Approved for Public Release
20-MDA-10481 (13 July 20)

SUPPORTING THE REUSE OF ALGORITHMIC SIMULATION MODELS

Nicholas Keller
Bernard Zeigler
Doohwan Kim

RTSync Corp.
6909 West Ray Rd. STE 15-107
Chandler, AZ 85226, USA
{nicholas.keller, zeigler, dhkim}@rtsync.com

Chase Anderson

nou Systems Inc.
7047 Madison Pike #305,
Huntsville, AL 35806, USA
andersonchasemichael@gmail.com

James Ceney

Missile Defense Agency
721 Irwin Avenue
Colorado Springs, CO 80912, USA
james.ceney@mda.mil

ABSTRACT
Stateless functions, also referred to as algorithmic models, return an output given inputs that all occur at the same time instant. As relatively simple dynamic models, which define the behavior of variables over a timeline, algorithmic models nevertheless encode knowledge of entities that can be essential for use within models in a particular domain. This paper presents a development methodology for representing algorithmic models within the Discrete Event Systems Specifications (DEVS) formalism and employing the System Entity Structure (SES) to organize these models for reuse in new compositions. A use case example is used for illustration of the development process and the benefits in savings of time and effort are illustrated. Finally, some future possibilities to enhance the support of DEVS environments for this methodology are discussed.
Keywords: DEVS, SES, algorithmic, reuse

1 INTRODUCTION

Our goal is to develop a capability to easily integrate algorithmic models (e.g. gravity model, radar detection model) directly into a simulation architecture as reusable modules. The objective is to allow distributed development of models by expert teams, while retaining the advantages of an integrated simulation (consistency of models, consistent input data formats, centralized data management, etc.). The context within which this problem requires urgent solution is as follows: Modeling and simulation (M&S) users must move beyond creating and simulating individual models, to managing whole ecosystems of related models through a simulation architecture framework. The biggest challenge is model integration, because the M&S architecture designers must anticipate the needs of a wide variety of users. Designing for model composability is difficult because models can come in a variety of forms. Basing the architecture design on a firm system theoretical and computationally supported composition framework is crucial to achieve such design. The Discrete Event System Specification (DEVS) M&S formalism

provides the theoretical and practical basis for a solution which can leverage the widespread academic research into DEVS, open-source software development of DEVS, and extensive experience with DEVS.

Towards this objective, we evolved a capability using algorithmic models in end-to-end engagement scenarios. This was developed in collaboration with subject matter experts in radar technology modeling. The use case was developed on the existing RTSync owned Modeling and Simulation environment (MS4 Me) with essential properties:

- *Ease of model integration:* Using the DEVS-based specifications of the collaboration interface, a new model can be integrated into the simulation architecture. Given that the architecture adheres to the DEVS simulation protocol, such a new model does not require any changes to the simulator software.
- *Model composability:* The architecture supports finding and selecting needed models from its library of integrated models to support a hierarchical stage-wise construction employing DEVS-based coupling.

As discussed in Related Research, although there are other DEVS-based frameworks that support reusability, ***MS4 Me is the unique DEVS-based environment that employs the System Entity Structure (SES) to provide strong support for combinatorial composition***.

2 BACKGROUND

The DEVS model formalization specifies a model's inputs, states, and outputs in a manner similar to a finite state automaton. However, a key difference is that the formal structure includes a time-advance function which allows it to represent discrete event systems as well as simulated continuous components. Due to its universality of realization (Zeigler et al. 2018), any real world system, or system of systems (SoS) can be modeled in DEVS and simulated in a compliant computational platform. We briefly state and stress some key aspects of DEVS:

