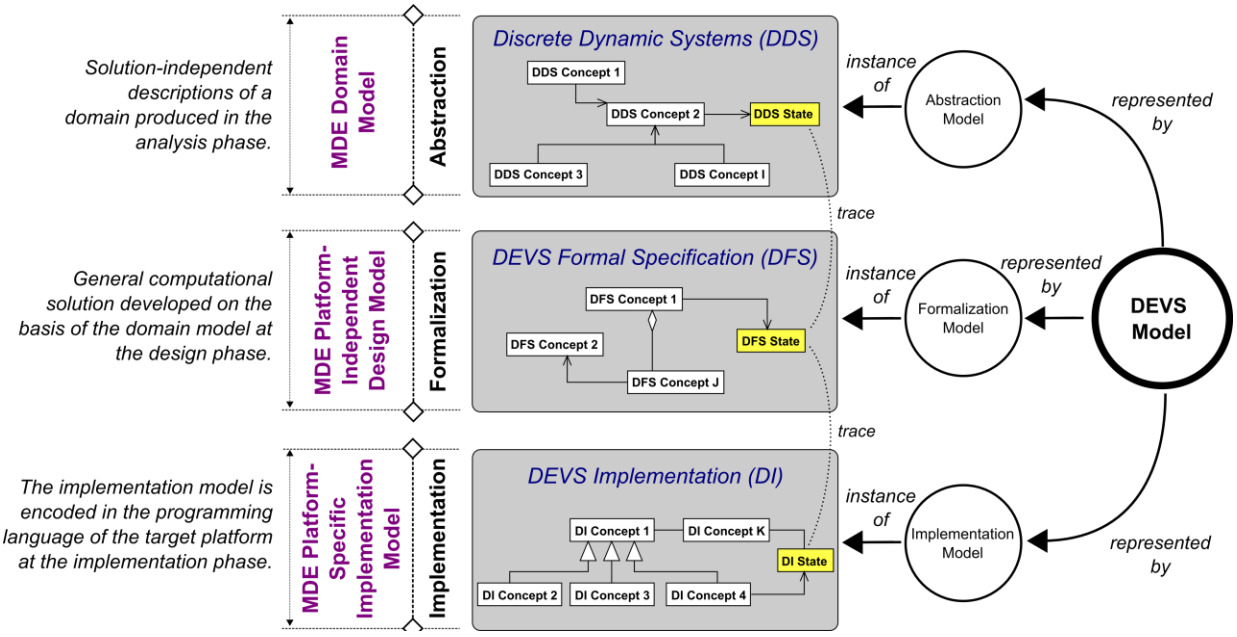

Figure 3: Employing the DEVS universal representation as basis of the DEVS M&S framework.

A corollary of the ontology-based universal representation of DEVS is the possibility of designing specific alignments with the DEVS conceptual models developed over the years. The value of such alignments depends on the M&S dimension under consideration. For the *Abstraction* dimension, different types of dynamic systems conceptualizations can be aligned to the *Discrete Dynamic Systems* ontology if and only if the dynamic system type can be reduced to a discrete system representation. This alignment will provide the possibility to obtain an *Abstraction Model* based on DEVS for the dynamic system types

involved in the mapping. For the *Formalization* dimension, conceptual models of other kinds of discrete-event formalisms can be aligned to the *DEVS Formal Specification* ontology if and only if the formalism can be reduced to a set of DEVS formal models. This alignment will provide the possibility to design multi-formalism models that can be described (ultimately) as a *Formalization Model* based on DEVS. Finally, for the *Implementation* dimension, software tools metamodels can be aligned to the *DEVS Implementation* ontology if and only if the concepts included in the software tool can be mapped to the common-abstractions of DEVS simulators. This alignment will provide the possibility to translate an *Implementation Model* based on DEVS to multiple software tools for its simulation.

Therefore, the MDE principles can be used as foundation for designing and implementing not only the ontologies for the M&S dimensions but also for defining new mappings with existing models. Over all these assumptions, the M&S framework will be capable of translating several representations of the same model with aims to get a full consistent DEVS model. Hence, the implementation of the DEVS M&S framework from the DEVS universal representation proposed in this paper should help mainly to:

- validate the consistency among discrete dynamic system descriptions and their DEVS formal models;
- verify the consistency among DEVS formal models and their simulation implementations developed using specific software tools (concrete simulators);
- provide a basis for building multi-formalism models based on DEVS;
- give a background for developing models composition based on DEVS;
- exploit the models reuse capabilities at semantic level in order to make compatible implementations;
- contribute to the deployment of holistic DEVS simulation models implemented with different software tools (concrete simulators);
- take measures related to the abstraction, formalization and implementation of DEVS simulation models.