- DEVS formalizes what a model contains and doesn't contain. For example, a model is distinguished from the simulator (that executes it) and from the experimental frame (that determines the conditions under which its behavior is generated to produce results of interest to the user.)
- A DEVS model can be executed by any DEVS-compliant simulator (i.e., a computational platform that implements the operational semantics expressed in the DEVS abstract simulator (Zeigler et al. 2018) DEVS distinguishes the model from the simulator. This allows a DEVS model library to grow in value as advances in DEVS simulators are made. Additionally, this allows a DEVS model to be run on a variety of hardware such as PCs and in clusters.
- DEVS models can be composed under well-defined coupling rules. The basic, or uncomposed DEVS model, is called an atomic model. Atomic models can be coupled together (sending outputs to inputs) to form coupled models. Hierarchical composition is supported where coupled models can be linked with atomic models and other coupled models to create higher level coupled models. This is possible because DEVS models have well-defined interfaces with specific input and output ports, which make it possible to couple them together in predictable fashion.
- DEVS coupled models can be implemented as federations in distributed simulations (Zeigler et al. 2018). Provided that the Run Time Infrastructure (RTI) is DEVS-compliant, the time management is well-defined and conforms to the time advance functions specified by the atomic federates.
- Facilitated by experimental frames, data can be automatically collected from the exposed input and output ports of DEVS models.
- DEVS is discrete event so it inherently supports efficient simulation. Any continuous time or discrete time model can be converted to an equivalent DEVS model.
- DEVS-compliant simulators execute DEVS models correctly, repeatably, and efficiently. A strong property called “closure under coupling” guarantees correctness in hierarchical composition of DEVS components.

- The Parallel DEVS Simulation Protocol provides close to the best possible performance except possibly where activity is very low or coupling among components is very small.

Therefore future, well-designed DEVS-based simulators and models will greatly ease the difficulty of constantly upgrading technology migration. Unfortunately, legacy M&S is not likely to be entirely DEVS-compliant.

Current DEVS environments do not readily support the integration of algorithmic models explicitly through explicit interfaces. Here we will show that such models can be incorporated into DEVS atomic and coupled models thus providing access to simulation constructs defined by DEVS simulators. Whereas an algorithmic model returns an output given its inputs all occurring at the same time instant, an atomic model specifies behavior of its variables over the full time base. For example, an atomic model that represents an airborne missile would refer to an atmospheric drag algorithmic model at every time instant that it updates its trajectory while it is in the atmosphere.

The DEVS atomic model structure is shown in Figure 1 below. An atomic model transitions between different states due to either external input or timed internal transitions. There can be an arbitrary number of states. Output is only generated during an internal transition. The time advance step determines the next scheduled internal transition. The functions that encode these operations offer convenient loci into which algorithmic models can be placed. For example, an algorithmic model atmospheric drag function can be added to the time advance transition to enable it to compute the amount of time that can be taken before the next state update computation. (We note that DEVS models can run much faster than discrete time models because they can "skip" over periods of time to go to the next scheduled event.)

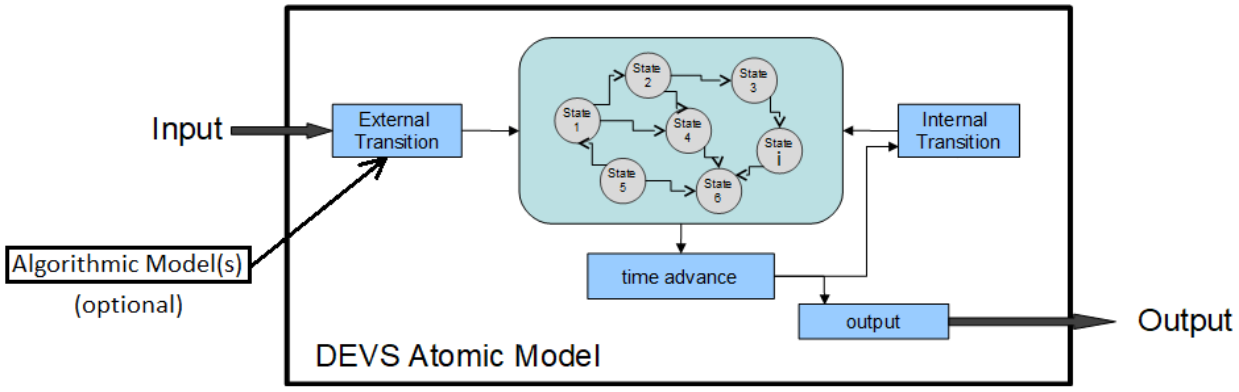

Figure 1: A DEVS atomic model with optional algorithmic models.

Atomic models can have algorithmic models, which can be used to encode re-usable physics equations. *Unlike atomic models, algorithmic models have no concept of time and are better thought of as re-usable functions.*

A model can be touched by 3 types of users: developing models, composing models, and simulating models (a particular user can take on more than one of these roles). Developing a new model will require using existing models, extending existing models, and creating entirely new models. Both atomic models and algorithmic models can also be re-used. Composing models (coupling them) allows previously generated models to be combined together through a graphical interface. There are two types of model composers: 1) those that are just composing existing models to create a scenario that they plan to simulate, or 2) those that are creating a coupled model that will use existing and newly created models. The first user may not be a programmer, so an intuitive graphical interface is a must. The second user can also benefit from a helpful graphical interface, but they will also want the power to directly code the coupling of models through text. The third and final use case is model simulation. This user may not be a programmer, so they need a graphical user interface to visualize the models and an intuitive way to run the models and collect data.

2.1 Review of MS4 Me

The MS4 Me is an environment to design general systems as well as Systems of Systems (SoS) based on systems theory (Zeigler et al. 2013; Zeigler and Nutaro 2016). MS4 Me supports the collaboration of domain experts and modelers in both top down and bottom up system construction. The *state diagram designer* supports graphical specification of atomic models and is automatically (and reversibly) converted to constrained natural language text. This text is translated into a Java atomic model class and compiled to execute in the MS4 Me execution environment based on the DEVS abstract simulator. An atomic DEVS model can be constructed within natural language for DEVS constructs such as time advance for each state, input/output ports, state transitions, internal transitions, external transitions, and output specification. However, a model expressed with limited natural language cannot specify a function's detailed behavior. To overcome this problem, the DNL file introduces tag blocks which enclose actual (Java) computer code to be inserted in specific locations within the Java class file, e.g., within the characteristic functions of the DEVS Java model. Concepts such as these tag blocks facilitate the inclusion of algorithmic model information into atomic model specifications

RTSync's MS4 Me modeling software uses the DEVS modeling formalism and System Entity Structure (SES) (Zeigler et al. 2013) ontology to support model composability. We demonstrated that MS4 Me provides an effective platform to create composite ballistic missile models that contain algorithmic models by having a domain expert who had never used MS4 Me before create a ballistic missile model of moderate complexity using this software. Although RTSync provided guidance in how to use MS4 Me, the SME developed the complete example for demonstration on his own.

2.2 DEVS and SES support model composability

We created a composable ballistic missile trajectory model using DEVS and SES using the tools of the MS4 Me platform. Figure 2 shows 5 of the 24 possible composite model generated ballistic missile trajectories with different drag algorithmic models and air density algorithmic model choices. As expected, the constant air density and low drag (with constant air density) trajectories have very short ranges because the high air resistance. These two trajectories are barely visible in figure 2 because they are very short. The effect of including higher resolution phenomenology for high altitudes clearly distinguishes these trajectories.