4 CONCLUSIONS AND FUTURE WORK

This paper is a work in progress aimed to define an universal representation of DEVS using ontologies. This representation extends the context of the M&S field trying to capture the associations among the *Abstraction*, *Formalization* and *Implementation* dimensions used in DEVS models. With the help of the MDE approach, we will use these ontologies as foundation for building a M&S framework for DEVS.

The future work includes the design and implementation of the semantic models required for each M&S dimension employing the set of concepts identified from the literature review. However, even when ontologies can improve the representation of the *Abstraction*, *Formalization* and *Implementation* dimensions, their integration as underlying mechanism of a M&S framework for DEVS will probably need to consider other aspects related to systems theory in order to get the complete picture. As there were presented in Figure 3, the ontologies proposed for the *Abstraction*, *Formalization* and *Implementation* dimensions are depicted as partial views of the complete DEVS description. Even with concepts traced among dimensions, the systems concept underlying DEVS could be included with aims to define a conceptual description at the general system level. The underlying *abstraction* of DEVS is the I/O system which is specifiable at multiple levels of structure and behavior with their associated morphisms. Moreover, DEVS itself is a system specification, indeed, an iterative system specification (*formalization*). And, finally, the *implementation* is the abstract DEVS simulator (Zeigler, Muzy and Kofman 2018). Therefore, each one of these components can give rise to its own set of concepts and relationships (i.e. its ontology class) which is distinct from the others but are all consistent from the I/O systems perspective. Therefore, prior unifying the M&S ontologies, we will study these domains with aims to include their conceptual information as part of the framework.

REFERENCES

Blas, M., S. Gonnet, H. Leone, and B. Zeigler. 2018. "A conceptual framework to classify the extensions of DEVS formalism as variants and subclasses". In *Proceedings of the 2018 Winter Simulation Conference*, edited by M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, pp. 560-571. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Cetinkaya, D., A. Verbraeck, and M. D. Seck. 2010. "A metamodel and a DEVS implementation for component based hierarchical simulation modeling". In *Proceedings of the 2010 Spring Simulation Multiconference*, pp. 130-138. Society for Computer Simulation International.

Cetinkaya, D., A. Verbraeck, and M. D. Seck. 2011. "MDD4MS: a model driven development framework for modeling and simulation". In *Proceedings of the 2011 Summer Computer Simulation Conference*, pp. 113-121. Society for Modeling & Simulation International.

Cristiá, M., D. A. Hollmann, and C. Frydman. 2019. "A multi-target compiler for CML-DEVS". *Simulation*, vol. 95(1), pp. 11-29.

Garredu, S., E. Vittori, J. Santucci, and P. Bisgambiglia. 2012a. "A Meta-Model for DEVS-Designed following Model Driven Engineering Specifications". In *Proceedings of the 10th International Conference on Simulation and Modeling Methodologies, Techniques and Applications*, pp. 152-157.

Garredu, S., E. Vittori, J. Santucci, and D. Urbani. 2012b. "Enriching a DEVS meta-model with OCL constraints". In *Proceedings of the 24th European Modeling and Simulation Symposium*, pp. 216-225.

Gonzalez, A., C. Luna, R. Cuello, M. Perez, and M. Daniele. 2014. "Metamodel-based transformation from UML state machines to DEVS models". In *Proceedings of the XL Latin American Computing Conference*, pp. 1-12.

Gruber, T. A. 1993. "A translation approach to portable ontology specifications". *Knowledge Acquisition*, vol. 5(2), pp. 199–220.

Guizzardi, G., and G. Wagner, G. 2010. "Towards an ontological foundation of discrete event simulation". In *Proceedings of the 2010 Winter Simulation Conference*, edited by B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, pp. 652-664. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Guizzardi, G., and G. Wagner, G. 2012. "Conceptual simulation modeling with Onto-UML". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, pp. 52-66. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Hollmann, D. A., M. Cristiá, C. Frydman. 2015. "CML-DEVS: A specification language for DEVS conceptual models". *Simulation Modelling Practice and Theory*, vol. 57, pp. 100-117.

Hong, K. J., and T. G. Kim. 2006. "DEVSpecL: DEVS specification language for modeling, simulation and analysis of discrete event systems", *Information and Software Technology*, vol. 48, pp. 221–234.