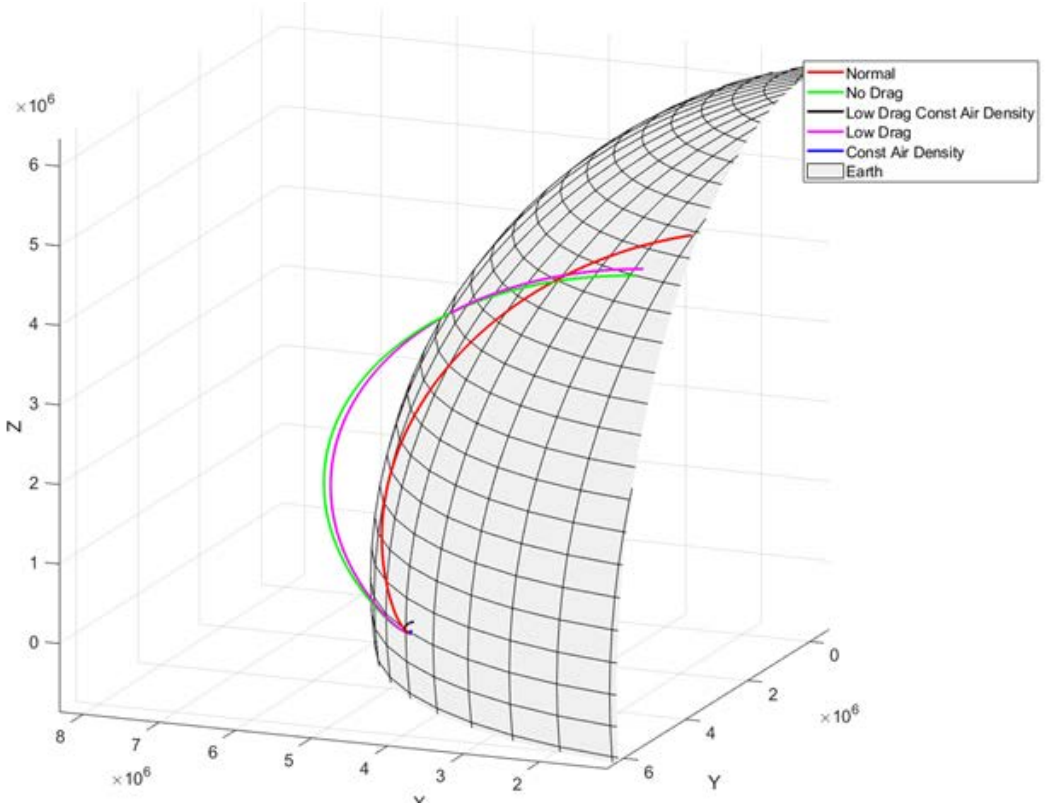

Figure 2: Ballistic missile trajectories under various drag conditions.

To create this model family, we followed the following process:

1. *Decide on specific objectives for the demonstration:* The goal was to replicate realistic missile trajectories that are currently generated by conventional technology at nSI while demonstrating the added value of the proposed technology.
2. *Identify the algorithmic models and transformational models that will become the basis for reusable atomic models*: analysis indicated that gravity, drag, and air resistance satisfy the requirements for expression in DEVS and SES reusable models. Also the reversible conversion of Cartesian to Spherical coordinates was recognized as a transformation that would be frequently employed in this context.
3. *Implement the algorithmic models as reusable atomic models*: with support from RTSync, the SME employed his experience in MATLAB missile computations to develop an approach to converting existing algorithms to DEVS models. The approach as implemented in MS4 Me is illustrated in Figure 3.

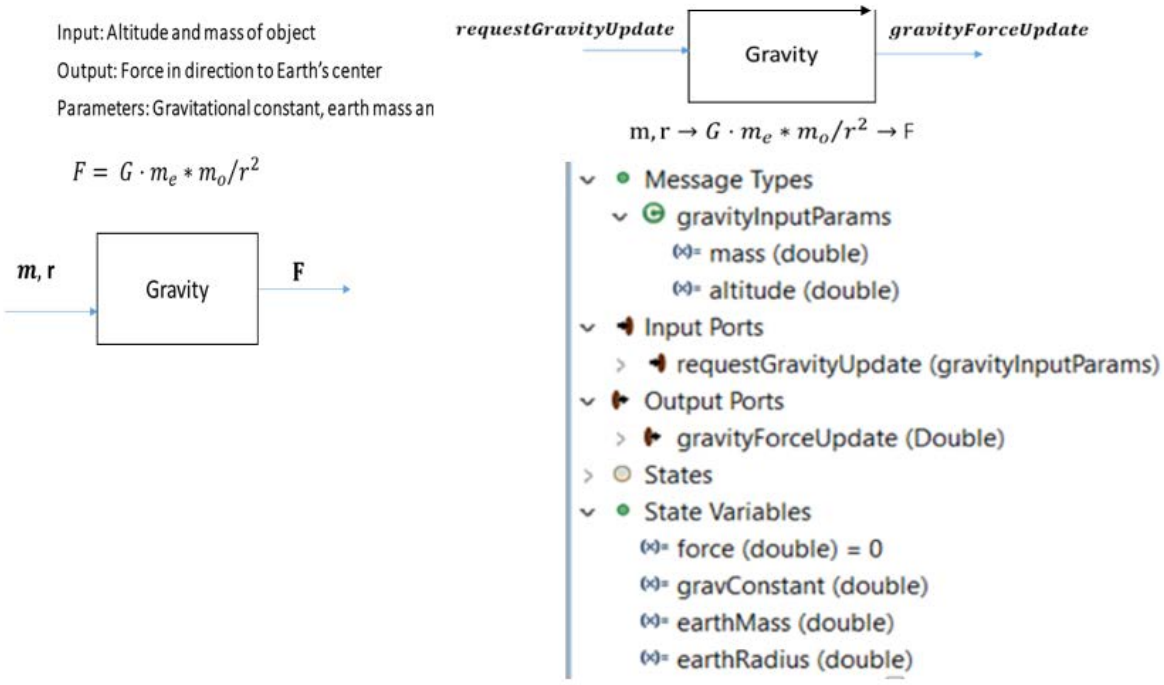

Figure 3: An algorithmic model representation in DEVS (MS4 Me DEVS outline on the right).

Gravity for example, is recognized as a function with input of altitude and mass of the reference object that computes an output representing a force pointing toward the earth's center. It gives rise to a DEVS atomic model that responds to message input requests for gravity update that are generated as events occur as the trajectory of the missile is updated in flight. Similarly, requests for drag calculation and air resistance values are generated as the missile computes successive values of its position. The latter is maintained in spherical coordinates and the missile requests a conversion to Cartesian form when computing the next position. Note that the drag is packaged as a coupled model that itself calls on drag calculation and air resistance for its function. This is an example of a reusable module at the composite level, i.e., composed of modules that are themselves reusable.

4. *Develop the overall structure for the coupled model to generate the desired trajectory behavior*: Figure 4 is an interaction diagram represented in MS4 Me that captures the exchange of messages required to compute a missile trajectory. DEVS coupled models can be composed of both other coupled models and atomic models. The lines between models represent message passing connections. This diagram automatically generates an SES and transforms to a coupled model of component models (as just described) that can be simulated with the DEVS simulator in MS4 Me.

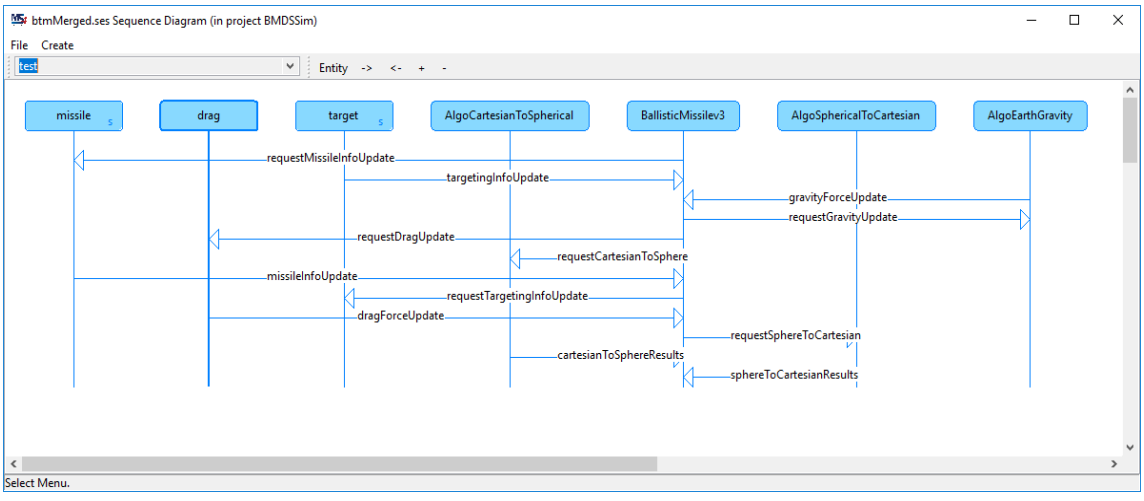

Figure 4: An interaction diagram for a missile trajectory simulation.

5. *Enhance the SES with specializations that express alternative choices for components of interest*: Figure 5 represents the enhanced SES that was employed to create the 24 composite ballistic models whose trajectories are illustrated in Figure 2.

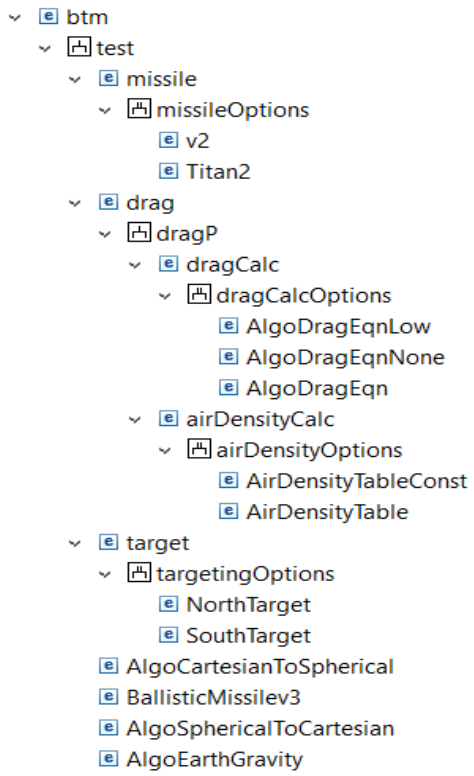

Figure 5: The SES outline of a ballistic trajectory missile model.