Hu, Y., J. Xiao, H. Zhao, and G. Rong. 2013. "DEVSMO: an ontology of DEVS model representation for model reuse". In *Proceedings of the 2013 Winter Simulation Conference*, edited by R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, pp. 4002-4003. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Kapos, G. D., V. Dalakas, A. Tsadimas, M. Nikolaidou, and D. Anagnostopoulos. 2014. "Model-based system engineering using SysML: Deriving executable simulation models with QVT". In *Proceedings of the 2014 IEEE International Systems Conference*, pp. 531-538.

Muzy, A., B. P. Zeigler, and F. Grammont. 2017. "Iterative specification as a modeling and simulation formalism for I/O general systems". *IEEE Systems Journal*, vol. 12(3), pp. 2982-2993.

Nikolaidou, M., V. Dalakas, L. Mitsi, G. D. Kapos, D. Anagnostopoulos. 2008. "A SysML profile for classical DEVS simulators". In *Proceedings of the 3rd International Conference on Software Engineering Advances*, pp. 445-450.

Orgun, M. A., and T. Meyer. 2008. "Introduction to the special issue on advances in ontologies". *Expert Systems: The Journal of Knowledge Engineering*, vol. 25(3), pp. 175–178.

Risco-Martín, J. L., S. Mittal, J. C. Fabero Jiménez, M. Zapater, and R. Hermida Correa. 2017. "Reconsidering the performance of DEVS modeling and simulation environments using the DEVStone benchmark". *Simulation*, vol. 93(6), pp. 459-476.

Robinson, S., R. Brooks, K. Kotiadis, and D. J. Van Der Zee. 2010. *Conceptual modeling for discrete-event simulation*. CRC Press.

Sarjoughian, H. S., A. Alshareef, Y. Lei. 2015. "Behavioral DEVS metamodeling". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz, W K V. Chan, I Moon, T. M K Roeder, C Macal, and M D Rossetti, pp. 2788-2799. Piscataway, New Jersey, Institute of Electrical and Electronics Engineers, Inc.

Touraille, L. 2012. *Application of model-driven engineering and metaprogramming to DEVS modeling & simulation*. Doctoral dissertation, Universite´ d'Auvergne, France.

Van Tendeloo, Y., and H. Vangheluwe. 2017. "An evaluation of DEVS simulation tools". *Simulation*, vol. 93(2), pp. 103-121.

Vangheluwe, H. 2000. "DEVS as a common denominator for multi-formalism hybrid systems modelling". In *Proceedings of the 2000 IEEE International Symposium on Computer-aided Control System Design*, pp. 129–134.

Vangheluwe, H. 2008. Foundations of modelling and simulation of complex systems. In *Proceedings of the 7th International Workshop on Graph Transformation and Visual Modeling Techniques*, pp. 1-12.

Wainer G., and P. Mosterman. 2010. *Discrete-Event Modeling and Simulation: Theory and Applications*. Florida: CRC Press.

Yang, L., K. Cormican, and M. Yu. 2019. "Ontology-based systems engineering: A state-of-the-art review". *Computers in Industry*, vol. 111, pp. 148-171.

Zeigler, B. P., A. Muzy, and E. Kofman. 2018. *Theory of Modeling and Simulation: Discrete Event & Iterative System Computational Foundations*. Academic Press.

Zhang, L., B. P. Zeigler, Y. LaiLi. 2019. *Model Engineering for Simulation*. Academic Press.

AUTHOR BIOGRAPHIES

MARIA JULIA BLAS is a Postdoctoral Fellow at Instituto de Desarrollo y Diseño INGAR and an Assistant Professor in the Information Systems Department at Universidad Tecnológica Nacional. She received her PhD degree in Engineering from Universidad Tecnológica Nacional in 2019. Her research interests include discrete-event modeling and simulation. Her email address is mariajuliablas@santafe-conicet.gov.ar.

SILVIO GONNET received his PhD degree in Engineering from Universidad Nacional del Litoral in 2003. He currently holds a Researcher position at CONICET, working also as an Assistant Professor at Universidad Tecnológica Nacional. His research interests are models to support design processes and conceptual modeling. His email address is sgonnet@santafe-conicet.gov.ar.

BERNARD P. ZEIGLER is Professor Emeritus of Electrical and Computer Engineering at the University of Arizona, Tucson, AZ, USA and Chief Scientist of RTSync Corp., Phoenix, AZ, USA. Dr. Zeigler is a Fellow of IEEE and SCS and received the INFORMS Lifetime Achievement Award. He is a co-director of the Arizona Center of Integrative Modeling and Simulation. His e-mail address is zeigler@rtsync.com.