2.3 Combinatorial Generation of Compositions

In the SES of Figure 5 there are 3 possible drag algorithmic models and 2 air density algorithmic model options. There are 2 possible ballistic missile options (V2 or Titan 2) and 2 possible targeting options. Thus there are 3*2*2*2 = 24 possible model compositions of the ballistic missile trajectory model.

The success of this demonstration, confirmed that algorithmic models are relatively simple structures embodying reusable domain knowledge. This suggests, that the utility advantages of reusable composition of models can begin with these models. Therefore, we conclude that support for finding, composing, and reusing such models would be crucial to wide spread adoption of reusable simulation model development methodology.

2.4 Efficiency Benefits of Reusable Model Development

The SME reported that the total development time for any one of the 24 variations took approx. 4 hours using conventional tools (MATLAB). Thus the 24 model variants whose trajectories are illustrated in Figure 2 would take approximately 96 hours to complete using such tools. In contrast, development of the infrastructure for this model family (the SES and basic atomic models (including the algorithmic models)) took approx. 8 hours. With the framework in place development of a variant could take times ranging from a few minutes up to an hour, depending on the amount of new content to be added. Taking 30 minutes as the average, the total development time with the proposed technology is estimated at 8+24*.5

= 20 hours (with a maximum of 32 hours). This represents an estimated reduction of 70 to 80% in total development time. Moreover, the savings would continue to increase as new variants were attached to the framework. Our performance evaluation is indicative of the magnitude of time saved; however, the exact time saved on another modeling project will likely differ.

Reuse of models that have a high Verification and Validation (V&V) rating can entail large savings in development time and cost. Imagine an algorithmic model, such as air resistance, that finds a high level of reuse in the domain of interest. Then in a development environment that lacks reusability, such a module must be re-developed, verified, and validated each time it is needed. In contrast, with reusability development, the module is basically developed only once, and continually improved as needed in future applications. The result is that the savings afforded by such development become exponentially greater as time and usage proceeds.

3 RELATED WORK

Recognizing that new M&S capabilities are needed to meet future Army warfighting requirements Sanders (Sanders 2018) lists several modeling approaches that are emerging to leverage new technologies to create inherently interoperable and reusable artifacts across community boundaries. Several of the enumerated M&S frameworks are aimed at abstraction levels other than the model (Petty and Weisel 2019).Two of those listed, OSM (Winfrey et al. 2014) and DEVS-DMF (Kewley et al. 2016), are DEVS-based software frameworks that leverage inherent properties of DEVS to support composability and reuse at the model level as is proposed here. Our reusability development methodology differs in a major respect in its use of the SES to support composability as illustrated in the example developed. We note that recently other integrated modeling, simulation and experimentation environments based on SES and DEVS are under development (Pawletta et al. 2019). The composability feature of SES results in significant reduction in time to develop models for new objectives that can only be emulated with the use of *ad hoc* configuration scripts that are needed when using DEVS alone.

Moreover, our approach is intrinsically based on the system-theoretical foundation underlying the Modeling and Simulation Framework (MSF) (Zeigler et al. 2018). Indeed, current Model-Based System Engineering (MBSE) calls for formalized models to replace documents as the fundamental building blocks of systems engineering using model-driven architecture and the like (Aliyu et al. 2016). However, practicality demands that such models eventually *support all the activities typically associated with the simulation discipline. Current MBSE formalisms such* as SysML (Amissah and Handley 2016) stop well short of this capability. One approach to bridging this gap is to enable mappings to be defined that precisely specify simulation models that realize their behaviors. Taken to practical limits, this approach entails building more capability into such a formalism so that it eventually replicates all capabilities associated with traditional simulation methodology. Although there are attempts to achieve this goal (Bocciarelli and Giglio 2018), there are also fundamental reasons why it is not attainable (Abdurrahman and Sarjoughian 2018). An approach that underlies this proposal is to tie MBSE models with informal but well documented links to DEVS realizations. Further, as experience grows with such cross-links it might eventually become feasible to formalize these associations. On the other extreme, the approach of software-based architecture (Steinman et al. 2007), does not have the system-theoretic basis and consequent properties that are associated with DEVS.

4 CONCLUSIONS AND FUTURE WORK

Several possibilities to enhance support for reusability that may be considered for future work are discussed below:

The current concept of input/output ports allows models to be coupled; however, it doesn't check that the coupling makes sense semantically. Two models could be connected syntactically using a numerical port, but if one model is sending values in *m/sec* and the other is receiving values in *miles/hr*, then the coupled model will behave incorrectly. To prevent this problem, the interface concept can expand the input/output

port concept to include, among other things, units. The objective is to introduce the interface concept to improve model composition by removing the burden of having to check for semantic compatibility when coupling models while minimizing the amount of extra work required. Further, the interface concept is a prerequisite for many composition-related features. Acquiring additional interface information was found to provide a foundation for desired features such as: 1) unit testing, 2) automatic unit conversion, and 3) the automatic detection of specialization and multi-aspect pruning options. Interfaces make the creation and management of unit tests easier for modelers. Unit conversion can be automated by inspecting the units of interfaces. The unit conversion feature includes the conversion of multi-dimensional units such as coordinate transformation (e.g., Polar to Cartesian). The SES concept can be expanded to allow for pruning options to include models that satisfy certain interfaces. For example, an SES could include a ballistic missile that satisfies a specific interface; this allows the SES to take advantage of ballistic missile models that haven't been written yet.

The use case discussed above established algorithmic models can embody useful reusable domain knowledge. Support for finding, composing, and reusing such models is crucial to wide spread adoption of the proposed system. This task concerns the design of a browser that best supports user search for such stored components. Such a browser is needed since the existing MS4 Me provides the basic functionality to work with algorithmic models but not for searching for existing ones after construction.

The model browser can be designed to allow users to access: algorithmic models, atomic models, coupled models, interfaces and meta-data. Users can find models through: search, an index, or through related models. The interface is inspired by the Internet's linked pages. The "page" for a model will list, among other things: its author, models it contains, interfaces it uses, and models that contain it. Each of these items will be a "link" that users can click on to go to the page associated with that item. For example, clicking on its author will take the user to a page which lists other models that the author has worked on. This interface can be evaluated for its intuitive support of both finding models and understanding existing ones. Key word search will be the first type of search we implement. Search results will be ordered based on relevance. Key words found in model names can be considered more relevant than keywords found in model descriptions. Relevance ties will be broken by considering more frequently used models to be more relevant. In addition to key word search we will examine designs in which users search for models based on specific properties. This will include searching for models that: 1) use a particular interface, 2) couple with a particular model, or 3) satisfy a type specification (e.g., algorithmic, atomic, coupled).

Trust in stored models will increase with time as usage statistics and unit tests accumulate. The more a model is simulated without a detectable error the more likely it is error free. As a model advances through the development and reuse process, it can be tested against an increasing number of unit tests. The unit testing history (and the unit tests themselves) will be stored. The model browser can present model usage and unit test information to users to promote trust in the model base.

Potential users will need familiarization with the concepts and tools provided for algorithmic model construction, general model composition, finding available models, and reuse of existing components. For training others, we need to deepen and formalize the process described in Section 2.2. Further this experience has shown that a minimal amount of training can significantly shorten the learning curve required for full effective use of the system. Other users with less software development experience may require more intense background and more examples oriented to their interests to develop the required facility.

In the future we will extend the simulation model reuse framework to consider the organizational environment in which it will operate. We anticipate that engineers will collectively be using the M&S development environment (MS4 Me) to compose increasing complex hierarchical DEVS models with algorithmic models at the lowest level. The organization will have a procedure for Verifying, Validating and Accrediting (VV&A) such models. We must extend the framework to consider the VV&A needs of such a large organizations. One method we will consider is assigning accredited models a cryptographic hash to detect the accidental (or malicious) modification of an accredited model. To support the evolution of the model base in a collaborative environment we will borrow ideas from version control software. As

an example, the Missile Defense Agency (MDA) may, over time, create accredited comprehensive model families of sophisticated hardware such as the Terminal High Altitude Area Defense (THAAD) System including multiple facets such as radar, interceptor, missile models, resupply models and maintenance models. When a user wants to reuse such an accredited model, they should be able to pick and choose component models as needed. In the THAAD example, many users will be interested in the logistics but will want to extract the radar and intercept launch processes. The SES concept may have to be expanded to support such pruning and selection of desired model compositions in such an enterprise context. A model re-use framework must consider not only the technical aspects of model re-use but also organizational aspects; in many ways dealing with the pragmatic context of use is the more difficult challenge.

REFERENCES

Abdurrahman A I, Sarjoughian. H., Model-Driven Time-Accurate DEVS-Based Approaches for Design, TMS/DEVS, SpringSim, 2018

Aliyu, Hamzat Olanrewaju, Oumar Maïga, Mamadou Kaba Traoré. 2016. The high level language for system specification: A model-driven approach to systems engineering, February 2016

Amissah, Toba, Handley, Seck, "Towards a Framework for Executable Systems Modeling: An Executable Systems Modeling Language (ESysML), SpringSim 2018

Bocciarelli, Ambrogio, Giglio, Paglia, "Model Transformation Services for MSaaS Platforms" SpringSim 2018

Charles Sanders A Call for a New Army Modeling Framework, Conference: Simulation Innovation Workshop Winter, Orlando, FL, February 2018

Folkerts H Pawletta T Deatcu C Santucci J and Capocchi L An Integrated Modeling, Simulation And Experimentation Environment In Python Based On SES/MB And DEVS, SummerSim-SCSC, 2019 July 22-24, Berlin, Germany

Kewley, R., Kester, N., & McDonnonell, J. DEVS Distributed Modeling Framework: A Parallel DEVS Implementation via Microservices. Proceedings of SpringSim, 2016

Mikel D. Petty, Eric W. Weisel Model Composition and Reuse, in: Model Engineering for Simulation, Eds: L. Zhang et al. Elsevier, 2019

MS4 Me, http://www.ms4systems.com/pages/ms4me.php

Steinman J, Walter B, Park J, Delane N. A Proposed Open System Architecture for Modeling and Simulation, Joint Program Executive Office for Chemical and Biological Defense, March. 2007

Winfrey, C. M., Baldwin, B. A, Cummings, M. A, & Ghosh, P. OSM – An Evolutionary System of Systems Framework for Modeling and Simulation, Proceedings of the 2014 Annual Simulation Symposium, 2014

Zeigler, B. P., Chungman Seo, and Doohwan Kim, "System Entity Structures for Suites of Simulation Models," Int. J. Model., Sim., Sci. Comp., vol. 4, no. 3, September 2013. DOI: 10.1142/S1793962313400060.

Zeigler, B. P., Muzy, A., and Kofman, E., Theory of Modeling and Simulation (3nd ed.), Academic Press, Elsevier 2018

Zeigler, B.P. and Nutaro J.(2016), "Towards a Framework for More Robust Validation and Verification of Simulation Models for Systems of Systems," Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, Vol. 13(1) 3–16 2015 DOI: 10.1177/1548512914568657.

AUTHOR BIOGRAPHIES

NICHOLAS KELLER is a Senior Research Scientist at RTSync Corp where he focuses on collaborative DEVS modeling. He received his PhD in computer science from Georgia State University. His email address is nicholas.keller@rtsync.com.

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 email address is zeigler@rtsync.com.

DOOHWAN KIM is interested in Predictive Analytics and Model based System Engineering based on DEVS M&S technology. He received his Ph.D. degree from the Department of Electrical and Computer Engineering of the University of Arizona. His email address is dhkim@rtsync.com.

CHASE ANDERSON is interested in BMDS and UAV simulations in addition to radar software development and clarification algorithms. He has a Bachelor's in computer science and a Bachelor's in mathematics from Athens State University. His email address is andersonchasemichael@gmail.com.

JAMES CENEY is a Senior Engineer in the Missile Defense Agency's Directorate of Engineering. He has a MS in Systems Engineering and a BS in Aeronautical & Astronautical Engineering. He works in MDA's Modeling & Simulation program. His email address is james.ceney@mda.mil.